KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA
2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim

Marwondo
Rini Melati

SMK/MAK Kelas X

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim
untuk SMK/MAK Kelas X

Penulis
Marwondo
Rini Melati

Penelaah
Irya Wisnubhadra
Asep Wahyudin

Penyelia/Penyelaras
Supriyatno
Wijanarko Adi Nugroho
Arifin Fajar Setia Utama
Adi Setyawan

Kontributor
Anjrah Mintana
Eko Prasetyo Julianto

Ilustrator
Dana Rizki Nur Adnan

Editor
Annis D. Raksanagara

Desainer
Eko Fitriono

Penerbit
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi

Dikeluarkan oleh
Pusat Perbukuan
Kompleks Kemdikbudristek Jalan RS. Fatmawati, Cipete, Jakarta Selatan
https://buku.kemdikbud.go.id

Cetakan Pertama 2023
ISBN 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

Isi buku ini menggunakan huruf Noto Serif 10/15 pt, Steve Matteson.
x, 278 hlm.: 17,6 × 25 cm.

KATA PENGANTAR

Pusat Perbukuan; Badan Standar, Kurikulum, dan Asesmen Pendidikan; Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi memiliki tugas dan fungsi mengembangkan buku pendidikan pada satuan Pendidikan Anak Usia Dini, Pendidikan Dasar, dan Pendidikan Menengah, termasuk Pendidikan Khusus. Buku yang dikembangkan saat ini mengacu pada Kurikulum Merdeka. Kurikulum ini memberikan keleluasaan bagi satuan/program pendidikan dalam mengimplementasikan kurikulum dengan prinsip diversifikasi sesuai dengan kondisi satuan pendidikan, potensi daerah, dan peserta didik.

Pemerintah dalam hal ini Pusat Perbukuan mendukung implementasi Kurikulum Merdeka di satuan pendidikan dengan mengembangkan buku siswa dan buku panduan guru sebagai buku teks utama. Buku ini dapat menjadi salah satu referensi atau inspirasi sumber belajar yang dapat dimodifikasi, dijadikan contoh, atau rujukan dalam merancang dan mengembangkan pembelajaran sesuai karakteristik, potensi, dan kebutuhan peserta didik. Adapun acuan penyusunan buku teks utama adalah Keputusan Kepala Badan Standar, Kurikulum, dan Asesmen Pendidikan Nomor 033/H/KR/2022 tentang Perubahan Atas Keputusan Kepala Badan Standar, Kurikulum, dan Asesmen Pendidikan Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Nomor 008/H/KR/2022 tentang Capaian Pembelajaran pada Pendidikan Anak Usia Dini, Jenjang Pendidikan Dasar, dan Jenjang Pendidikan Menengah pada Kurikulum Merdeka.

Sebagai dokumen hidup, buku ini tentu dapat diperbaiki dan disesuaikan dengan kebutuhan dan perkembangan keilmuan dan teknologi. Oleh karena itu, saran dan masukan dari para guru, peserta didik, orang tua, dan masyarakat sangat dibutuhkan untuk pengembangan buku ini di masa yang akan datang. Pada kesempatan ini, Pusat Perbukuan menyampaikan terima kasih kepada semua pihak yang telah terlibat dalam penyusunan buku ini, mulai dari penulis, penelaah, editor, ilustrator, desainer, dan kontributor terkait lainnya. Semoga buku ini dapat bermanfaat khususnya bagi peserta didik dan guru dalam meningkatkan mutu pembelajaran.

Jakarta, Maret 2023
Kepala Pusat,

Supriyatno
NIP 196804051988121001

PRAKATA

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan buku Dasar-dasar Pengembangan Perangkat Lunak dan Gim ini sebagai salah satu sumber belajar bagi peserta didik dalam mempelajari dasar-dasar pengembangan perangkat lunak dan gim pada kelas X Sekolah Menengah Kejuruan dengan Kompetensi Keahlian Pengembang Perangkat Lunak dan Gim.

Buku ini disusun berdasarkan Capaian Pembelajaran yang bertujuan membekali peserta didik dengan dasar-dasar pengetahuan, keterampilan, dan sikap (*hard skills* dan *soft skills*) yang diarahkan untuk mengembangkan kemampuan memahami proses bisnis di bidang industri Pengembangan Perangkat Lunak dan Gim, mengembangkan wawasan tentang perkembangan teknologi dan isu-isu global bidang Perangkat Lunak dan Gim, memahami profesi dan kewirausahan (*job profile* dan *technopreneurship*) serta peluang usaha di bidang industri Perangkat Lunak dan Gim, memahami lingkup kerja bidang Pengembangan Perangkat Lunak dan Gim; serta memahami pemrograman terstruktur dan pemrograman berorientasi objek.

Buku ini disusun sedemikian rupa dengan gaya bahasa yang mudah dipahami serta dilengkapi berbagai pengayaan pada setiap bab-nya sehingga dapat membantu peserta didik lebih memahami materi. Penulis yakin dengan beberapa materi yang ada dalam buku ini, para peserta didik kelas X dapat belajar lebih komprehensif lagi. Apalagi ditunjang dengan adanya ilustrasi dan foto-foto yang sudah penulis lampirkan. Semoga buku ini dapat memberikan motivasi dan inspirasi bagi peserta didik kelas X untuk mendalami lebih jauh semua hal tentang dasar-dasar pengembangan perangkat lunak dan gim.

Terima kasih yang sebesar-besarnya penulis ucapkan kepada semua pihak yang turut berperan membantu penyusunan buku ini. Kritik dan saran penulis harapkan agar ke depan menjadi lebih baik. Semoga buku ini bermanfaat bagi peserta didik khususnya dan pembaca pada umumnya. Salam Merdeka Belajar!

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PETUNJUK PENGGUNAAN BUKU

Tujuan Pembelajaran

Menggambarkan apa saja yang harus dicapai peserta didik dalam proses pembelajarannya.

Peta Materi

Peta materi yang disajikan di awal bab ini dapat berupa diagram yang bisa menggambarkan materi apa saja yang akan dibahas dalam bab. Peserta didik harus mencermati terlebih dahulu peta materi ini agar mendapatkan gambaran tentang materi yang akan dibahas secara luas.

Kata Kunci

Merupakan kata yang berisi informasi penting dalam materi buku, yang berfungsi untuk memudahkan peserta didik dalam menemukan informasi penting apa yang disampaikan dalam setiap bab.

Apersepsi

Di awal bab, bagian ini berisi paparan yang mengaitkan materi yang akan dipelajari dengan apa yang telah diketahui dan dialami untuk memberikan informasi awal kepada peserta didik sebelum mempelajari materi yang baru yang akan disampaikan oleh guru.

Aktivitas Belajar

Aktivitas belajar terdapat pada setiap subbab yang melatih kemampuan berpikir dan keterampilan peserta didik sesuai dengan materi.

Uji Kompetensi

Uji kompetensi ini berada pada bagian akhir bab, yang bertujuan agar peserta didik bisa mengukur kemampuannya dalam menguasai materi yang telah dibahas. Peserta didik akan diberikan soal dengan tingkat kesulitan yang berbeda mulai dari yang sederhana hingga kompleks.

Pengayaan

Pada bagian ini peserta didik akan diberikan informasi atau materi tambahan yang berkaitan dengan materi yang sedang dipelajari dan peserta didik bisa melakukan eksplorasi untuk menambah wawasan dari materi dalam bab. Informasi ini bisa berupa link atau pun literatur.

Refleksi

Pada akhir bab, guru akan mengajak peserta didik untuk memikirkan kembali apa yang sudah dipelajari serta melihat sejauh mana peserta didik mampu memahami dan menguasai materi yang diajarkan dalam pembelajaran tersebut.

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim
untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

BAB 1

Perkembangan Dunia Kerja Bidang Pengembangan Perangkat Lunak dan Gim

Tujuan Pembelajaran

Setelah mempelajari ini, diharapkan kalian dapat menjelaskan apa saja perkembangan teknologi di bidang Perangkat Lunak dan Gim, kemudian kalian juga tahu bagaimana penerapan Industri 4.0 pada bidang Perangkat Lunak dan Gim, menganalisis isu-isu penting dalam bidang Perangkat Lunak dan Gim seperti dampak positif dan negatif Gim, IoT, *Cloud Computing*, *Information Technology*, *Big Data*, dan HAKI (Hak Atas Kekayaan Intelektual), serta kalian juga bisa menjelaskan jenis-jenis profesi kewirausahaan, dan peluang usaha.

Peta Materi

Kata Kunci

♦ Gim ♦ Revolusi Industri ♦ IoT ♦ *Cloud Computing* ♦ *Information Security* ♦ *Big Data* ♦ HAKI ♦ Profesi ♦ Peluang Usaha ♦ *Vision* ♦ *Passion*

Gambar 1.1 Profesi di bidang perangkat lunak dan gim
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

Apakah kalian tahu apa saja profesi di bidang Pengembangan Perangkat Lunak dan Gim? Apa saja perangkat lunak yang saat ini banyak digunakan dalam bidang Pengembangan Perangkat Lunak dan Gim?

Apersepsi

Teknologi sudah berkembang pesat. mampu membuat perangkat lunak yang bisa menggantikan pekerjaan manusia. Beberapa profesi atau pekerjaan yang dilakukan manusia saat ini sudah bisa digantikan oleh sebuah teknologi perangkat lunak seperti *bookeeper*, *retail store* bahkan termasuk gim sudah menjadi *e-sport*. Di sisi lain, perkembangan teknologi ini membutuhkan tenaga kerja untuk menciptakannya. Salah satu profesi yang terlibat dalam pembuatan ataupun pengembangan teknologi ini adalah *programmer*, *data analyst*, *software developer*, dan *game artist*.

A. Perkembangan Teknologi pada Bidang Pengembangan Perangkat Lunak dan Gim

Aktivitas Belajar 1-1

Sebelum kalian memulai pembelajaran pada bab ini, silakan pindai kode QR di bawah ini dan coba kalian simak videonya. Selanjutnya, kalian tuliskan tujuh hal yang kalian dapatkan dalam video tersebut. Tulislah dalam sebuah kertas karton, kemudian kalian tempelkan karton tersebut di papan tulis kelas kalian dan kalian presentasikan tujuh hal yang kalian dapatkan tersebut. Lakukan ini per kelompok ya.

https://www.youtube.com/watch?v=pquPUX1EihM.

No	Hal dalam video
1.	.....
2.	.....
3.	.....
4.	.....
5.	.....
6.	.....
7.	.....

1. Perkembangan Teknologi

Sering mendengar kata teknologi? Pasti kalian sering mendengar kata teknologi dalam kehidupan sehari-hari. Teknologi diciptakan untuk memberikan kemudahan dalam kehidupan manusia. Teknologi ini akan terus berkembang seiring dengan terus berkembangnya kebutuhan manusia. Hampir setiap tahun selalu saja ada teknologi baru yang ditemukan. Selain itu, sangat banyak aspek kehidupan manusia yang menggunakan teknologi di antaranya pendidikan, medis, komunikasi, dan ekonomi.

Nah, seperti yang kalian ketahui, beberapa waktu lalu kita mengalami situasi pandemi Covid-19. Banyak aspek kehidupan mengalami perubahan, termasuk teknologi. Beberapa tren teknologi yang muncul saat pandemi Covid-19 di antaranya masyarakat melakukan belanja secara *online*, pembayaran dilakukan secara digital, pekerjaan

dan pendidikan pun dilakukan secara *online* atau *teleworking* dan menghasilkan teknologi pendukung lain. Artinya bahwa teknologi itu akan selalu berkembang seiring berkembangnya zaman dan kebutuhan manusia dalam situasi tertentu.

Lalu bagaimana perkembangan teknologi saat ini? Apakah kalian mengetahuinya? Berikut ini adalah beberapa tren teknologi yang sedang dan mungkin akan dikembangkan lagi di tahun 2023 dan sesudahnya.

a. *Artificial Intelligence* (AI)
b. *Machine Learning* (ML)
c. *Edge Computing*
d. *Quantum Computing*
e. *Cyber Security*
f. *Metaverse*
g. *Virtual Reality*
h. *Internet of Things*

Dan masih ada teknologi lainnya, ya, yang bisa kalian cari tahu lebih banyak lagi. Sebuah perangkat lunak dikembangkan oleh para ahli yang mengembangkan program komputer. Beberapa di antara mereka ada yang mengembangkan aplikasinya, perangkatnya, dan juga jaringannya. Perangkat lunak yang melibatkan beberapa ahli ini contohnya seperti permainan komputer, aplikasi bisnis, dan peralatan medis. Pada situasi tertentu para pengembang perangkat lunak ini akan menuliskan kode programnya sendiri dan memberikan instruksi kepada *programmer*.

Ada beberapa kategori dari perangkat lunak ini yang perlu kalian ketahui, di antaranya perangkat lunak sistem, perangkat lunak aplikasi, dan perangkat lunak kecerdasan buatan. Masih ada lagi kategori perangkat lunak lainnya yang bisa kalian eksplorasi lebih banyak lagi.

a. Perangkat Lunak Sistem

Perangkat lunak ini biasanya dikembangkan dan digunakan untuk melayani perangkat lunak yang lain. Tingkat kompleksitas untuk perangkat lunak ini cukup tinggi. Yang termasuk perangkat lunak jenis ini di antaranya *compiler*, *interpreter*, *driver,* dan utilitas. Perangkat lunak ini juga dibuat dan dirancang untuk menyediakan beberapa *platform* untuk perangkat lainnya, seperti perangkat lunak yang termasuk sistem operasi di antaranya Linux, Microsoft Windows, Android, mesin pencari, otomasi industri, dan perangkat lunak yang digunakan sebagai aplikasi layanan.

b. Perangkat Lunak Aplikasi

Biasanya perangkat lunak ini dibuat mengikuti proses bisnis tertentu guna mendukung operasional ataupun pengambilan keputusan dalam manajemen. Contoh dari perangkat lunak ini di antaranya aplikasi Microsoft Word dan Paint 3D.

c. Perangkat Lunak Kecerdasan Buatan

Memanfaatkan heuristik untuk memecahkan masalah kompleks yang tidak hanya melakukan perhitungan biasa atau melakukan analisis langsung. Aplikasi dalam area ini termasuk robotika, sistem pengambilan keputusan, pengenalan pola (gambar dan suara), pembelajaran mesin, teorema pembuktian, dan permainan (gim).

Setelah kalian tahu perangkat lunak di atas, lalu bagaimana dengan gim? Apakah gim termasuk perangkat lunak? Jawabannya, ya. Gim merupakan salah satu jenis perangkat lunak yang masuk dalam kategori perangkat lunak kecerdasan buatan. Karenanya gim dibangun dengan prinsip-prinsip dasar kecerdasan buatan, yaitu menyerupai proses berpikir manusia. Aplikasi apa saja yang bisa digunakan untuk membuat sebuah gim? Ada beberapa aplikasi yang bisa digunakan untuk membuat gim, di antaranya:

a. Adobe Creative Cloud
b. Autodesk
c. Unity
d. Game Maker Studio
e. Construct 3
f. Quest
g. Adventure Game Studio

Kalian bisa mencari tahu bagaimana cara kerja aplikasi-aplikasi di atas di internet. Kalau kalian mencoba membuat sebuah gim dengan salah satu aplikasi di atas juga sangat bagus. Selain aplikasi di atas, ada juga bahasa pemrograman yang digunakan untuk membuat gim, di antaranya:

a. Java
b. Kotlin
c. C# dan C++
d. Swift
e. Objective-C (OBJ-C)

Masih ada lagi bahasa pemrograman lain yang bisa kalian gunakan untuk membuat gim seperti Javascript dan Python. Sebagai permulaan, kalian bisa membuat gim yang sederhana saja dulu untuk membiasakan

menggunakan sintaks-sintaks bahasa pemrograman tersebut. Nanti jika sudah mahir, kalian bisa mengembangkan gim sederhana tadi secara perlahan.

Gim memiliki beberapa genre. Apakah kalian tahu apa saja genre gim tersebut? Nah, salah satu yang kalian sebutkan mungkin ada dalam genre berikut ini:

a. *Action*
b. *Adventure*
c. *Action-Adventure*
d. RPG
e. *Simulation*
f. *Strategy*
g. *Sports*
h. *Idle Gaming*

Setelah kalian mengetahui genrenya, kita lihat ada beberapa gim yang banyak diminati saat ini. Menurut artikel di katadata.co.id, di antaranya:

a. Minecraft
b. PUBG
c. Apex Legends
d. Fortnite Battle Royale
e. Counter Strike
f. Heartstone
g. League of Legends
h. Among Us
i. Mobile Legends
j. Free Fire
k. Call of Duty Warzone

Jika kalian ingin membuat sebuah gim, ada dua elemen yang bisa kalian rancang, yaitu elemen formal dan elemen dramatis. Apa maksud dari kedua elemen tersebut? Elemen formal adalah elemen yang membentuk struktur dari sebuah gim. Jika tidak ada elemen ini, maka belum bisa disebut gim. Yang termasuk elemen formal di antaranya:

- Prosedur
- Pemain
- Tujuan
- Aturan
- Konflik
- Sumber Daya
- Batasan
- Hasil

Sedangkan elemen dramatis adalah elemen yang melibatkan emosi pemain dalam gim tersebut. Elemen ini biasanya terdiri dari:

- Pemain
- Tantangan
- Karakter
- Motif.

B. Penerapan Industri 4.0 pada Bidang Perangkat Lunak dan Gim

Aktivitas Belajar 1-2

Pada bagian ini kalian akan diminta mengisi tabel KWL. KWL mempunyai kepanjangan **K** sebagai ***Know***, **W** sebagai ***Want to know***, dan **L** sebagai ***Learn***. Untuk mengisi tabel KWL tersebut, amatilah gambar berikut.

Gambar 1.2 Revolusi Industri
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

1. Apa yang kalian ketahui tentang gambar di atas?
2. Tuliskan apa yang ingin kalian ketahui tentang industri 4.0?

K (Apa yang ingin diketahui) *Diisi di awal pembelajaran*	W (Apa yang ingin dipelajari) *Diisi di awal pembelajaran*	L (Apa yang sudah dipelajari) *Diisi di awal pembelajaran*

Setelah mengisi tabel KWL di atas, silakan kalian diskusikan jawaban kalian bersama teman-teman lain, untuk saling berbagi informasi.

3. Revolusi Industri

Menurut Aptika Kominfo dalam artikel Revolusi Industri 4.0, Revolusi Industri 4.0 merupakan kolaborasi antara teknologi siber dan otomatisasi. Karena penerapannya sudah berpusat pada otomatisasi, maka keterlibatan tenaga manusia dalam prosesnya dapat berkurang.

Perlu kalian ketahui, hal ini pun akan berimbas pada bertambahnya efektivitas dan efisiensi di lingkungan kerja.

Apakah kalian tahu teknologi yang menjadi pilar utama dalam mengembangkan industri siap digital? Ya salah satu yang kalian sebutkan mungkin ada dalam penjelasan berikut:

a. *Internet of Things* (IoT)

Kalian pasti sudah sering mendengar istilah IoT. Ada empat komponen yang terintegrasi dalam IoT, yaitu perangkat sensor, konektivitas, pemrosesan data, dan antarmuka pengguna.

Contoh aplikasi IoT di Indonesia salah satunya adalah eFishery yang mampu menggabungkan sistem pemberian pakan ikan mulai dari kapan harus memberi pakan, berapa banyak takaran pakan yang akan diberikan, sampai memantau pertumbuhan atau perkembangan ikan yang ada dalam tambak. Semua ini bisa dilakukan secara *online* menggunakan ponsel pintar. Contoh lainnya Gowes (IoT untuk *bike sharing*), Qlue (IoT untuk *smart city*), dan Hara (IoT untuk pangan dan pertanian).

b. *Big Data*

Istilah *big data* ini pun sudah tidak asing lagi bagi kalian. *Big data* biasanya digunakan untuk pengambilan keputusan berdasarkan data yang sudah dianalisis ataupun untuk menentukan strategi bisnis yang lebih baik. Salah satu aplikasi *big data* yang digunakan saat ini adalah *Fraud Detection* pada bank. *Fraud Detection* ini biasanya digunakan untuk melakukan identifikasi atau mendeteksi lebih awal kemungkinan terjadinya sebuah kecurangan dalam transaksi atau penipuan, termasuk penyuapan yang dilakukan sesudah atau sebelum transaksi.

c. *Artificial Intelligence* (AI)

Kalian pasti sering mendengar istilah AI. Istilah ini sudah sangat umum. AI merupakan teknologi yang memungkinkan mesin (benda mati) mempunyai kecerdasan layaknya manusia dan bisa diatur sesuai keinginan manusia. Semakin banyak data yang dianalisis oleh AI ini maka akan semakin baik pula AI ini membuat prediksi. Aplikasi *chatbot* dan pengenalan wajah (*face recognition*) merupakan salah satu contoh penerapan AI.

d. *Cloud Computing*

Komputasi awan (*cloud computing*) ini biasanya menjadikan internet sebagai pusat pengelolaan data dan aplikasinya. Setiap penggunanya akan diberikan hak akses untuk menggunakan *cloud* dan dapat mengkonfigurasi servernya melalui internet. Perlu kalian ketahui, jenis model layanan komputasi awan ini adalah:

- *Cloud Software as a Service* (SaaS),
- *Cloud Platform as a Service* (PaaS),
- *Infrastructure as a Service* (IaaS),

Sebagai informasi, di Indonesia ada beberapa produk dari *cloud computing* ini, di antaranya:

- K-Cloud;
- CloudKilat;
- Dewaweb;
- IDCloudHost;
- FreeCloud.

e. *Addictive Manufacturing*

Dalam industri manufaktur, pemanfaatan mesin pencetak 3D adalah hal yang sudah sangat banyak dilakukan. Di mana produk yang dibuat akan dibentuk menjadi benda yang nyata sesuai dengan ukuran dan bentuk sebenarnya dengan desain dan skala tertentu.

Berikut ini kalian akan mengetahui bagaimana perkembangan dari sebuah Revolusi Industri mulai dari revolusi industri 1.0 sampai revolusi industri 4.0:

a. Revolusi Industri 1.0

Ini terjadi pada abad ke-18, yang ditandai dengan ditemukannya mesin uap untuk proses produksi barang dan juga transportasi.

b. Revolusi Industri 2.0

Ini terjadi pada awal abad ke-20, yang ditandai dengan ditemukannya tenaga listrik. Adanya perubahan masyarakat agraris menjadi masyarakat indsustri menjadi ciri dari revolusi ini.

c. Revolusi Industri 3.0

Jika revolusi pertama itu dipicu oleh adanya mesin uap, revolusi kedua dipicu oleh adanya ban berjalan dan listrik, maka pada revolusi ketiga, mesin dapat digerakkan secara otomatis, seperti komputer dan robot. Komputer pun mulai banyak menggantikan manusia sebagai operator dan pengendali produksi.

d. Revolusi Industri 4.0

Revolusi keempat ini mungkin merupakan revolusi yang sangat sering kalian dengar. Dalam revolusi ini, terjadi penggabungan teknologi siber dan teknologi otomatisasi.

C. Isu-Isu Penting dalam Bidang Perangkat Lunak dan Gim (PLG)

Aktivitas Belajar 1-3

Nah, sekarang coba kalian cari tahu tentang hal-hal berikut dalam bidang PLG. Apa saja itu? Coba kalian uraikan dalam tabel berikut ini, ya:

Materi	Uraian	Contoh Penerapan	Kelebihan dan kekurangan
Gim	.....	.....	.....
IoT	.....	.....	.....
Cloud Computing	.....	.....	.....
Information Technology	.....	.....	.....
Big Data	.....	.....	.....

Setelah kalian mengisi tabel di atas, presentasikan hasil eksplorasi kalian tersebut di depan kelas untuk saling berbagi informasi ya.

1. Gim

Istilah gim ini sudah tidak asing lagi bagi kalian. Mungkin kalian adalah salah satu pengguna atau pemain gim. Dalam pembahasan sebelumnya sudah disampaikan tentang aplikasi pembuat gim dan jenis-jenis gim.

Bermain gim itu sangat menyenangkan bagi mereka yang memainkannya. Namun kalian harus tahu bahwa bermain gim itu mempunyai beberapa dampak. Jika bermain gim mengikuti ketentuan

dan sesuai umur maka akan memberikan dampak positif. Dampak negatif akan muncul, jika bermain gim ini dilakukan secara berlebihan.

Berikut ini beberapa dampak positifnya:

a. Memicu aktivitas asah otak.
b. Melatih sportivitas.
c. Menambah pengetahuan.
d. Melatih kemampuan dalam menyelesaikan masalah.
e. Meningkatkan kreativitas.

Selain dampak posistif, ada juga dampak negatif dari bermain gim, di antaranya:

a. Memiliki resiko terhadap kesehatan mata.
b. Menurunnya daya konsentrasi.
c. Memberi gangguan motorik.
d. Memunculkan masalah dalam komunikasi.
e. Memicu tingkat keagresifan anak.

Lalu bagaimana mencegah agar bermain gim tidak menjadi kecanduan? Beberapa cara ini bisa diterapkan saat bermain gim:

a. Mengatur atau memberi batasan waktu.
b. Melakukan pendampingan pada anak.
c. Melakukan kegiatan yang kreatif dan menyenangkan bersama anak, agar anak tidak selalu bermain gim.
d. Menerapkan aturan yang disepakati bersama, lengkap dengan konsekuensi jika terjadi pelanggaran.

2. IoT

Internet of Things atau lebih dikenal dengan IoT adalah istilah yang sangat sering kalian dengar. IoT memungkinkan semua benda yang ada di sekitar kalian terhubung ke perangkat lain dan saling berkomunikasi, bertukar data, dan bisa mengontrol suatu alat dengan adanya sensor dan koneksi internet.

IoT dapat menimbulkan beberapa dampak, di antaranya:

a. Peningkatan ekonomi.
b. Terjadi perubahan pada standarisasi ataupun kebiasaan dari sebuah perusahaan, dari konvensional ke penerapan IoT.

c. Karena selalu terhubung ke internet, maka perlu dikondisikan keamanan penggunaan dan penyimpanan data.

d. Kurang memadainya komunikasi antar beberapa teknologi yang terkait dengan IoT

e. Penggunaan IoT akan membutuhkan penyimpanan data yang sangat besar dan bisa menggunakan *Big Data*, hanya saja diperlukan energi yang sangat besar untuk mengoperasikannya.

f. Bisa terjadi kekacauan sistem jika terjadi kesalahan dalam penggunaan IoT.

Selain dampak di atas, ada beberapa dampak lain yang ditimbulkan dalam penerapan IoT ini secara positif, di antaranya pekerjaan yang dilakukan bisa menjadi lebih efisien, lingkungan sekitar bisa lebih bersih, terlahirnya beberapa bisnis baru seperti penggunaan sensor dan kamera, dan memungkinkan lahirnya pengetahuan-pengetahuan baru. Dampak negatif lain yang timbul di antaranya terjadinya PHK dan semakin maraknya pencurian atau pembobolan data.

3. *Cloud Computing*

Kalian pasti sudah sering mendengar istilah *Cloud Computing*. Atau mungkin kalian tidak sadar bahwa saat ini kalian sedang menggunakan *Cloud Computing*. *Cloud Computing* atau komputasi awan adalah penggunaan teknologi komputer (komputasi) yang dikombinasikan dengan pengembangan teknologi berbasis internet ('awan'). Yang dimaksud awan disini merupakan metafora untuk internet yang umum digunakan untuk memvisualisasikan atau menggambarkan jaringan komputer dan internet. *Cloud Computing* ini biasanya digunakan untuk menyimpan dan mengakses data di internet, sehingga kalian tidak perlu lagi khawatir kehilangan data-data penting, seperti saat kalian menyimpan data di dalam *hard disk* komputer. Kalian juga dapat mengakses *file* tersebut kapan saja di *Cloud Computing* selama memiliki koneksi internet. Lalu seperti apa kelebihan dan kekurangan dari *Cloud Computing* ini?

Beberapa kelebihan *Cloud Computing* di antaranya:

a. Biaya yang digunakan akan lebih hemat.

b. Adanya kemudahan dalam melakukan manajemen bisnis.

c. Memudahkan kalian mengaksesnya di manapun dengan perangkat *smartphone*.

d. Kapasitas untuk menyimpan data hampir tidak terbatas.
e. *Software* bisa terintegrasi secara otomatis.
f. Tersedianya fitur untuk *back up* data dan pemulihan data.

Masih ada keuntungan lain yang bisa kalian dapatkan dan rasakan saat menggunakan *Cloud Computing* ini.

Selain keuntungan di atas, ada juga hal-hal yang harus kalian ketahui sebagai sebuah tantangan dalam menggunakan *Cloud Computing*, di antaranya:

a. Risiko keamanan yang harus kalian pahami saat akan membagikan informasi penting dan sensitif kepada pihak ketiga, yaitu penyedia layanan *Cloud Computing*.
b. Terjadinya *downtime* karena koneksi internet lambat, perawatan rutin dari penyedia layanan.
c. Adanya pembatasan *bandwidth*. Pengguna yang memerlukan *bandwidth* besar akan dikenakan biaya, biasanya cukup mahal.

4. *Information Technology*

Sudah tidak asing dengan istilah *Information Technology* (IT) atau dalam bahasa Indonesianya Teknologi Informasi (TI). Apa itu IT/TI?. Manusia banyak sekali terbantu dan dimudahkan oleh TI ini, di antaranya membantu dalam membuat, menyimpan, melakukan perubahan dalam informasi dan mengkomunikasikannya serta menyebarluaskan informasi.

TI ini selalu dikaitkan dengan komputer dalam melakukan pengolahan informasinya dan juga berkaitan dengan penggunaan alat atau perangkat yang saling berkomunikasi. TI ini juga memiliki beberapa fungsi di antaranya menangkap (*capture*), mengolah (*processing*), menghasilkan (*generating*), menyimpan (*storage*), mencari kembali (*retrieval*), dan juga sebagai transmisi. Beberapa komponen yang membentuk IT di antaranya *hardware*, *software*, *infoware*, *fireware,* dan *brainware*.

Penggunaan dan perkembangan TI ini akan memberikan beberapa dampak, di antaranya:

a. Kalian yang memiliki kemampuan teknologi yang tinggi akan lebih mudah memperoleh dan mengakses informasi, sehingga akan kaya informasi.

b. Penyebaran informasi dan arus data yang cukup tinggi, bahkan bisa melintas geografis, bisa menjadi sesuatu yang mengancam stabilitas keamanan suatu negara.

c. Penyalahgunaan hak cipta dalam informasi elekronik, misalnya pengkopian kode program tanpa izin pemiliknya.

d. Sensor informasi perlu dilakukan karena tidak semua informasi bisa disebarluaskan. Pemerintah harus bisa membuat peraturan atau menyensor penyebaran informasi di kalangan masyarakatnya.

Kalian bisa mencari tahu dampak lain dari TI ini. Kalian harus bisa bersikap bijak dalam memanfaatkan perkembangan TI, agar tidak merugikan diri sendiri dan orang lain.

5. *Big Data*

Seperti yang telah disebutkan sepintas sebelum ini, *big data* merupakan kumpulan data yang volume ataupun kecepatannya sangat tinggi, sehingga sulit untuk disimpan, diproses atau dikelola, dan dianalisis hanya menggunakan basis data yang konvensional ataupun menggunakan alat pemroses data.

Banyak sekali data yang diproses dan dihasilkan setiap harinya, bahkan setiap menitnya, dari penggunaan teknologi saat ini, baik itu data berupa konten, video, gambar, panggilan telepon, pengunjung laman web, atau pengguna aplikasi. *Big data* ini sangat berpotensi mendukung sebuah aplikasi yang menggunakan data menjadi lebih cerdas.

Ciri khas dari *Big Data* ini dikenal dengan istilah 5V, yaitu:

a. *Volume*
b. *Velocity*
c. *Variety*
d. *Veracity*
e. *Value*

Coba kalian cari tahu tentang 5V, agar kalian lebih paham lagi tentang *Big Data*. Lalu bagaimana dengan kelebihan dan kekurangan dari *Big Data*? Berikut ini beberapa kelebihan dari *Big Data*:

a. Bisa digunakan untuk melakukan analisis tren pasar.

b. Kegagalan yang terjadi bisa diketahui secara *real time*.

c. Anomali yang terjadi dalam suatu proses bisnis dapat diketahui dan dideteksi.

d. Lebih hemat waktu dan biaya dalam penggunaannya

Dan beberapa kekurangan dari *Big Data* ini adalah:

a. Banyaknya kekhawatiran akan keamanan data pribadi.
b. Perlu tenaga ahli atau infrastruktur yang mumpuni untuk memaksimalkan pemanfaatan *Big Data*.

Nah, setelah mengetahui kelebihan dan kekurangan dari *Big Data* ini, ke depannya kalian harus bisa menyiapkan kemampuan terbaik untuk memanfaatkan teknologi ini dengan lebih bijak dan lebih bermanfaat bagi banyak orang. Kalian harus terus menambah wawasan dan memperdalam pemahaman tentang *Big Data* dan implementasinya, juga meng-*upgrade* kemampuan atau kompetensi kalian dalam teknologi ini.

D. Isu-isu dalam Permasalahan Hak Atas Kekayaan Intelektual (HAKI)

Aktivitas Belajar 1-4

Tuliskan apa yang ingin kalian ketahui tentang HAKI (Hak Atas Kekayaan Intelektual) dengan mengisi tabel **KWL** berikut ini, kemudian diskusikan hasil jawaban kalian bersama teman-teman di kelas untuk saling berbagi informasi

1. Apa yang kalian ketahui tentang gambar di atas?
2. Tuliskan apa yang ingin kalian ketahui tentang industri 4.0?

K (Apa yang ingin diketahui) *Diisi di awal pembelajaran*	W (Apa yang ingin dipelajari) *Diisi di awal pembelajaran*	L (Apa yang sudah dipelajari) *Diisi di awal pembelajaran*

Setelah kalian mengisi tabel **KWL** di atas, silakan diskusikan jawaban kalian bersama teman-teman lain, untuk saling berbagi informasi.

1. Hak Atas Kekayaan Intelektual (HAKI)

Apa yang kalian rasakan jika hasil karya kalian yang sudah dibuat dengan susah payah kemudian digandakan tanpa izin? Kesal, kan? Pasti. Sebaliknya, bisa jadi kita menggunakan hasil karya orang lain dan kemudian diakui sebagai karya kita. Terkadang tanpa kita sadari, ada

bagian hasil karya orang lain yang kita gunakan untuk menghasilkan karya kita. Bolehkah? Mari kita cari jawabannya bersama-sama.

Pernahkah kalian mendengar istilah *copyright*? Istilah ini sering kita jumpai dalam berbagai produk perangkat lunak maupun produk-produk teknologi yang lain. *Copyright* adalah istilah dalam Bahasa Inggris yang digunakan untuk menandai kepemilikan atas suatu ciptaan (karya). Dalam Bahasa Indonesia, istilah ini dikenal dengan nama Hak Atas Kekayaan Intelektual (HAKI) atau secara umum dikenal dengan istilah Hak Cipta.

Apa itu kekayaan intelektual? Kalian mungkin sering mendengar istilah ini. Kekayaan intelektual adalah hasil dari olah pikir berupa produk atau proses yang bisa digunakan dan dimanfaatkan oleh manusia yang kemudian menimbulkan hak atas hasil olah pikir tersebut. Dengan dengan kata lain kekayaan intelektual ini bisa dikatakan hak yang bisa kalian gunakan untuk menikmati apa yang sudah kalian hasilkan secara ekonomis berupa kreativitas intelektual.

Lalu bagaimana dengan istilah HAKI? Pernah mendengarnya juga, kan. Nah, HAKI ini adalah hak atas kekayaan yang timbul karena kemampuan intelektual manusia. Kalian tentu tahu manfaat dari HAKI ini, di antaranya:

a. Dunia usaha akan memiliki perlindungan dari pemalsuan ataupun penyalahgunaan karya intelektual mereka oleh pihak lain yang tidak bertanggung jawab. Selain itu akan ada respon positif ataupun citra baik bagi perusahaan yang telah memiliki perlindungan hukum dalam HAKI-nya.
b. Adanya penjaminan hukum bagi individu atau kelompok sehingga mereka akan terhindar dari kecurangan-kecurangan atau pemalsuan yang dilakukan oleh pihak lain.
c. Bagi pemerintah sendiri akan mendapatkan citra baik dari WTO (*World Trade Organization*) karena menerapkan HAKI dan juga akan ada penerimaan devisa dari pendaftaran atas hak kekayaan intelektual.
d. Adanya kepastian hukum bagi pemegang hak atas kekayaan intelektualnya.

Setelah kalian mengetahui manfaat dari HAKI ini, lalu apakah kalian juga tahu apa saja yang termasuk ke dalam kekayaan intelektual? Berikut ini adalah beberapa di antaranya:

a. Hak Cipta, hak ini timbul secara otomatis setelah karya yang kalian buat terwujud dalam dunia nyata dan sudah kalian publikasikan, yang disebut juga hak eksklusif pencipta. Contohnya buku dan karya tulis, musik dan lagu, program komputer.

b. Merek, bisa berupa logo, kata, suara, bentuk 3D, atau hologram. Merek ini biasanya sebuah tanda yang membedakan sebuah produk atau jasa. Contoh merek mi instan atau logo perusahaan.

c. Desain Industri, suatu kreasi atau bentuk yang kalian buat kemudian dikonfigurasi dengan komposisi garis dan warna sehingga memberikan kesan yang estetis pada produk tersebut, biasanya berbentuk 2 dimensi atau 3 dimensi.

d. Paten, hak eksklusif ini diberikan kepada kalian sebagai penemu atau inventor atas hasil dari invensi kalian di bidang teknologi. Contohnya vaksin covid-19, sedotan fleksibel, atau teknik konstruksi cakar ayam.

Tidak hanya jenis ini saja yang termasuk dalam kekayaan intelektual, ya. Kalian bisa mengeksplorasi lebih jauh lagi tentang kekayaan intelektual ini. Sebagai informasi untuk kalian, berikut ini adalah undang-undang terkait kekayaan intelektual:

a. Hak Cipta diatur dalam Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

b. Merek diatur dalam Undang-Undang Nomor 20 Tahun 2016 tentang Merek dan Indikasi Geografis.

c. Desain Industri diatur dalam Undang-Undang Nomor 31 Tahun 2000 tentang Desain Industri.

d. Paten diatur dalam Undang-Undang Nomor 13 Tahun 2016 tentang Paten.

2. Lisensi Perangkat Lunak

a. **Perangkat Lunak Berbayar** – jika kalian ingin menggunakan perangkat lunak jenis ini, maka kalian akan dikenakan biaya dengan nominal tertentu. Walaupun kalian sudah membeli perangkat lunak ini, tetap saja kalian tidak diberikan lisensi untuk menyebarluaskannya. Tindakan ini termasuk ilegal. Beberapa perangkat lunak yang berbayar di antaranya Adobe Photoshop, Microsoft Office, dan Microsoft Windows.

b. ***Freeware*** – ini adalah perangkat lunak yang bisa kalian gunakan secara gratis dan tanpa batas waktu. Perangkat lunak ini biasanya dikembangkan untuk komunitas tertentu. Perangkat lunak ini tetap mempertahankan hak ciptanya, agar bisa melakukan *update* perangkat lunak terbaru. Beberapa perangkat lunak yang termasuk *freeware* di antaranya Google Chrome dan Mozilla Firefox.

c. ***Free Software*** – kalian bisa membeli perangkat lunak ini terlebih dahulu, kemudian ke depannya kalian bisa menggandakan, memodifikasi, atau mendistribusikannya. Contoh *software* yang termasuk ke dalam *free software* di antaranya Sistem Operasi Linux, Ubuntu, dan Libre Office.

d. ***Shareware*** – kalian bisa memperoleh perangkat lunak ini secara gratis, karena memang disediakan gratis untuk keperluan tertentu seperti jika kalian ingin melakukan uji coba dengan fitur yang terbatas dan masa waktu penggunaannya pun terbatas, biasanya 15 sampai 30 hari saja. Biasanya perangkat lunak ini diberikan secara gratis agar kalian sebagai pengguna bisa mempelajarinya terlebih dahulu sebelum kalian membeli lisensi dengan versi yang lengkapnya. Contoh dari *software* ini di antaranya Internet Download Manager dan Winrar.

e. ***Malware*** – perangkat lunak ini dikenal sebagai perusak. Sehingga akan berakibat bahaya jika kalian atau pengguna lainnya menyalahgunakan perangkat lunak ini. Perangkat lunak ini biasanya digunakan untuk menyusup dan merusak sebuah sistem jaringan komputer. Beberapa perangkat lunak yang termasuk malware di antaranya *spyware* (perangkat lunak pengintai), *adware* (perangkat lunak untuk iklan yang tidak jujur), virus komputer, dan *software* lainnya yang dibuat dengan tujuan merugikan.

f. ***Open Source Software*** – jika kalian menggunakan sebuah perangkat lunak yang kode sumbernya bisa kalian pelajari, modifikasi, ditingkatkan dan disebarluaskan, maka perangkat lunak tersebut termasuk *open source software*. Biasanya kalian bisa memperolehnya secara gratis dan digunakan oleh komunitas tertentu untuk dikembangkan dengan lisensi GPL (*General Public License*). termasuk dalam perangkat lunak ini adalah Linux yang mempunyai fungsi seperti Microsoft Windows.

g. ***Firmware*** – perangkat lunak ini sifatnya paten sehingga tidak bisa kalian modifikasi atau kembangkan meskipun terdapat masalah

dalam fungsinya. Perangkat lunak ini biasanya sudah menyatu dengan perangkat kerasnya, sehingga dianggap bukan perangkat lunak seutuhnya.

Berikut ini adalah beberapa pelanggaran yang sering terjadi pada kekayaan intelektual yang perlu kalian ketahui:

a. Menjiplak karya tulis.
b. Menjiplak konten yang ada di internet.
c. Melakukan pembajakan sebuah *software*.
d. Pelanggaran hak cipta lagu.

Masih ada lagi bentuk-bentuk pelanggaran yang bisa kalian cari tahu untuk menambah wawasan dan kewaspadaan, agar kalian tidak terlibat dalam pelanggaran tersebut.

E. Jenis-Jenis Profesi, Kewirausahaan, dan Peluang Usaha

Aktivitas Belajar 1-5

Kalian tentu banyak mengetahui dan mendapat informasi tentang pengusaha-pengusaha sukses di bidangnya. Nah, coba kalian cari tahu siapakah orang yang menjadi inspirasi kalian untuk bisa menjadi sukses. Lakukan wawancara langsung dengan orang tersebut agar kalian bisa mendapatkan gambaran usaha atau strategi apa yang mereka lakukan untuk menjadi pengusaha sukses. Kemudian bagikan hasil wawancara kalian di depan kelas untuk saling berbagi insipirasi dengan teman-teman yang lain.

1. *Job Profile* dan *Technopreneurship*

Technopreneurship adalah istilah yang sangat sering kalian dengar, apalagi jika kalian memang sudah terbiasa beraktivitas di bidang bisnis dan wirausaha. *Technopreuneurship* ini biasanya mereka wirausahawan di zaman sekarang yang memanfaatkan perkembangan teknologi untuk membuat dan menciptakan suatu produk atau karya, bahkan menciptakan sebuah solusi dan selalu berinovasi dengan karyanya, sehingga mereka bisa mengubah cara kerja sesuatu tersebut yang sebelumnya masih dilakukan secara tradisional.

Lalu apakah kalian bisa menjadi seorang *technopreuner*? Tentu saja bisa, yang penting kalian tidak pernah takut untuk mencoba atau bereksperimen, kemudian kalian mempunyai kemampuan dalam *problem solving*, kalian harus mempunyai kemampuan dalam mengambil keputusan, dan bisa memilih strategi terbaik dalam mengimplementaskan sebuah solusi.

Berikut ini adalah beberapa profil profesi di bidang PLG:

a. Perusahaan Startup.

Beberapa profesi dalam bidang Startup ini di antaranya:

1) CEO (*Chief Executive Officer*), seorang pimpinan dalam sebuah startup yang bertindak mewakili perusahaan.
2) CTO (*Chief Technical Officer*), penanggung jawab terhadap kualitas akhir karena dia yang terlibat langsung dalam mengelola timnya dan proses pelaksanaannya.
3) CFO (*Chief Finance Officer*), penanggung jawab dalam bidang keuangan, pembukuan, termasuk didalamnya menggalang dan menganggarkan dana perusahaan.
4) CMO (*Chief Marketing Officer*), penanggung jawab dalam menentukan strategi pemasaran perusahaan.
5) COO (*Chief Operating Officer*), penanggung jawab terhadap semua urusan organisasi perusahaan termasuk di dalamnya adalah pengaturan karyawan dan operasional kantor.

b. Dalam Perencanaan PLG

1) *System Analyst*, mempunyai kemampuan menganalisis sistem.
2) *Data Analyst*, mempunyai kemampuan menganalisis berbagai jenis data.
3) *Game Designer*, mempunyai kemampuan dalam melakukan kegiatan pra produksi seperti membuat *mock u*p gim, *story board*, dan menentukan genre gim.

c. Dalam Implementasi PLG

1) UI *Designer*, merancang tampilan sebuah aplikasi agar terlihat menarik.
2) UX *Designer*, merancang sebaik dan sebagus mungkin sebuah aplikasi agar *user* merasakan kesenangan dan manfaat dari aplikasi.

3) *Game Artist*, membuat sebuah gim agar terlihat menarik baik secara visual ataupun dalam penggunaannya.

4) *Front End Developer*, merancang antarmuka atau tampilan sebuah halaman web.

5) *Back End Developer*, memberikan aksi untuk melakukan sesuatu dalam sebuah web.

6) *Software Engineer*, menerapkan prinsip-prinsip atau teknik dalam mengembangkan perangkat lunak.

d. Dalam Pengujian PLG

Quality Assurance atau tester adalah profesi dalam pengujian PLG yang melakukan pengujian atau uji coba suatu produk serta penjamin mutu dari produk yang dihasilkan.

2. *Personal Branding*

Pernah mendengar tentang *personal branding*? Mungkin kata ini sudah tidak asing di telinga kalian. Tetapi, apakah kalian tahu apa itu *personal branding*? Mengapa kita perlu membahasnya pada bab ini?

Personal branding itu adalah citra diri, yang menggambarkan identitas pribadi. *Personal branding* ini salah satu strategi yang bisa gunakan untuk membangun *brand* atau citra diri. Ketika mem-*brand*-kan diri, maka sebenarnya sedang "menjanjikan" sesuatu yang ada dalam diri kepada orang lain dengan tujuan untuk menimbulkan respon dari orang lain tentang nilai dan kualitas diri. Saat sudah *brand*-kan diri, maka harus konsisten dalam menepati "janji" tesebut. Misalnya saat mem-*brand*-kan diri sebagai seorang *programmer* profesional, maka harus bisa membuktikan kualitas diri dalam pekerjaan tersebut.

Umumnya *personal branding* ini berkaitan dengan kesan dan reputasi kalian. Kesan ini mencakup semua yang kalian katakan atau kenakan. Cara orang memperlakukan kalian, bahasa tubuh, dan kepribadian kalian juga termasuk kesan yang di-*brand* orang lain terhadap kalian.

Sedangkan reputasi di sini erat kaitannya dengan wawasan pengetahuan yang kalian miliki termasuk pengalaman unik kalian. Biasanya mereka yang memiliki wawasan luas, keterampilan yang tinggi, disertai banyaknya pengalaman baik, reputasi mereka akan dinilai baik oleh orang lain. Nah, kesan dan reputasi inilah yang nantinya akan jadi *brand value* kalian, yang akan selalu diingat oleh banyak orang.

Personal branding ini bisa kalian jadikan kunci untuk mengembangkan karier jika kalian ada di kalangan profesional. Sedangkan jika kalian baru lulus dari sekolah, maka personal branding ini bisa membantu kalian untuk mendapatkan pekerjaan yang kalian impikan. Bagi kalian yang merupakan seorang pengusaha atau *entrepreuneur*, *personal branding* ini harus kalian ciptakan dengan sangat baik karena hal ini akan sangat membantu sekali dalam menarik para investor agar mereka mau bekerja sama dengan kalian.

Lalu apa saja manfaat dari *personal branding* ini yang kalian ketahui? Berikut ini beberapa manfaat dari *personal branding*:

a. Kalian akan dilihat sebagai seorang yang ahli di bidangnya.
b. Meningkatnya reputasi kalian di mata orang-orang sekitar.
c. Kalian akan memperoleh penghargaan dari orang lain.
d. Banyak orang yang akan merekomendasikanmu walaupun tidak diminta. Hal ini bisa terjadi jika *brand*-mu sudah terbentuk.

3. Peluang Usaha (*Vision* dan *Passion*)

Sering, ya, mendengar tentang peluang usaha? Seperti apa peluang usaha itu dan bagaimana melihat sebuah peluang usaha? Nah, penjelasan berikut ini mungkin sama dengan pendapat kalian.

Jika kalian ingin melihat sebuah peluang usaha, kalian harus mencari tahu dulu apa, sih, yang jadi kebutuhan masyarakat saat ini dan yang akan datang. Untuk memenuhi kebutuhan tersebut, ada pengamatan lingkungan usaha yang harus dilakukan secara menyeluruh di antaranya faktor ekonomi, persaingan, pasar, teknologi, dan geografi.

Peluang juga bisa muncul dari sebuah ide yang kalian ciptakan, jika kalian mau mengevaluasi setiap peluang yang ada secara terus-menerus dengan cara membuat sesuatu yang baru dan berbeda. Peluang juga bisa kalian peroleh dengan cara kalian harus mempunyai kemampuan dan wawasan untuk menghasilkan suatu produk atau jasa.

Kalian harus bisa menyaring dan menilai kelayakan dari sebuah ide dan peluang. Ini tidak mudah, lo. Tetapi sangat penting. Pada intinya, saat kalian sedang menilai atau mengidentifikasi sebuah peluang, sebenarnya kalian sedang menentukan risiko dan hasil/imbalan dari faktor berikut ini:

a. Lamanya masa dari suatu peluang.
b. Bagaimana keadaan suatu pasar atau industri.
c. Apa saja tujuan dari pengusaha dan kompetensi yang dimilikinya.
d. Kondisi lingkungan.
e. Persaingan.

Ketika teknologi, khususnya teknologi informasi berkembang pesat dan telah menjadi bagian besar dari kehidupan manusia, maka muncul pula beberapa peluang yang bisa dimanfaatkan sebagai peluang bisnis. Ada banyak ide bisnis yang dapat dieksekusi. Nah, dari mana kalian mendapatkan ide bisnis?

Pada dasarnya menemukan ide bisnis tidaklah sulit. Kebanyakan para pemula kesulitan menemukan ide bisnis karena tidak tahu caranya. Padahal ide bisnis itu banyak sekali dan bertebaran di mana-mana. Ingat, prinsip dasar dari teknologi informasi adalah "Apa yang bisa dilakukan secara otomatis?". Kalian tahukah caranya? Yuk, mari kita lihat dari mana saja ide bisnis itu berasal.

a. Berangkat dari hobi

Hobi menjadi salah satu pilihan terbaik untuk memulai ide bisnis. Mulailah dengan melihat apa yang kalian senangi. Misalkan kalian suka mengoleksi sepatu. Jika dipandang oleh orang biasa mungkin hanya dilihat sebagai fashion atau gaya. Namun di balik itu, pernahkah kalian terpikir untuk:

- Menjadi reseller?
- Membuka jasa reparasi sepatu?
- Menjual zat pembersih sepatu?
- Membuka laundry sepatu?
- Bahkan menjual sepatu bekas?

Hal ini juga bisa kalian lakukan pada perangkat lunak. Misalkan jika kalian gemar bermain gim, pernahkah kalian berpikir untuk:

- Menjual *diamond* atau pernak-pernik yang ada dalam gim?
- Menjual perlengkapan untuk kenyamanan bermain gim?

b. Berangkat dari malas

Ide bisnis juga dapat berasal dari kemalasan kalian. Pernahkah kalian berpikir mengapa tercipta mesin cuci? Ya, mesin cuci diciptakan karena kita malas mencuci secara manual. Nah, sekarang perhatikan kembali,

- Mengapa perangkat lunak transportasi *online* dikembangkan?
- Mengapa perangkat lunak perpesanan dikembangkan?

c. Berangkat dari sulit

Kesulitan dalam mengerjakan suatu hal juga merupakan ide sekaligus peluang bisnis yang bisa kalian manfaatkan. Kalian pernah mengalami kesulitan mencari suatu tempat atau alamat? Ya, itulah salah satu peluang yang ditangkap oleh pengembang perangkat lunak yang kemudian mengembangkan perangkat lunak navigasi. Pernahkah kalian terpikir, mengapa mesin ekskavator diciptakan?

Berikut beberapa peluang bisnis dalam pengembangan perangkat lunak yang bisa dijadikan inspirasi:

a. *Web development*
b. SaaS dan *Mobile Apps Development*
c. Layanan *digital marketing*
d. VAR – *Value-Added Reseller*
e. Konsultan *Big Data*
f. Kreator *Theme* dan *Plugin* CMS

Lalu bagaimana dengan *passion* dan *vision* kalian dalam bekerja? Umumnya *passion* itu berhubungan dengan sesuatu yang sangat menarik menurut kalian untuk dilakukan atau dipelajari walaupun kalian tidak mendapatkan imbalan dan kalian sangat antusias untuk melakukannya. Biasanya *passion* ini akan memberikan dampak dan manfaat bagi orang lain.

Mereka yang bekerja sesuai dengan *passion*-nya biasanya akan menemukan kesuksesan. Karena, kalian akan bekerja dengan perasaan dan hati yang senang dan bahagia. Memiliki *passion* dalam bekerja juga bisa membuat kalian lebih menikmati setiap proses dalam menjalani hidup dan mencapai tujuan dari pekerjaan kalian. Lalu bagaimana

cara menemukan *passion* kalian? Berikut ada beberapa cara yang bisa kalian lakukan:

a. Kalian bisa memikirkan hal-hal atau kegiatan yang membuat kalian merasakan kebahagiaan dan hal yang membuat kalian senang membicarakannya.
b. Coba tuliskan apa saja sifat dalam diri kalian yang merupakan kekuatan kalian.
c. Tanyakan pada diri apa yang menjadi impian dan cita-cita kalian sewaktu kecil.
d. Coba rasakan dan renungkan apakah pekerjaan yang kalian jalani membuat kalian merasa bahagia dan bersemangat dalam mengerjakannya.

Setelah kalian mengetahui tentang *passion*, berikut ini akan dijelaskan tentang *vision*. Istilah *passion* dan *vision* ini terkadang membuat kalian bingung membedakannya. Biasanya orang memahami *vision* itu adalah sebuah visi yang ingin dicapai sedangkan *passion* adalah suatu semangat yang mendorong kalian untuk melakukan sesuatu untuk mencapai suatu visi.

Visi di sini akan menggambarkan bagaimana cara kalian untuk mencapai dan memiliki sesuatu yang kalian impikan agar bisa terwujud, dalam waktu singkat ataupun panjang. Visi ini harus bisa kalian visualisasikan dan tergambar dalam pikiran kalian dan juga harus realistis untuk memudahkan dan memastikan kalian dapat mencapainya.

Uji Kompetensi

Coba kalian jawab pertanyaan di bawah ini untuk mengetahui apakah kalian sudah menguasai teori tentang Perkembangan Dunia Kerja di Bidang PLG . Kalian bisa memilih opsi yang menurut kalian benar!

1. Ada beberapa perangkat yang berkaitan dengan IoT, di antaranya Internet, *Cloud*, Sensor, *Microcontroller*, *Artificial Inteligence*. Manakah yang berkaitan dengan data dalam IoT?
   a. *Sensor*
   b. *Network*
   c. *Cloud*

d. *Internet*

e. *Microcontroller*

2. Dalam gim kalian sering mendengar istilah *Action*, *Strategy*, *Simulation*, *Adventure,* dan RPG. Istilah-istilah tersebut dikenal sebagai ...

   a. *Simulasi gim*

   b. *Genre gim*

   c. *Aplikasi gim*

   d. *Software gim*

   e. *Judul* gim

3. Coba kalian perhatikan tabel berikut ini:

Artificial Intelligence
Metaverse
Virtual Reality
IoT
Big Data
Cloud Computing

   Data pada tabel di atas menggambarkan tren teknologi saat ini yang bisa dikembangkan pada tahun-tahun yang akan datang. Berdasarkan tabel di atas, manakah teknologi yang memanfaatkan jumlah data yang besar yang dikelola dengan teknologi terkini?

   a. *Cloud Computing*

   b. *Big Data*

   c. *IoT*

   d. *Virtual Reality*

   e. *Metaverse*

4. Hak yang timbul secara otomatis setelah karya yang kalian buat terwujud dalam dunia nyata dan sudah kalian publikasikan disebut juga hak eksklusif pencipta. Hak ini disebut dengan ...

   a. Hak *Cipta*

   b. *Hak Paten*

   c. *Hak Karya*

   d. *Hak Desain*

   e. *Hak* Merek

5. Salah satu strategi yang bisa kalian gunakan untuk membangun citra diri dikenal dengan istilah
   a. *Passion*
   b. *Vision*
   c. *Personal Branding*
   d. *Brand*
   e. *Pe*luang usaha
6. Banyak sekali profesi yang bisa didapatkan dalam bidang PLG. Salah satu profesi dalam implementasi PLG yaitu ...
   a. *System analyst*
   b. *Game Artist*
   c. *Data Analyst*
   d. *Front End Developer*
   e. *CE*O
7. Berikut ini adalah ciri khas dari Big data yaitu 5V, adalah ....
   a. V*olume, Velocity, Variety, Veracity, Value*
   b. *Volume, Velocity, Variety, Veracity, Vission*
   c. *Visual, Velocity, Variety, Veracity, Value*
   d. *Volume, Velocity, Variety, Vibration, Value*
   e. *V*olume, *Vocal*, *Variety*, *Veracity*, *Value*
8. Dalam revolusi ini, terjadi penggabungan teknologi siber dan teknologi otomatisasi....
   a. *Revolusi Industri 1.0*
   b. *Revolusi Industri 2.0*
   c. *Revolusi Industri 3.0*
   d. *Revolusi Industri 4.0*
   e. *Re*volusi Industri 5.0
9. Perhatikan beberapa informasi berikut ini:
   - ► Memicu aktivitas asah otak
   - ► Melatih sportivitas
   - ► Menambah pengetahuan baru
   - ► Melatih kemampuan menyelesaikan masalah
   - ► Meningkatnya suatu kreativitas

Informasi di atas menggambarkan ....

a. D*ampak positif IoT*
b. *Dampak positif Cloud Computing*
c. *Dampak positif Metaverse*
d. *Dampak positif Big Data*
e. *Da*mpak positif Gim

10. Perangkat lunak ini dikenal sebagai perusak...
    a. *Malware*
    b. *Free software*
    c. *Share software*
    d. *Open sourse software*
    e. *Firmware*

Pengayaan

Berikut ini adalah sumber materi yang bisa kalian gunakan untuk menggali lebih dalam lagi materi terkait pengembangan perangkat lunak dan gim.

1. https://www.ibm.com/topics/software-development
2. https://itchronicles.com/what-is-software-development/
3. Tentang kewirausahaan (cari referensi)
4. Tentang HAKI (cari referensi)

https://www.ibm.com/topics/software-development

https://itchronicles.com/what-is-software-development/

Refleksi

Kalian bisa memberikan tanda (√) pada kotak yang kalian anggap sesuai! Setelah kalian mempelajari bab ini, bagaimanakah penguasaan kalian terhadap materi-materi berikut?

No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Perkembangan Teknologi			
2.	Penerapan Industri 4.0			
3.	Isu-isu penting bidang PLG			
4.	Isu-isu penting tentang HAKI			
5.	Profesi, Kewirausahaan, dan Peluang Usaha			

1. Berdasarkan materi-materi yang sudah kalian pelajari di atas, manakah bagian yang kalian sangat sukai? Jelaskan alasannya!
2. Manfaat apa saja yang kalian peroleh untuk kehidupan sehari-hari, setelah kalian mempelajari semua materi-materi diatas?
3. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

BAB 2

Kesehatan, Keselamatan Kerja, dan Lingkungan Hidup (K3LH)

Tujuan Pembelajaran

Setelah mempelajari ini, diharapkan kalian dapat menerapkan Kesehatan, Keselamatan Kerja, dan Lingkungan Hidup (K3LH) dan budaya kerja industri untuk praktik-praktik kerja yang aman, kemudian menerapkan K3LH dan budaya kerja industri terkait bahaya-bahaya di tempat kerja, prosedur-prosedur dalam keadaan darurat, penerapan budaya kerja industri (Ringkas, Rapi, Resik, Rawat, Rajin) dan melakukan pencegahan kecelakaan kerja dan prosedur kerja.

Peta Materi

Kata Kunci

♦ Ringkas ♦ Rapi ♦ Resik ♦ Rawat ♦ Rajin ♦ Kecelakaan Kerja

Gambar 2.1 Kecelakaan kerja yang umum terjadi
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

Apakah kalian tahu bagaimana suatu kecelakan kerja bisa terjadi atau bagaimana bisa kesehatan kita terganggu akibat bekerja?

Apersepsi

Kalian pasti sering mendengar istilah K3LH, ya? K3LH ini singkatan dari Kesehatan, Keselamatan Kerja, dan Lingkungan Hidup. K3LH ini selalu diterapkan di manapun kalian berada, baik itu di lingkungan umum, di lingkungan rumah, di tempat kerja, dan juga di sekolah. Penerapan K3LH ini sangat penting dilakukan demi menjaga dan mencegah terjadinya hal-hal buruk pada diri kita yang ada kaitannya dengan kesehatan kita, kegiatan kita, pekerjaan kita, dan lingkungan sekitar kita. Banyak hal yang akan kalian pelajari dalam menerapkan K3LH ini di antaranya bagaimana melakukan praktik kerja yang aman, mengetahui bahaya-bahaya di tempat kerja dan pencegahannya, prosedur dalam keadaan darurat, juga mempelajari tentang budaya kerja 5R.

Aktivitas Belajar 2-1

Tuliskan apa yang ingin kalian ketahui tentang Budaya kerja Industri, bahaya di tempat kerja dan budaya 5R dengan mengisi tabel ***Know***, ***Want to Know***, ***and Learned*** **(KWL)** berikut ini, kemudian diskusikan hasil jawaban kalian bersama teman-teman di kelas untuk saling berbagi informasi.

K (Apa yang ingin diketahui) *Diisi di awal pembelajaran*	W (Apa yang ingin dipelajari) *Diisi di awal pembelajaran*	L (Apa yang sudah dipelajari) *Diisi di awal pembelajaran*

A. Praktik-Praktik Kerja yang Aman

Aktivitas Belajar 2-2

Dalam aktivitas kali ini kalian diminta untuk membuat sebuah poster himbauan bagaimana melakukan praktik aman di laboratorium komputer. Kemudian silakan kalian tempelkan poster tersebut di laboratorium komputer. Kerjakan secara berkelompok.

Dalam bidang perangkat lunak dan gim, komputer merupakan perangkat teknologi yang paling sering kalian gunakan dalam melakukan praktik di laboratorium komputer. Kalian akan lebih sering membuat program dengan komputer. Saat kalian sedang mengetik kode program depan komputer, maka saat itu juga kondisi kalian bisa berada dalam keadaan tidak aman. Salah satunya adalah terpapar radiasi komputer atau risiko tersengat listrik.

Oleh karena itu, ada beberapa hal yang harus kalian perhatikan saat bekerja atau melakukan praktik dengan komputer, untuk mengurangi dampak buruk komputer pada tubuh dan kesehatan kalian, diantara hal yang harus kalian perhatikan yaitu bagaimana menyambungkan perangkat, menyalakan komputer, dan mematikan komputer. Pada saat kalian melakukan praktik membuat kode program, ada beberapa perangkat yang terhubung dengan komputer kalian, di antaranya *mouse*, *keyboard*, *printer*, alat *scan*, *speaker*, dan kabel LAN. Kalian harus memastikan semua perangkat ini terhubung dengan tepat pada dudukannya untuk menghindari terjadinya kesalahan prosedur dan membuat aktivitas praktik kalian terhambat.

Setelah kalian memastikan perangkat terhubung dengan baik pada komputer, berikutnya yang kalian lakukan adalah memperhatikan prosedur menyalakan dan mematikan komputer dengan benar.

Berikut ini adalah cara menghidupkan komputer yang benar:

1. Kalian harus menyalakan *stabilizer* terlebih dahulu. Jika tidak menggunakan *stabilizer*, kalian bisa langsung ke langkah berikutnya.
2. Kemudian kalian tekan tombol *power* pada CPU, dan tunggu komputer selesai *booting* dan menjalankan sistem operasi.
3. Jika komputer kalian meminta *user name* dan *password*, maka kalian masukkan *user* dan *password* yang sesuai.
4. Setelah itu di layar monitor kalian akan muncul tampilan *desktop*, dan *pointer mouse*, yang berarti komputer kalian sudah siap dipakai.

Setelah kalian mengetahui cara menghidupkan komputer, maka berikut ini adalah cara mematikan komputer yang benar yang juga harus bisa kalian lakukan dengan baik:

1. Kalian tutup atau *close* semua aplikasi yang telah kalian gunakan.
2. Gerakkan *pointer mouse* ke atas tombol *start* [] biasanya berbentuk jendela atau *window*, kemudian klik *power* dan akan muncul beberapa pilihan untuk mematikan komputer.
3. Lalu kalian pilih dan klik *Shut Down*.
4. Tunggu sampai komputer benar-benar mati.
5. Lalu matikan *stabilizer*, jika ada.

Selain prosedur terkait perangkat dan komputer, ada beberapa hal lain yang juga sangat penting untuk kalian terapkan dalam melakukan praktik dengan komputer yaitu pastikan kalian tidak makan, atau menaruh sisa makanan dan minuman kalian di sekitar perangkat komputer. Hal ini bisa sangat tidak aman dan berisiko bahaya saat kalian praktik. Misalnya sisa makanan kalian akan dikerubungi semut, semut ini bisa saja masuk ke dalam perangkat komputer, membentuk sarang semut, dan menyebabkan komponen rusak, dan tidak berfungsi dengan baik. Bisa juga sisa air minum kalian tumpah atau menetes mengenai perangkat komputer kalian dan mengakibatkan perangkat kalian rusak atau memicu terjadinya arus pendek listrik saat kalian menyalakan komputer yang terkena atau tertumpah sisa air minum kalian tersebut.

B. Bahaya-Bahaya di Tempat Kerja

Aktivitas Belajar 2-3

Dalam aktivitas ini kalian dapat memperhatikan dan menganalisis risiko bahaya yang mungkin terjadi di lingkungan sekolah. Setelah itu, kalian lakukan tindakan pencegahan dengan memberikan tanda-tanda bahaya pada tempat yang dianggap memiliki risiko bahaya. Kegiatan ini kalian lakukan secara berkelompok. Hasil analisis kalian silakan ditulis dalam tabel ini:

Kondisi/Situasi/Tempat berpotensi bahaya	Tanda-tanda bahaya yang digunakan
.....	
.....	
.....	

1. Kondisi Bahaya di Tempat Kerja

Dalam melaksanakan suatu pekerjaan akan selalu timbul risiko atau bahaya, hal ini bisa membahayakan diri kalian ataupun orang lain. Dalam situasi normal ataupun sibuk, bahaya di tempat kerja bisa saja terjadi seperti cairan permbersih tumpah, benda jatuh, barang pecah, dan lain sebagainya.

Kecelakaan juga bisa terjadi di tempat kerja, tidak hanya karena musibah saja. Kecelakaan pun bisa terjadi karena beberapa faktor, baik yang datang dari diri sendiri maupun orang lain, seperti tindakan yang tidak aman pada diri sendiri atau tindakan tidak aman pada lingkungan kerja.

Berikut ini adalah bentuk tindakan tidak aman yang harus kalian ketahui, di antaranya:

a. Melakukan pekerjaan dengan terburu-buru atau tergesa-gesa.
b. Tidak menggunakan pelindung diri yang sudah disediakan.
c. Dengan sengaja kalian melanggar peraturan keselamatan yang diwajibkan.
d. Melakukan pekerjaan sambil bercanda, bergurau, dan lain sebagainya.

Sedangkan bentuk tindakan tidak aman dari lingkungan kerja yang perlu kalian ketahui di antaranya:

a. Masih ada mesin-mesin rusak yang tidak diberi pengamanan, konstruksi kurang aman, bising, dan alat-alat kerja yang kurang baik dan rusak.
b. Lingkungan kerja yang tidak aman bagi manusia seperti becek, licin, ventilasi atau pertukaran udara, bising atau suara keras-keras, suhu tempat kerja, tata ruang kerja/kebersihan, dan lain-lain.

2. Jenis-Jenis Bahaya di Tempat Kerja

Kalian pasti sudah sering mendengar tentang bahaya-bahaya di tempat kerja. Sama halnya saat kalian sedang melakukan praktik di laboratorium komputer. Bahaya ini merupakan faktor-faktor yang berhubungan dengan pekerjaan yang bisa menimbulkan kecelakaan. Beberapa bahaya yang bisa ditimbulkan saat bekerja dengan komputer di antaranya tersengat aliran listrik dan radiasi komputer.

Bahaya ini bisa dibagi dua kategori, yaitu bahaya yang bersifat khusus dan umum. Bahaya yang berkaitan dengan kerugian materil di tempat kerja dalam sarana dan prasarana.Ini dikategorikan dengan bahaya yang bersifat khusus. Misalnya tidak tersedianya peralatan keamanan atau perlindungan saat bekerja.

Sedangkan bahaya yang menimbulkan kerugian non materil dalam proses kerja seperti pegawai kurang istirahat saat bekerja, atau prosedur kerja yang tidak diterapkan dengan baik. Ini merupakan kategori bahaya umum.

Kalian juga harus mengetahui seperti apa jenis bahaya di tempat kerja atau akibat kerja, berikut ini beberapa pengklasifikasiannya:

a. Jenis bahaya menurut kecelakaannya di antaranya terjatuh, tertimpa benda jatuh, terjepit oleh benda, tertumbuk atau terkena benda-benda, dan pengaruh suhu yang tinggi.
b. Jenis bahaya berdasarkan penyebabnya di antaranya bisa disebabkan oleh mesin, alat angkut, atau peralatan lainnya.
c. Jenis bahaya menurut sifat luka atau kelainan di antaranya patah tulang, keseleo, luka bakar, dan amputasi.

3. Tanda-Tanda Bahaya di Tempat Kerja

Setelah kalian mengetahui kondisi bahaya dan jenis bahaya di tempat kerja, berikutnya kalian juga harus mengetahui mengenai tanda bahaya yang bisa digunakan di tempat kerja yang bertujuan untuk mencegah dan meminimalisir terjadinya kecelakaan kerja atau bisa digunakan sebagai tanda peringatan awal terjadinya kecelakaan atau adanya situasi darurat.

Tanda bahaya ini bisa berupa alat yang dibunyikan dan juga bisa dinyalakan secara otomatis atau manual. Berbagai macam tanda bahaya di antaranya alarm untuk kebocoran gas, kebakaran, pencurian atau bunyi sirine ambulans.

Selain tanda-tanda bahaya, ada juga yang namanya tanda peringatan bahaya yang berbentuk tanda atau kode tertentu yang biasanya digunakan sebelum bahaya terjadi. Nah, bentuk tanda peringatan tersebut bisa berupa gambar, lampu warna, atau kata-kata.

Gambar 2.2 Tanda dilarang mengaktifkan *handphone.*
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

Contoh tanda peringatan berupa gambar, seperti pada gambar di samping ini.

Kalian pasti tahu tanda bahaya berupa lampu warna. Ya, biasanya kalian akan melihatnya pada lampu rambu lalu lintas yang terdiri dari warna merah, hijau, dan kuning. Kalian pasti sudah tahu arti dari setiap warnanya. Selain itu ada juga lampu yang berkelip seperti lampu sirine.

Selain tanda peringatan berupa gambar dan warna, ada juga tanda peringatan berupa kata-kata, seperti "PINTU DARURAT", "RAWAN KECELAKAAN", atau bisa juga kata-kata "JALAN PELAN-PELAN, BANYAK ANAK-ANAK".

Gambar 2.3 Tanda rawan kecelakaan.
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

C. Prosedur-Prosedur dalam Keadaan Darurat

Aktivitas Belajar 2-4

Pada aktivitas kali ini, coba kalian melakukan pengamatan di lingkungan sekitar kalian, mengidentifikasi kemungkinan-kemungkinan bisa terjadinya keadaaan darurat, dan bagaimana kalian menanggulanginya. Kemudin kalian presentasikan hasil identifikasi kalian di depan kelas.

Setelah kalian mempelajari tentang bahaya di tempat kerja, selanjutnya kalian harus mengetahui seperti apa prosedur yang harus kalian ikuti saat mengalami keadaan darurat. Prosedur ini merupakan tata cara atau acuan yang bisa dijadikan pedoman dalam melakukan kegiatan kerja. Ada beberapa aspek yang memang harus diperhatikan dalam membuat sebuah prosedur keadaan darurat, di antaranya:

- Mengidentifikasi jenis-jenis bahaya yang memungkinkan terjadinya kecelakaan kerja.
- Selalu tersedia perlengkapan darurat seperti penyediaan Alat Pemadam Api Ringan (APAR), alarm kebakaran, dan alat P3K.
- Adanya tim tanggap darurat K3LH.
- Melakukan sosialisasi prosedur tanggap darurat.
- Mengadakan pelatihan terkait keadaan darurat.

Selain prosedurnya, ada juga kategori untuk keadaan yang terdiri dari tiga, yaitu kategori satu, dua, dan tiga. Kategori satu adalah suatu keadaan yang bisa menimbulkan kecelakaan skala kecil dan sangat berpotensi mengancam nyawa manusia. Sedangkan keadaan darurat kategori dua ini berpotensi menimbulkan kecelakaan besar yang pada saat itu petugas tim dan peralatan pencegahan pun tidak mampu untuk mengendalikan keadaan tersebut, mengakibatkan adanya banyak korban, serta memerlukan bantuan dari luar.

Keadaan darurat kategori tiga adalah suatu keadaan yang bisa dikategorikan seperti bencana dahsyat, keadaan di mana diperlukan adanya bantuan dan koordinasi tingkat nasional bahkan internasional.

Jika ada suatu tindakan yang salah atau kondisi tidak aman yang disebabkan oleh suatu kelalaian yang mengakibatkan timbulnya keadaan darurat, maka kalian harus mengetahui apa yang harus dilakukan saat itu dan bagaimana menyikapinya. Berikut ini adalah sikap yang bisa kalian terapkan saat terjadinya keadaan darurat, di antaranya:

1. Cepat dan tanggap.
2. Harus memiliki kesadaran dalam mencegah terjadinya keadaan darurat yang bisa menimbulkan kecelakaan.
3. Selalu tenang dalam menghadapi keadaan darurat dan harus bisa bertindak cepat dalam menangani keadaan tersebut.

Pengetahuan yang harus dimiliki ketika menghadapi Keadaan darurat antara lain:

1. Mengetahui jenis dan tanda peringatan bahaya yang dapat terjadi disekitar anda.
2. Mengidentifikasi keadaan yang dapat menimbulkan bahaya.
3. Mengenali karakteristik orang yang mencurigakan.
4. Mengetahui prosedur keadaan darurat di sebuah lokasi dimana anda berada.

Sebagai tenaga kerja, berikut ini adalah keterampilan yang harus dimiliki saat menghadapi keadaan darurat:

1. Selali mengikuti dan mematuhi setiap tanda-tanda peringatan bahaya yang ada di sekitar kalian.
2. Harus bisa menentukan langkah-langkah cepat apa yang bisa diambil dalam keadaan darurat.
3. Mampu mengoperasikan dan menggunakan perlengkapan keadaan darurat.

D. Penerapan Budaya Kerja Industri (Ringkas, Rapi, Resik, Rawat, Rajin)

Aktivitas Belajar 2-5

Pada bagian ini kalian akan diminta mengelompokkan benda-benda berikut berdasarkan fungsinya: Penggaris, sapu, gelas, kemoceng, lap pel, sendok, garpu, buku, tisu, gunting, ember, pengki, pulpen, dan piring. Kemudian kalian isi tabel berikut ini:

Fungsinya sebagai:		
Alat	**Alat .....**	**Alat .....**
Nama bendanya:		
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....
Dst.	Dst.	Dst.

1. Pengertian 5R

Dalam dunia indsustri sudah sangat sering kalian dengar yang namanya budaya kerja. Saat ini budaya kerja yang sudah sangat banyak diterapkan dikenal dengan istilah 5R, yaitu Ringkas, Rapi, Resik, Rawat, dan Rajin. Di Jepang, budaya kerja seperti ini dikenal dengan 5S yaitu *Seiri*, *Seiton*, *Seiso*, *Seiketsu,* dan *Shitsuke*.

a. Ringkas (*Seiri*)

Kalian harus bisa memisahkan atau memilah segala seuatu yang diperlukan atau tidak diperlukan di tempat kerja. Kalian harus mengetahui benda apa saja yang masih bisa dipakai atau sudah tidak bisa digunakan lagi, mengetahui mana barang yang perlu disimpan dan cara penyimpanannya agar bisa diakses dengan mudah.

b. Rapi (*Seiton*)

Di sebuah perusahaan, menempatkan atau meletakkan benda-benda tertentu tidak bisa dilakukan dengan sembarangan atau asal-asalan, karena ini akan menghambat saat mencari dan menemukan kembali barang tersebut. Kalian harus membiasakan diri menyimpan atau menaruh barang kembali pada tempatnya. Hal ini akan mempermudah kalian menemukan kembali barang tersebut dengan cepat saat dibutuhkan.

c. Resik (*Seiso*)

Resik ini berkaitan dengan kebersihan, baik itu kebersihan tempat kerja, kebersihan mesin atau peralatan kerja, atau barang-barang lain agar tidak berdebu kotor dan juga bau.

Menjaga kebersihan ini harus menjadi kebiasaan semua orang mulai dari pimpinan perusahaan sampai pelaksana atau operator.

d. Rawat (*Seiketsu*)

Rawat ini prinsipnya adalah bisa mempertahankan atau menjaga hasil yang telah dicapai sebelumnya dengan melakukan standarisasi, misalnya.

e. Rajin (*Shitsuke*)

Setiap pribadi karyawan harus bisa dan memiliki sikap rajin. Ini berarti sebuah pengembangan kebiasaan positif di lingkungan kerja. Prinsip dari rajin ini bisa dikatakan seperti berikut, "lakukanlah apa yang harus dilakukan dan janganlah melakukan apa yang tidak boleh dilakukan".

2. Manfaat 5R dan Penerapannya

Menerapkan budaya 5R ini banyak sekali manfaatnya, di antaranya menjadikan suatu proses lebih cepat selesai, kualitas dari suatu produk bisa lebih terjaga, dan menjadikan pribadi menjadi lebih disiplin.

Bentuk penerapan 5R yang bisa kalian lakukan dalam kegiatan sehari-hari di antaranya menyimpan peralatan kerja pada tempatnya dan sesuai jenisnya, mengelompokkan barang sesuai fungsinya, menggunakan alat praktik mengikuti standar kebersihan, dan memelihara lingkungan setelah melakukan praktik di laboratorium.

Berikut ini beberapa kegiatan yang bisa kalian lakukan dalam menerapkan 5R:

a. Ringkas

1) Kalian bisa melakukan pengelompokan barang praktik di laboratorium komputer.
2) Memilah mana subjek dan objek dalam praktikum.
3) Kalian bisa membuat lembar ceklis untuk melihat kondisi barang yang ada di ruang praktikum.
4) Tidak berlebihan dalam menggunakan alat dan bahan praktikum.

b. Rapi

1) Kalian bisa mengatur tempat praktik dan meletakkan atau menyimpan alat praktik pada tempat yang seharusnya.
2) Bisa dengan mudah mengambil dan menemukan dan menyimpan kembali alat dan bahan praktik di tempatnya.
3) Kalian menjaga agar tidak tercecernya alat dan bahan praktik.
4) Kalian melakukan pengklasifikasian dalam penyimpanan barang menurut fungsi dan frekuensi pemakaian praktikum.
5) Tempat penyimpanan alat praktikum harus disesuaikan dengan fitur alat, penggunaannya apakah di tempat terbuka, atau tempat tertutup.
6) Memberikan label pada semua alat praktik untuk memudahkan identifikasi alat tersebut terkait penyimpanan dan penggunaannya.

c. Resik

1) Setiap alat atau bahan praktikum harus memenuhi prosedur dan standar kebersihan.
2) Lakukan pembersihan pemeriksaan alat praktik secara berkala.
3) Selalu menjaga agar lingkungan kerja atau praktik selalu bersih.
4) Selalu tersedianya alat kebersihan di setiap tempat kerja atau ruang praktik.

d. Rawat

1) Selalu memelihara lingkungan pembelajaran praktik agar tetap bersih, rapi, dan ringkas.
2) Menggunakan standarisasi yang telah ditentukan dalam pemakaian sarana dan prasarana praktik.
3) Lakukan optimalisasi ruangan belajar praktik agar selalu terjaga dengan baik.

e. Rajin

1) Terlaksananya standarisasi yang telah dibuat.
2) Penerapan budaya kerja disiplin yang tinggi dalam menggunakan sarana dan prasarana praktik.
3) Melakukan kebiasaan positif dalam praktik kerja di sekolah.
4) Melaksanakan standar penggunaan alat praktik.
5) Efisiensi dalam menggunakan alat dan sarana praktikum.
6) Meningkatkan produktivitas dan evaluasi berkala.

E. Pencegahan Kecelakaan Kerja dan Prosedur Kerja

Aktivitas Belajar 2-6

Coba kalian cari tahu tentang bagaimana risiko bekerja di depan komputer dan bagaimana langkah pencegahannya. Kemudian hasil eksplorasi kalian silakan ditulis dalam tabel berikut:

Risiko yang terjadi	Langkah pencegahan
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
Dst.	Dst.

1. Kecelakaan Kerja

Apakah kalian pernah mendengar kalimat kecelakan kerja? Risiko kecelakaan kerja memang tidak bisa dihindari, tetapi bisa dicegah atau diminimalisir. Apakah kalian tahu apa itu kecelakaan kerja? Kecelakaan kerja itu ini adalah suatu kejadian yang tentu saja tidak pernah kalian harapkan, kejadian yang tidak pernah kalian duga kapan datangnya dan kapan terjadinya. Kejadian ini pasti menimbulkan beberapa kerugian, di antaranya kerugian secara material atau pun kerugian secara moril. Bentuk kerugian yang bisa ditimbulkan oleh kecelakaan kerja ini di antaranya kematian, kerusakan alat praktik, dan cacat secara fisik.

Hal-hal berikut ini merupakan beberapa penyebab terjadinya kecelakaan kerja di antaranya adalah kecerobohan, alat praktik yang sudah rusak, atau sudah tidak aman lagi jika digunakan. Bentuk kecelakaan kerja yang bisa ditimbulkan saat kalian melakukan kegiatan dengan komputer di antaranya adalah terkena sengatan listrik. Hal ini bisa saja disebabkan karena kecerobohan dan tidak mengikuti prosedur penggunaan komputer yang benar.

Jenis kecelakaan yang sering terjadi pada saat bekerja langsung antara lain: terkena sengatan arus listrik, tersambar petir, tertimpa benda jatuh, tertumbuk atau terkena benda, terpeleset, terjatuh, terjepit oleh benda kerja, gerakan yang melebihi kemampuan, pengaruh suhu tinggi, kontak dengan bahan-bahan berbahaya, dan lain sebagainya.

Kecelakaan kerja biasanya terjadi karena dua faktor utama, yaitu tindakan pekerja yang ceroboh dan kondisi alat/tempat yang sudah rusak atau tidak aman. Contoh kecelakaan kerja yang disebabkan oleh kecerobohan pekerja yaitu bercanda atau berkelakar yang berlebihan saat bekerja. Sedangkan contoh kecelakaan kerja karena kondisi kerja yang tidak aman yaitu alat kerja yang sudah rusak, lingkungan yang berbahaya, dan prosedur kerja yang salah.

2. Prosedur Kerja

Salah satu hal yang bisa kalian lakukan untuk mencegah atau meminimalkan terjadinya kecelakaan kerja yaitu mengetahui dan mengikuti prosedur K3LH yang berlaku baik di tempat praktik seperti laboratorium komputer atau di tempat kerja.

Bentuk prosedur K3LH yang bisa kalian terapkan di antaranya saat kalian melakukan praktik menggunakan komputer.

a. Penggunaan komputer

Saat meggunakan komputer, ada beberapa hal yang harus kalian perhatikan, yaitu:

- **Pencahayaaan**, dalam hal ini adalah pencahayaan di dalam ruangan komputer. Pencahayaan ini perlu diperhatikan agar kegiatan kerja atau kegiatan praktik yang kalian lakukan dapat berjalan baik dan efektif. Kita memerlukan sistem pencahayaan yang sangat baik, tepat, dan harus sesuai dengan kebutuhan di dalam suatu ruangan.

- **Mengatur posisi tubuh**, maksudnya adalah saat melakukan praktik dengan komputer maka posisi tubuh ini harus kalian perhatikan, mulai dari posisi kepala, bahu, pinggang, paha, sampai kaki. Posisi tubuh kalian harus rileks, seperti yang tampak pada gambar di bawah ini. Hal ini sangat penting agar kalian merasa nyaman saat praktik dan kesehatan kalian pun terjaga saat melakukan praktik dengan komputer.

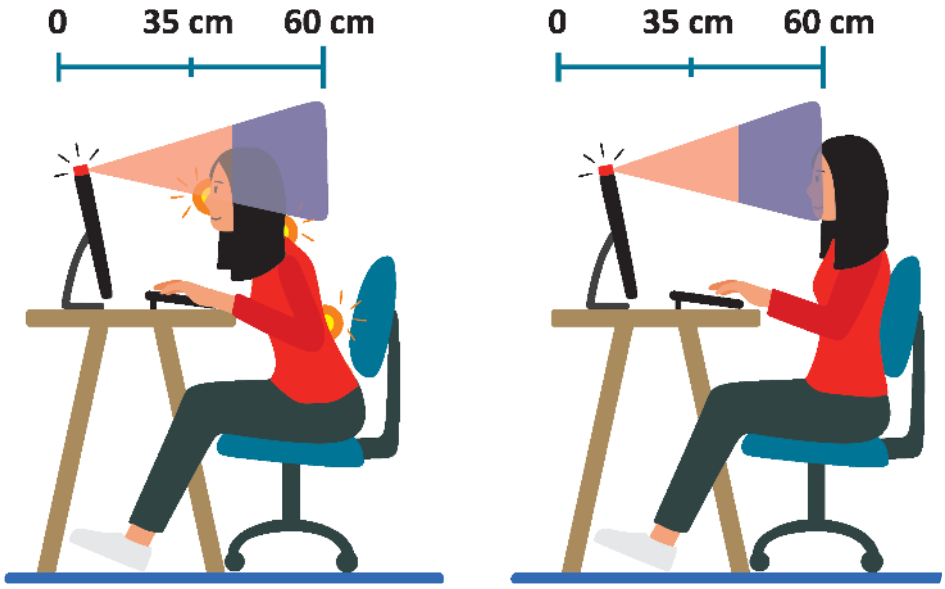
Gambar 2.4 Posisi duduk yang benar
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

- **Mengatur posisi duduk**. Berikut ini beberapa hal yang bisa kalian perhatikan:
  1) Paha dalam posisi horizontal dan punggung bagian bawah atau pinggang tersandar.
  2) Hindari posisi duduk terlalu di ujung kursi. Bila kursi kurang dapat diatur, bagian bawah punggung dapat dibantu dengan diberi bantal.
  3) Telapak kaki harus dapat menumpu secara rata di lantai ketika duduk dan ketika menggunakan *keyboard*. Apabila tidak dapat maka kursinya mungkin terlalu tinggi. Solusinya dengan memanfaatkan penyangga kaki.

4) Perlu untuk mengubah posisi duduk selama bekerja karena duduk dalam posisi tetap dalam jangka lama bisa menimbulkan ketidaknyamanan.

Gambar 2.5 Posisi duduk dan berdiri yang benar
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

- **Mengatur posisi mata**. Nah, hal ini sering kita abaikan atau lupa karena terlalu asyik bekerja atau praktik dengan komputer. Perlu kalian ketahui bahwa jarak pandang mata kita pada layar monitor itu jangan terlalu jauh atau terlalu dekat, karena hal ini bisa menyebabkan mata kita lelah. Kalian bisa melihat gambar di bawah ini. Mengikuti aturan jarak pandang mata pada layar monitor ini akan membuat kalian bekerja atau melakukan praktik dengan nyaman dan kesehatan mata terjaga.



Gambar 2.6 Jarak pandang yang benar di depan monitor
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

- **Mengatur posisi lengan dan siku**. Sama halnya seperti posisi mata, posisi lengan dan siku ini pun sering kita abaikan saat bekerja dengan komputer. Memperhatikan posisi lengan dan siku ini akan memberikan kenyamanan saat bekerja dan juga akan menjaga kesehatan tubuh kita. Posisi yang baik adalah jika saat mengetik atau menggunakan *mouse* dan *keyboard* terasa nyaman pada tubuh. Umumnya posisi lengan yang baik adalah jika tangan berada disamping badan dan siku membentuk sudut lebih besar dari 90 derajat.
- Mengatur perangkat

  1) Monitor

  - Mengatur posisi monitor agar pantulan cahaya dari lampu atau sumber lainnya dapat dikurangi.
  - Kalian bisa mengguanakan filter monitor untuk mengurangi radiasinya mengenai mata.
  - Mengatur posisi monitor agar mata sama tingginya dengan tepi atas monitor sekitar 5-6 cm. Jika posisi monitor terlalu rendah, maka bisa mengakibatkan sakit pada leher dan pundak.

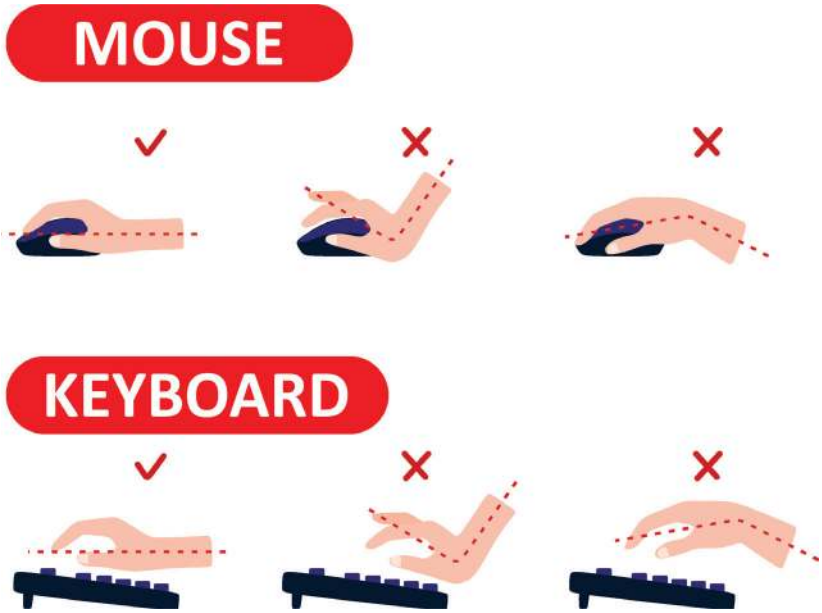
- Atur posisi monitor 45-60 cm jaraknya dengan kita, agar mata tidak terlalu lelah, tegang, sehingga berpotensi terjadinya gangguan pada mata.
- Jagalah kebersihan layar monitor agar tidak menimbulkan efek buram.

2) *Keyboard*

- Letakkan *keyboard* sesuai dengan arah layar monitor.
- Posisikan *keyboard* sesuai arah layar monitor. Pastikan posisi lengan sangat nyamandan rileks.
- Posisi pergelangan tangan kita harus lurus, jangan sampai menekuk ke atas atau ke bawah.
- Tekanlah tombol *keyboard* secara halus agar tangan dan jari kita tetap rileks. Maka gunakanlah *keyboard* yang berfungsi dengan baik.

3) *Mouse*

- Ukuran *mouse* yang sesuai dengan ukuran tangan akan membuat nyaman pada tangan kita.
- Posisi *mouse* dekat dengan *keyboard* agar dapat diraih dengan baik tanpa harus meregangkan tangan ke posisi yang lain dan jangan sampai merentangkan seluruh tangan, karena hal ini bisa menimbulkan otot tegang dan lelah.
- Memegang dan menggunakan *mouse* harus seringan mungkin dan melakukan kliknya dengan tegas. Saat menggerakkan *mouse*, posisi pergelangan tangan kita jangan bertumpu pada bagian depan meja.



Gambar 2.7 Cara menggunakan *mouse* dan *keyboar*d yang benar
Sumber: Marwondo, Rini Melati, & Dana R. N. Adnan/2023

b. Pengaturan Laboratorium Komputer

1) Pemilihan ruang komputer

- Pastikan ruang komputer tidak berada langsung di bawah lantai yang kegiatannya banyak menggunakan air.
- Pastikan juga ruang komputer tidak dekat dengan pusat medan listrik atau medan magnet.
- Ruang komputer juga harus dijauhkan dari ruangan yang melakukan kegiatan proses kimia.

2) Persyaratan teknis ruang komputer

- Temperatur dan kelembaban ruangan harus bisa terjaga dengan baik.
- Ruang komputer hendaknya terbebas dari debu, medan magnet, getaran, zat kimia, dan gas-gas tertentu.

Uji Kompetensi

Coba kalian jawab pertanyaan di bawah ini untuk mengetahui apakah kalian sudah menguasai teori tentang K3LH. Kalian bisa memilih opsi yang menurut kalian benar!

1. Berikut ini hal yang tidak termasuk dalam sasaran diterapkannya K3 di suatu usaha/industri/laboratorium adalah ....
   a. Menjamin keselamatan operator dan orang lain.
   b. Menjamin penggunaan peralatan aman dioperasikan.
   c. Menjamin proses produksi aman dan lancar.
   d. Menjamin terlaksananya perintah UU K3.
   e. Menjamin terlaksananya peraturan kesehatan.
2. Perilaku yang disengaja untuk membudayakan hidup bersih untuk mencegah manusia bersentuhan langsung dengan kotoran dan bahan buangan berbahaya lainnya, dengan harapan dapat menjaga dan meningkatkan kesehatan manusia, disebut ....
   a. *Hazard*
   b. Hygiene
   c. Sanitasi
   d. Safety
   e. He*althy*
3. Berikut ini adalah prinsip-prinsip dasar dalam menangani suatu keadaan darurat, kecuali ....
   a. Memeriksa pernafasan dan denyut jantung korban.
   b. Memastikan kita bukan menjadi korban berikutnya.
   c. Menggunakan metode atau cara pertolongan yang cepat, mudah, dan efesien.
   d. Mencatat usaha-usaha pertolongan yang telah kita lakukan
   e. Memastikan ada yang bisa menolong
4. Berikut adalah hal-hal yang perlu diperhatikan dalam cara bekerja yang aman sehingga penampilan diri ketika kerja selalu baik, kecuali ....
   a. Menggunakan Alat Pelindung Diri (APD).
   b. Menerapkan Konsep 5R (Ringkas, Rapi, Resik, Rawat, dan Rajin) dalam bekerja terutama setelah selesai melakukan pekerjaan.

c. Meposisikan badan sewaktu bekerja sesuai prinsip ergonomis.
d. Langkah dan urutan kerja dibuat fleksibel serta tidak selalu mengikuti prosedur operasi baku (SOP).
e. Disiplin mematuhi dan menjalani SOP di tempat kerja.

5. Dengan menerapkan prinsip ..., kalian mengetahui jumlah barang fisik yang ada di tempat kerja sehingga menghindari adanya barang yang berlebihan atau bahkan tidak diperlukan memenuhi tempat kerja kalian.
   a. Rapi
   b. Ringkas
   c. Resik
   d. Rawat
   e. Rajin
6. Pada prinsip ini yang ditekankan adalah bagaimana cara anda meletakkan barang-barang dan mudah untuk mendapatkannya kembali saat diperlukan.
   a. Rapi
   b. Ringkas
   c. Resik
   d. Rawat
   e. Rajin
7. Bising, vibrasi, suhu lingkungan yang ekstrem, dan radiasi merupakan jenis bahaya ....
   a. Fisik
   b. Kimiawi
   c. Laboratorium
   d. Ergonomic
   e. Biologi
8. Gerakan berulang atau posisi yang menetap selama melakukan pekerjaan tersebut dapat menimbulkan keluhan pegal linu, nyeri sendi, sakit pinggang, atau masalah lain yang lebih parah lagi. Ini merupakan bahaya kerja ....
   a. Fisik
   b. Kimiawi
   c. Laboratorium
   d. Ergonomic
   e. Biologi

9. Beberapa daftar kerugian yang tersembunyi dari kecelakaan kerja menurut Heinrich (1959) dalam ILO (1989:11) antara lain sebagai berikut, kecuali ....
   a. Kerugian akibat hilangnya waktu karyawan yang luka, karyawan lain yang ikut membantu (menolong), dan para pimpinan.
   b. Kerugian akibat rusaknya mesin, perkakas, dan/atau peralatan lainnya.
   c. Kerusakan akibat tercemarnya bahan-bahan baku.
   d. Kerugian insidental akibat terganggunya produksi, kegagalan memenuhi pesanan pada waktunya, kehilangan bonus, pembayaran denda ataupun akibat-akibat lain yang serupa.
   e. Keluhan, kesedihan, dan cacat.
10. Keadaan darurat yang berpotensi mengancam nyawa manusia dan hilangnya aset akibat kecelakaan kerja. Kecelakaan pada kategori ini merupakan kecelakaan skala kecil yang ditimbulkan oleh satu sumber atau kerusakan korban dan benda hanya terbatas. Keadaan darurat ini termasuk kategori ....
   a. I
   b. II
   c. III
   d. IV
   e. V

Pengayaan

Jika kalian tertarik dengan materi ini dan ingin mendalaminya lebih jauh, coba kalian cari tahu seperti apa penerapan K3LH dalam bidang lain melalui video berikut ini:

https://www.youtube.com/watch?v=7epjbQAc-eQ.

https://www.youtube.com/watch?v=6ZhBPy_5el0.

Refleksi

Kalian bisa memberikan tanda (√) pada kotak yang kalian anggap sesuai! Setelah kalian mempelajari bab ini, bagaimanakah penguasaan kalian terhadap materi-materi berikut?

No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Praktik-praktik kerja yang aman			
2.	Bahaya-bahaya di tempat kerja			
3.	Prosedur- prosedur dalam keadaan darurat			
4.	Penerapan budaya kerja industri 5R			
5.	Pencegahan kecelakaan kerja dan prosedur kerja			

1. Berdasarkan materi-materi yang sudah kalian pelajari di atas, manakah bagian yang kalian sangat sukai? Jelaskan alasannya!
2. Manfaat apa saja yang kalian peroleh untuk kehidupan sehari-hari, setelah kalian mempelajari semua materi di atas?
3. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim
untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

BAB 3

Proses Bisnis Pengembangan Perangkat Lunak dan Gim

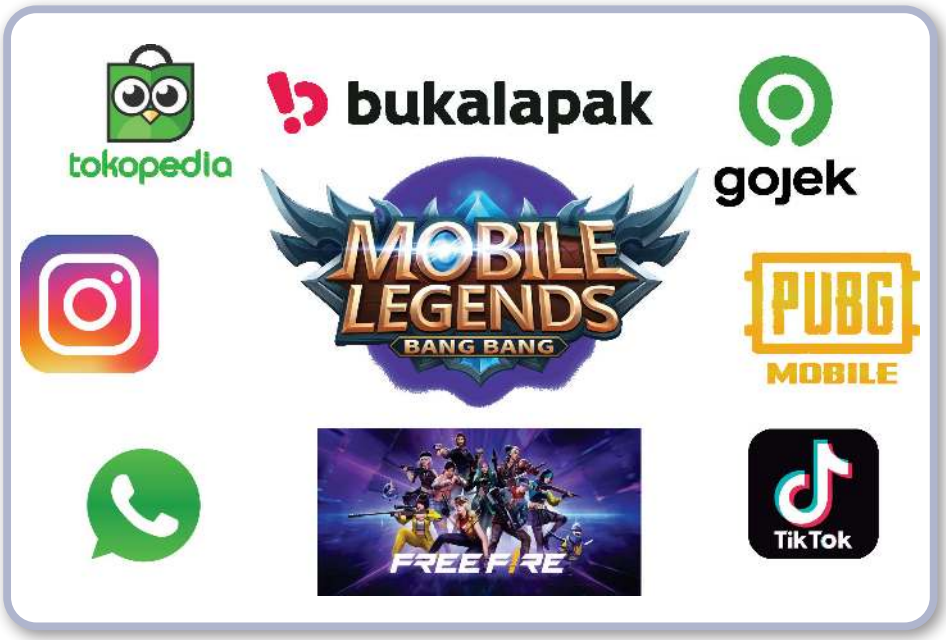
Tujuan Pembelajaran

Pada akhir pembelajaran materi ini, kalian diharapkan mampu mendeskripsikan proses pengembangan, pemasaran, penjaminan mutu, serta penerapan manajemen proyek dalam pembuatan perangkat lunak dan gim. Kalian juga diharapkan mampu mendeskripsikan kebutuhan-kebutuhan perangkat lunak sesuai dengan yang diharapkan oleh pelanggan.

Peta Materi

Kata Kunci

♦ Proses ♦ Rekayasa ♦ Perangkat Lunak ♦ Pengembangan ♦ Komunikasi
♦ Perencanaan ♦ Pemasaran ♦ Distribusi ♦ Mutu ♦ Kebutuhan
♦ Pengguna ♦ Proyek



Gambar 3.1 Perangkat lunak dan gim populer.
Sumber: Marwondo, Rini Melati, dan Dana R. N. Adnan/2023

Logo-logo tersebut pasti tidak asing dengan kalian, tetapi tahukah kalian bagaimana aplikasi dan permainan itu dikembangkan?

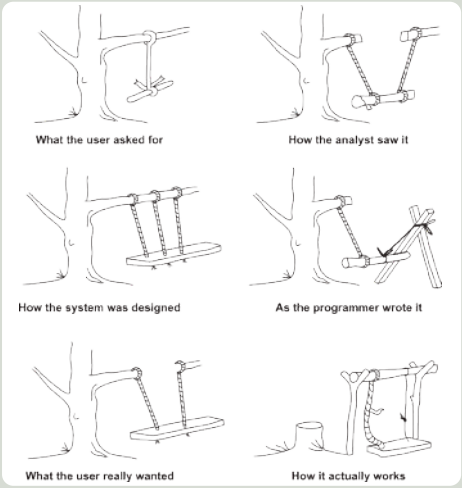
Apersepsi

Kalian pernah membuat prakarya, kan? Nah, saat membuat prakarya apa yang kalian lakukan? Tentu ada beberapa hal yang harus dilakukan sebelum (persiapan), pada saat (pelaksanaan), dan sesudah karya tersebut dihasilkan.

Dalam mengembangkan perangkat lunak pun mengalami langkah-langkah seperti itu. Ada kegiatan pra pengembangan, saat pengembangan, serta pasca pengembangan. Bagaimana prosesnya, kita akan pelajari dalam bab ini.

A. Proses Pengembangan dan Pemasaran Perangkat Lunak dan Gim

Aktivitas Belajar 3-1



Gambar 3.2 *Software Engineering Perspective*
Sumber: medium.com/David Martin/2019

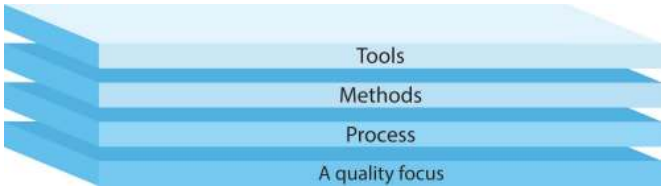
Perhatikan gambar 3.2 di atas, diskusikan dengan teman sekelompokmu. Berdasarkan hasil diskusi kalian gambar di atas bercerita tentang apa?

Salah satu karakteristik dari perangkat lunak adalah **direkayasa** bukan dimanufaktur. Untuk itu, mengembangkan perangkat lunak berarti merekayasa perangkat lunak. Rekayasa adalah istilah yang digunakan untuk engineering. Mengapa harus direkayasa?

Seperti yang telah kalian ketahui sebelumnya, ada berbagai jenis perangkat lunak yang mendukung kebutuhan manusia mulai dari yang sederhana sampai dengan yang sangat kompleks. Permasalahan yang berbeda-beda pada perangkat lunak membutuhkan pendekatan yang berbeda pula dalam memecahkan masalahnya. Mengembangkan perangkat lunak aplikasi tidak akan sama dengan sistem tertanam, begitu juga untuk perangkat lunak sains dan juga permainan (gim). Mengembangkan perangkat lunak yang berbeda membutuhkan metode dan teknik yang berbeda.

Merekayasa berarti menerapkan ilmu pengetahuan untuk menjadikan kehidupan manusia lebih baik. Merekayasa perangkat lunak berarti menerapkan rekayasa pada perangkat lunak. Merekayasa perangkat lunak menurut IEEE (*Institute of Electrical and Electronics Engineers*) berarti bagaimana menerapkan pendekatan yang sistematis, terdisiplin, terukur untuk pengembangan,

pengoperasian, dan pemeliharaan perangkat lunak (Presman & Maxim, 2020). Rekayasa perangkat lunak tidak hanya menekankan pada proses pembangunan, tetapi seluruh aspek produksi perangkat lunak (Sommerville, 2016).



Gambar 3.3 *Software Engineering Layer*
Sumber: Pressman & Maxim/2020

Rekayasa perangkat lunak merupakan lapisan-lapisan yang bertumpu pada kualitas. Komitmen akan kualitas menjadi dasar untuk melakukan proses, memilih metode yang tepat, serta menentukan alat bantu yang sesuai dengan permasalahan sebagaimana terlihat pada gambar 3.3.

1. Proses Perangkat Lunak

Proses pengembangan perangkat lunak merupakan Peta Jalan (*roadmap*) untuk membangun produk perangkat lunak berkualitas tinggi. Proses tersebut berupa sekumpulan aktivitas yang tujuannya adalah pengembangan atau evolusi perangkat lunak. Proses diadaptasi untuk memenuhi kebutuhan perekayasa dan manajer perangkat lunak. Berbagai jenis proyek memerlukan proses perangkat lunak yang berbeda. Produk kerja dihasilkan oleh proses perangkat lunak. Indikator terbaik dari seberapa baik proses perangkat lunak telah bekerja adalah kualitas, ketepatan waktu, dan kelangsungan hidup jangka panjang dari produk perangkat lunak yang dihasilkan.

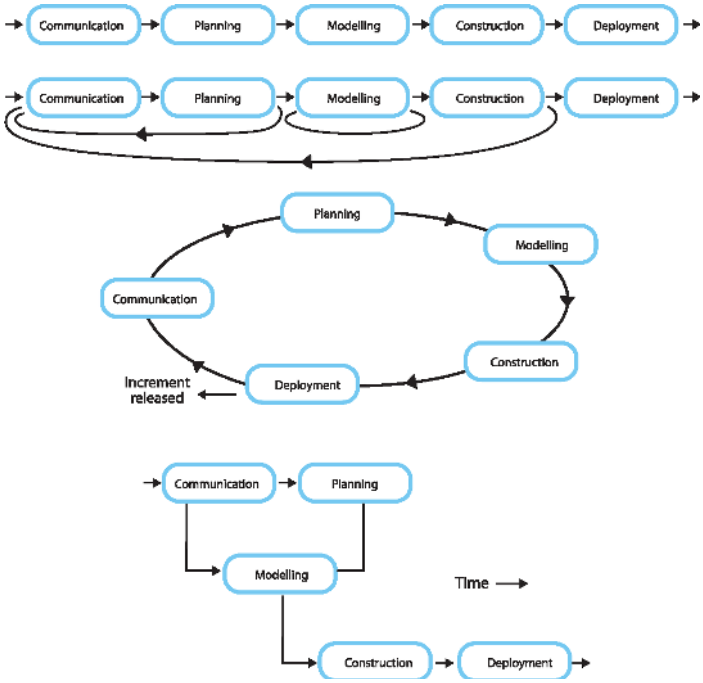
Proses perangkat lunak (*software process*) merupakan suatu peta jalan (*roadmap*) untuk menghasilkan suatu produk perangkat lunak yang berkualitas. Proses perangkat lunak berisi serangkaian aktivitas yang bertujuan untuk membangun perangkat lunak maupun pengembangannya, dimulai dari aktivitas sebelum pengembangan, saat pengembangan, maupun pasca pengembangan. Tentu tidak semua permasalahan dapat dipecahkan dengan proses yang sama. Setiap permasalahan memerlukan proses yang unik dan berbeda satu dengan yang lainnya. Meskipun berbeda, tetapi ada serangkaian aktivitas yang menjadi dasar untuk kemudian disesuaikan oleh para perekayasa perangkat lunak.

Secara umum kerangka kerja pengembangan perangkat lunak terbagi ke dalam lima hal, yaitu (Presman & Maxim, 2020):

a. Komunikasi (*Communication*)
b. Perencanaan (*Planning*)
c. Pemodelan (*Modeling*)
d. Konstruksi (*Construction*)
e. Penyebaran (*Deployment*)

Kelima kerangka kerja umum tersebut dapat digunakan untuk mengembangkan perangkat lunak baik berskala kecil maupun besar, dari pembuatan perangkat lunak berbasis teks sampai berbasis web, dari yang sederhana sampai dengan yang kompleks. Meskipun secara umum sama, tetapi detail proses pengembangan perangkat lunak dapat berbeda sesuai dengan permasalahan yang dihadapi.

Dalam penerapannya, kelima kerangka kerja umum dapat dilakukan secara berurut (sekuen) maupun iteratif. Bisa juga beberapa tahapan dikerjakan beruntun sedangkan tahapan lainnya dikerjakan secara iteratif. Iterasi dibutuhkan manakala pengembang dan pelanggan belum menemukan kesesuaian terhadap apa yang sudah dicapai. Berikut contoh beberapa penerapan kerangka kerja umum.



Gambar 3.4 Ragam Proses
Sumber: Pressman & Maxim/2020

a. Komunikasi

Sebelum memulai melaksanakan pengembangan perangkat lunak, sangat perlu untuk berkomunikasi dengan semua pemangku kepentingan. Komunikasi ini dilakukan untuk memahami permasalahan apa yang ada pada semua pemangku kepentingan. Selain itu juga komunikasi diperlukan agar tidak ada kesalahpahaman dalam pelaksanaan pekerjaan. Masih ingat gambar 3.2? Komunikasi akan dapat menghindari kesalahan penafsiran dari masing-masing pemangku kepentingan.

Tim pengembang perangkat lunak membutuhkan lebih banyak informasi sebelum mulai menjalankan tugasnya masing-masing. Tim pengembang juga memerlukan patokan atau peta jalan agar dalam pengerjaan tidak mengalami kesalahan penafsiran. Oleh karena itu, kalian sebagai perekayasa dihadapkan pada pertanyaan kunci yang harus bisa dijawab: "Tindakan apa yang sesuai untuk perangkat lunak yang akan dikembangkan?"

Untuk menjawab pertanyaan tersebut, maka kalian harus berkomunikasi dengan semua pemangku kepentingan. Komunikasi dimaksudkan untuk mendapatkan informasi secara signifikan agar tidak ada hal-hal yang bisa menyulitkan tim pada saat pengembangan perangkat lunak dilakukan. Kalian harus dapat berkomunikasi dengan seluruh pemangku kepentingan. Beberapa tugas yang dapat dilakukan dalam komunikasi jika pekerjaan perangkat lunak sederhana antara lain:

1) Melakukan kontak dengan pemangku kepentingan melalui sarana komunikasi yang ada maupun bertatap muka langsung.
2) Diskusikan kebutuhan dan kembangkan catatan untuk mendapatkan lebih detail.
3) Atur catatan menjadi pernyataan tertulis singkat tentang kebutuhan.
4) Kirimkan kembali ke pemangku kepentingan apa yang sudah ditulis dalam catatan untuk ditinjau dan disetujui.

Tugas yang harus kalian lakukan tentu berbeda jika pekerjaan pembangunan perangkat lunak atau gim jauh

lebih kompleks dengan jumlah pemangku kepentingan lebih banyak, masing-masing dengan serangkaian kebutuhan berbeda (bahkan kadang-kadang saling bertentangan), aktivitas komunikasi mungkin dapat dilakukan dalam enam tindakan berbeda: **insepsi (*inception*)**, **elisitasi (*elicitation*)**, **elaborasi (*elaboration*)**, **negosiasi (*negotiation*)**, **spesifikasi (*specification*)**, dan **validasi (*validation*)** yang akan dijelaskan pada bagian berikutnya.

Masing-masing tindakan rekayasa perangkat lunak ini mungkin memiliki banyak tugas kerja dan dalam beberapa kasus sejumlah produk kerja yang berbeda. Masalah yang telah ditemukan harus dinyatakan secara eksplisit (tegas dan jelas) termasuk ruang lingkup atau batasan, sasaran serta tujuannya. Pernyataan masalah harus memuat sudut pandang pelanggan serta pengembang, tidak boleh hanya salah satu. Gunakan kalimat yang bisa dipahami oleh kedua belah pihak. Contoh permasalahan dalam sudut pandang pelanggan misalkan: kehadiran, persediaan, penggajian, potongan, pengendalian, dan lain-lain. Contoh masalah dari sudut pandang pengembang misalkan: struktur data, relasi data, algoritma, struktur kelas, teknik pemrograman, dan lain-lain. Jangan ajak pelanggan untuk memikirkan masalah teknis pengembang.

b. Perencanaan

Apa yang kalian lakukan jika akan melakukan perjalanan? Ya, kalian membuat rencana perjalanan terlebih dahulu. Ke mana tujuannya, singgah di mana saja, menggunakan kendaraan apa, dengan siapa saja, serta berapa biaya yang harus dikeluarkan. Begitu pun dalam pengembangan perangkat lunak. Kalian harus punya perencanaan yang digunakan sebagai peta agar tidak tersesat.

Berdasarkan hasil komunikasi yang dilakukan sebelumnya, tentunya kalian sudah memahami permasalahan yang harus diselesaikan, kan? Kalian juga sudah menemukan apa saja kebutuhan yang diinginkan oleh pemangku kepentingan, kan? Nah, sekarang waktunya kalian mendefinisikannya dalam sebuah perencanaan.

Nyatakan hasil identifikasi permasalahan dalam sebuah pernyataan yang menggambarkan masalah serta ruang lingkupnya. Berdasarkan permasalahan yang sudah dinyatakan tersebut, identifikasilah kemungkinan-kemungkinan solusi yang ada. Identifikasi juga sumberdaya-sumberdaya yang dimiliki untuk menyelesaikan permasalahan tersebut, misalkan jumlah *programmer*, *database administrator*, perangkat pengembangan, perangkat implementasi, dan sistem lain yang sudah ada sebelumnya.

Tentukan juga pendekatan serta model pengembangan apa yang akan digunakan pada pembangunan perangkat lunak ini nantinya. Tentukan alat bantu yang digunakan selama pengembangan, pertimbangkan skala perangkat lunak, tingkat kesulitan, waktu pengerjaan serta biaya. Uraikan juga rencana manajemen pelaksanaan, pengendalian, dan penjaminan kualitas.

Perencanaan pengembangan perangkat lunak lebih detail dituliskan dalam dokumen *Sotware Development Plan* (SDP) atau Rencana Pengembangan Perangkat Lunak (RPPL). RPPL berisi hal-hal seperti ruang lingkup, asumsi-asumsi, apa saja yang harus dihasilkan dari perangkat lunak ini, organisasi pengembangan perangkat lunak, proses manajemen, dan rencana proses teknis. Dalam perencanaan tersebut juga diidentifikasi risiko-risiko yang kemungkinan terjadi selama proses pengembangan.

c. Pemodelan

Jika kalian akan membuat baju, apakah kalian akan langsung menjahitnya? Tentu tidak. Ada sketsa atau coret-coretan yang kalian buat dulu agar tukang jahit paham yang kalian maksud, paling tidak kalian menunjukkan contoh bajunya seperti apa. Atau jika kalian datang ke tukang pangkas rambut, kalian bisa lihat contoh-contoh potongan rambut yang dipasang sebelum tukang cukur mencukur rambut kalian? Setelah kalian memilih sketsa ataupun contoh, kemudian mendetailkan seperti apa yang diinginkan. Proses membuat sketsa atau menggunakan contoh adalah salah satu bentuk dari memodelkan.

Memodelkan berarti merepresentasikan permasalahan dalam bentuk yang lebih sederhana. Memodelkan berarti menggunakan aturan- aturan atau notasi tertentu agar permasalahan dalam skala yang besar dapat dibuat menjadi lebih sederhana dan mudah dipahami.

Dalam pengembangan perangkat lunak juga kalian memerlukan pemodelan. Model proses pengembangan perangkat lunak, biasa juga disebut dengan siklus hidup pengembangan perangkat lunak atau *System Development Life Cycle* (SDLC) merupakan representasi dari proses perangkat lunak yang dibuat menjadi lebih sederhana (Sommerville, 2016). Setiap model proses biasanya akan mewakili suatu perspektif tertentu dalam pengembangan perangkat lunak. Artinya, berbeda model proses akan mewakili suatu cara penyelesaian yang berbeda. Beberapa perspektif pemodelan yang ada pada model proses perangkat lunak antara lain:

- Alur kerja - urutan aktivitas,
- Aliran data - aliran data dan informasi,
- Peran/tindakan - siapa akan melakukan apa.

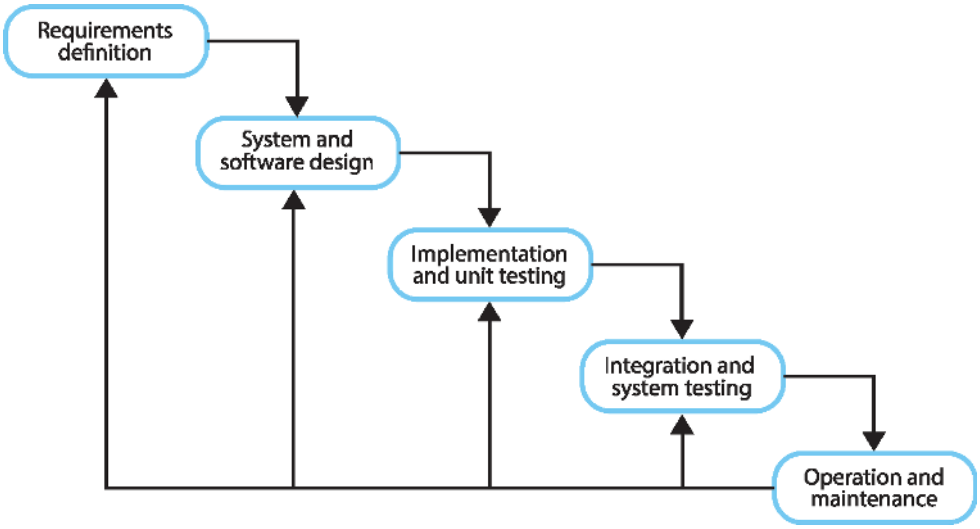
Perspektif alur kerja, misalkan, menunjukkan aktivitas dan urutan pekerjaannya, tetapi mungkin saja di dalamnya tidak menunjukkan peran atau tindakan apa yang dilakukan. Demikian juga pada perspektif lainnya.

Meskipun ada banyak pemodelan yang bisa digunakan dalam mengembangkan perangkat lunak, kalian dapat menggunakan pemodelan yang bersifat generik. Model generik ini adalah deskripsi abstrak tingkat tinggi dari proses perangkat lunak yang dapat digunakan untuk menjelaskan pendekatan yang berbeda untuk pengembangan perangkat lunak. Kalian dapat menganggapnya sebagai kerangka kerja proses yang dapat diperluas dan disesuaikan untuk membuat proses rekayasa perangkat lunak yang lebih spesifik.

Meskipun ada banyak pemodelan yang bisa digunakan dalam mengembangkan perangkat lunak, kalian dapat menggunakan pemodelan yang bersifat generik. Model generik merupakan deskripsi abstrak yang menjadi kerangka kerja dasar yang dapat diperluas sesuai kebutuhan pengembangan

perangkat lunak yang spesifik. Model proses generik yang umum digunakan adalah:

1) **Model Air Terjun (*waterfall*)**, merupakan model proses dasar yang memiliki aktivitas terdiri dari spesifikasi (*specification*), pengembangan (*development*), validasi (*validation*), dan evolusi (*evolution*). Ciri utama dari model ini adalah fase yang terpisah- pisah dan berurutan dalam pengerjaannya. Model ini juga biasa disebut sekuensial linier.



Gambar 3.5 Contoh model air terjun
Sumber: Ian Sommerville/2016

Model air terjun ini pertama kali diperkenalkan oleh Winston Royce pada tahun 1970-an dan merupakan model tertua pada pengembangan perangkat lunak. Seiring dengan perjalanannya, model ini juga masih banyak digunakan untuk pengembangan perangkat lunak dengan kualifikasi:

- Proyek nyata yang alur kerjanya berurutan sesuai model.
- Pelanggan yang mau atau bisa menyatakan semua kebutuhan secara eksplisit di awal.
- Pelanggan yang mau bersabar karena hasil kerja baru akan terlihat di akhir proyek.

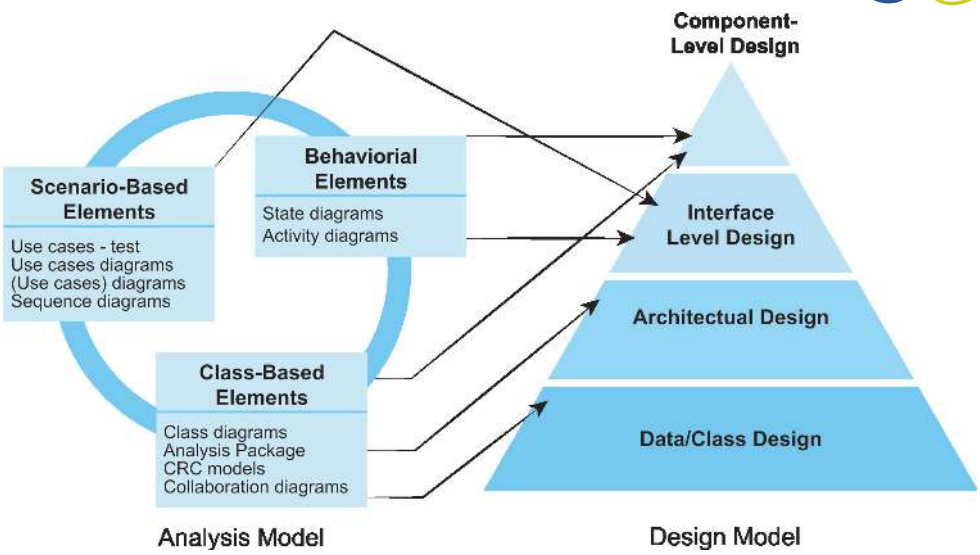
- Kemungkinan kesalahan besar sudah terdeteksi dan teridentifikasi di awal sampai program kerja ditinjau.

Tahapan yang ada pada model ini mencerminkan kegiatan pengembangan perangkat lunak mendasar, yaitu:

- **Analisis Kebutuhan** (*Requirement Analysis*). Kegiatan ini dilakukan untuk mendefinisikan kebutuhan perangkat lunak yang berasal dari pelanggan maupun sistem. Kebutuhan mencakup informasi, layanan, data, antarmuka. Hasil analisis digunakan sebagai spesifikasi dari perangkat lunak. Kalian dapat menggunakan satu atau lebih jenis model berikut:
  a) pemodelan berbasis skenario,
  b) pemodelan berorientasi kelas,
  c) pemodelan perilaku,
  d) pemodelan data,
  e) pemodelan berorientasi aliran.

Model-model tersebut dapat memberikan informasi kepada perekayasa perangkat lunak yang kemudian diterjemahkan ke dalam desain. Model ini akan menjadi sarana juga bagi pengembang untuk melakukan penilaian kualitas (pengujian) setelah perangkat lunak selesai dibangun.

- **Desain Sistem dan Perangkat Lunak** (*System and Software Design*). Kegiatan ini dilakukan untuk menerjemahkan kebutuhan yang telah dimodelkan pada kegiatan sebelumnya menjadi desain perangkat lunak menyeluruh. Desain yang harus didefinisikan meliputi desain data, desain arsitektur, desain antarmuka, desain komponen, serta desain lainnya dari keseluruhan perangkat lunak. Proses penerjemahan dari model analisis ke dalam model desain seperti pada gambar 3.6 berikut.

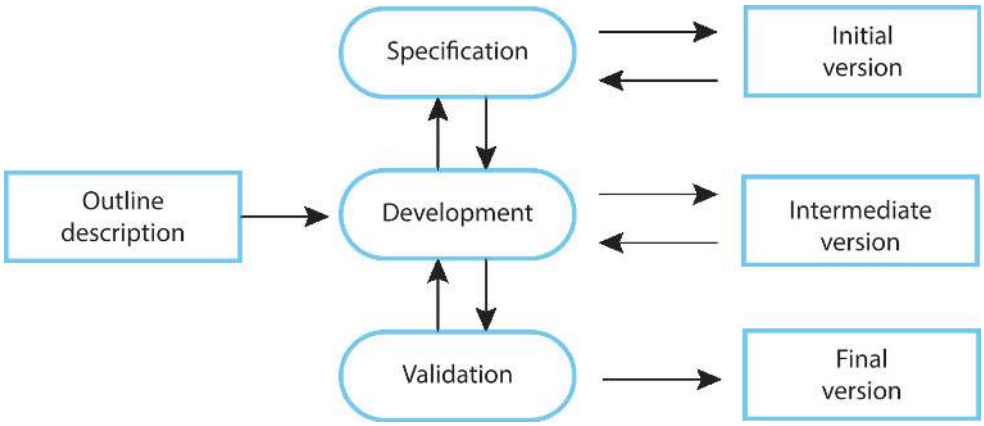


Gambar 3.6 Penerjemahan model analisis ke dalam model desain
Sumber: Pressman & Maxim/2020

- **Implementasi dan Pengujian Unit** (*Implementation and Unit Testing*). Desain perangkat lunak telah dibuat, maka harus direalisasikan. Kegiatan inilah yang merealisasikan setiap elemen desain ke dalam unit program. Setelah direalisasikan, pada unit program dilakukan pengujian untuk memastikan tidak ada kesalahan yang terjadi, terutama kesalahan logika program.
- **Integrasi dan Pengujian Sistem** (*Integration and System Testing*). Unit-unit program yang telah dibuat disatukan menjadi sebuah sistem secara lengkap. Meskipun pada unit program telah dilakukan pengujian, tetapi harus dipastikan ketika sudah diintegrasikan apakah masih terdapat kesalahan. Setelah dilakukan pengujian secara keseluruhan, perangkat lunak diserahkan ke pelanggan untuk digunakan.
- **Operasi dan Pemeliharaan** (*Operation and Maintenance*). Perangkat lunak yang telah diserahkan ke pelanggan akan dioperasikan. Selama proses pengoperasian terhadap perangkat lunak, perlu dilakukan pemeliharaan. Pemeliharaan

diperlukan sebagai bentuk koreksi atas kesalahan yang tidak ditemukan pada saat pengujian atau kebutuhan peningkatan layanan sistem yang sebelumnya tidak tercatat pada awal proses pengembangan.

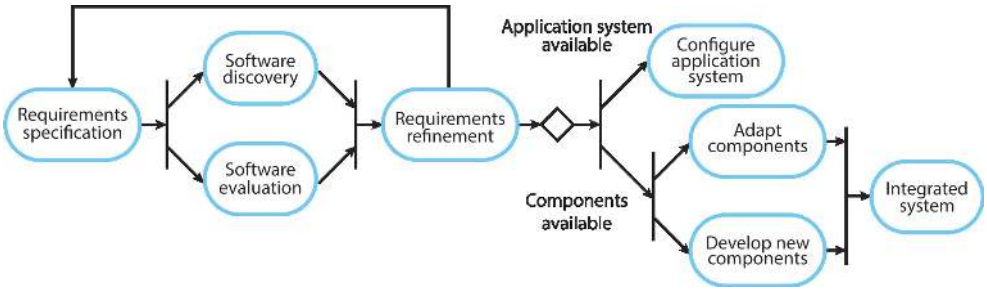
2) **Model Inkremental (*Incremental*)**, didasarkan pada keadaan bahwa pelanggan tidak dapat dengan pasti mendeskripsikan kebutuhannya serta ketidaksabaran pelanggan menunggu hasil akhir perangkat lunak yang dikembangkan. Pengembang membuat penafsiran awal sesuai deskripsi keluaran dalam bentuk purwa rupa yang diperlihatkan kepada pelanggan untuk mendapatkan masukan dari pengguna dan orang lain serta dapat mengembangkan dalam versi yang berbeda-beda. Kegiatan ini dilakukan berulang sampai dengan adanya kesepakatan pelanggan terhadap layanan yang diberikan oleh perangkat lunak. Model incremental ini merupakan cikal bakal lahirnya metode *agile development*. Model pengembangan inkremental ini digambarkan pada gambar 3.7.



Gambar 3.7 Model proses inkremental (*incremental*)
Sumber: Ian Sommerville/2016

3) **Model Integrasi dan Konfigurasi**. Sebagian besar pengembangan perangkat lunak saat ini dilakukan dengan cara menggunakan kembali (*reuse*) perangkat lunak sejenis yang telah dibangun sebelumnya. Ini dilakukan mengingat tuntutan waktu pengerjaan yang lebih singkat serta keengganan pelanggan

untuk mendeskripsikan kebutuhannya secara detail. Pengembang yang mengerjakan mencari kode yang mirip dengan yang dibutuhkan untuk kemudian dimodifikasi sesuai kebutuhan serta mengintegrasikannya ke dalam kode yang baru.



Gambar 3.8 Model proses integrasi dan konfigurasi
Sumber: Ian Sommerville/2016

Model seperti ini sudah mulai digunakan sejak tahun 2000. Model pengembangan ini berfokus pada penggunaan kembali komponen perangkat lunak yang ada dan banyak digunakan. Komponen yang ada pada perangkat lunak digunakan kembali pada perangkat lunak yang baru dengan beberapa penyesuaian. Komponen-komponen yang sering digunakan kembali antara lain (Sommerville, 2016):

1) Sistem *standalone* yang dikonfigurasi ulang untuk digunakan pada lingkungan yang berbeda. Biasanya sistem bersifat umum atau pada awalnya dikembangkan secara generik yang memiliki banyak fitur. Fitur-fitur yang ada disesuaikan kembali dengan lingkungan yang baru.
2) Sekumpulan objek yang digunakan sebagai library untuk komponen-komponen umum. Kumpulan tersebut biasanya diintegrasikan menjadi kerangka kerja (*framework*) tertentu. Pengembang tinggal menggunakan dan mengkonfigurasi komponen yang tersedia sesuai dengan kebutuhan.
3) Layanan web yang dengan sengaja dikembangkan untuk memberikan layanan standar perangkat lunak jenis tertentu dan dapat dikerjakan jarak jauh.

d. Konstruksi

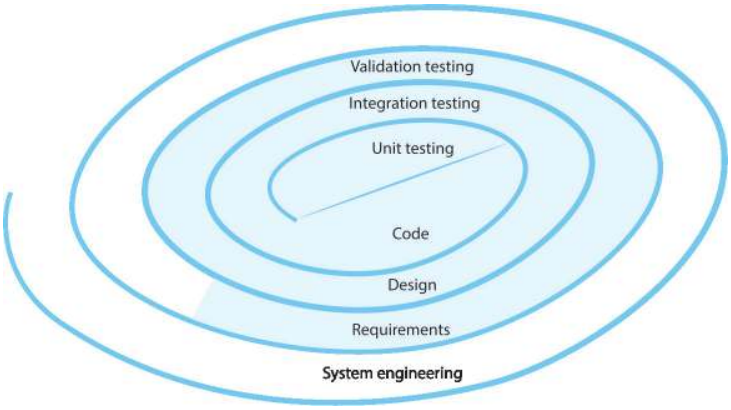
Jika kalian sudah memiliki sketsa baju, apa yang kemudian dilakukan? Ya, tentu membawa sketsa tersebut ke tukang jahit agar baju bisa dibuat. Tahapan proses berikutnya pada pengembangan perangkat lunak adalah konstruksi. Proses ini menerjemahkan desain yang sudah kalian buat baik itu desain tampilan, desain data, maupun desain algoritma menjadi kode program. Kalian harus menentukan bahasa pemrograman, DBMS, maupun alat bantu penerjemah lainnya yang tepat untuk mewujudkan desain yang kalian buat. Jika kalian menggunakan alat bantu berupa *Computer Aided Software Engineering* (CASE), maka koding dapat di- *generate* secara otomatis, jika tidak maka harus dilakukan secara manual.

Tugas ini dilakukan oleh pemrogram maupun *database implementor*. Seorang pemrogram harus mampu membaca dengan baik desain yang telah dibuat oleh *software architect*. Meskipun koding dapat langsung dibuat otomatis, tetapi tetap saja ada bagian-bagian tertentu yang harus disesuaikan kembali dengan kebutuhan rancangan.

Agar hasil penerapan sesuai dengan kebutuhan, maka perangkat lunak harus menjalani pengujian. Pengujian dilakukan oleh pihak lain selain pemrogram. Pengujian dimaksudkan untuk memastikan bahwa perangkat lunak yang dibuat sudah tidak lagi memiliki kesalahan dan sudah sesuai dengan kebutuhan.

Pengujian perangkat lunak dibagi menjadi beberapa tingkatan, yaitu:

- pengujian unit (*unit testing*)
- pengujian integrasi (*integration testing*)
- pengujiaan validasi (*validation testing*)
- pengujian sistem (*system testing*)



Gambar 3.9 *Testing strategy*
Sumber: Pressman & Maxim/2020

e. Penyebaran/Distribusi

Rangkaian kegiatan yang tidak kalah penting dalam pengembangan perangkat lunak adalah bagaimana ia didistribusikan setelah dikembangkan. Distribusi perangkat lunak adalah proses mengirimkan hasil perangkat lunak kepada pengguna. Istilah distribusi yang popular adalah distro.

Distro merupakan sekumpulan komponen perangkat lunak yang dibangun, dirakit, serta dikonfigurasi sedemikian rupa agar dapat digunakan oleh pengguna. Sebuah distro dapat berbentuk distribusi biner yang dapat langsung di-*install*-kan pada perangkat pengguna atau harus menggunakan piranti tambahan agar dapat dieksekusi. Contoh peng-*install* Sistem Operasi, peng-*install* MsOffice, dan lain- lain.

Pada perkembangan teknologi perangkat lunak dan gim saat ini yang mengandalkan komputasi awan (*cloud computing*), distribusi perangkat lunak atau gim juga dapat dilakukan melalui teknologi ini. Perangkat lunak tidak perlu di-*install* ke dalam perangkat tetapi dapat langsung dijalankan.

Pada perangkat *mobile*, distribusi perangkat lunak atau gim dilakukan melalui pangkalan aplikasi (*store*) sesuai dengan sistem operasi yang digunakan, misalkan Playstore, Appstore, Microsoft Store, atau Chome Web Store.

Aktivitas Belajar 3-2

Buatlah sebuah dokumen sederhana yang menggambarkan rangkaian aktivitas kalian jika kalian akan mengembangkan sebuah perangkat lunak penjualan di minimarket. Gunakan *outline* berikut.

Nama Perangkat Lunak : <nama generik>

Alias : <nama komersil atau akronim>

Versi : <versi>

Tim Pengembang : 1. <Nama>
2.<Nama>
3.<Nama>

1. Gambaran Umum
   - 1.1 Deskripsi Umum Sistem

     Jelaskan secara singkat maksud, ruang lingkup, dan tujuan pembangunan perangkat lunak.
   - 1.2 Asumsi dan Batasan

     Tuliskan asumsi-asumsi yang ada, misalkan ketersediaan *server*, jaringan, *backup power*, dan lainnya.
   - 1.3 Hasil Proyek

     Tuliskan dan jelaskan apa saja yang harus dihasilkan serta jangka waktu penyelesaiannya, termasuk jika ada penyerahan bertahap sesuai dengan kontrak kerja.
2. Organisasi Projek
   - 2.1 Struktur Organisasi

     Jelaskan struktur organisasi proyek pengembangan perangkat lunak. Organisasi pengembangan perangkat lunak biasanya minimal terdiri dari: *Project Manager*, *System Analyst*, *Developer*, dan *Tester*. Buat dalam bentuk organogram serta isi nama-nama personil sesuai posisinya.
   - 2.2 Peran dan Tanggung Jawab

     Jelaskan peran dan tanggung jawab dari setiap elemen pada struktur organisasi yang ada. Dapat dibuat dalam bentuk tabel agar lebih ringkas.

3. Proses Manajemen

   3.1 Estimasi Pengerjaan

   Uraikan rentang waktu pengerjaan pengembangan perangkat lunak ini dilakukan. Kapan dimulai, kapan prakiraan akan diselesaikan, serta batas-batas waktu toleransi yang dimungkinkan.

   3.2 Rencana Pengerjaan

   Tuliskan dan uraikan fase-fase pengembangan perangkat lunak ini. Sesuaikan dengan metode dan pendekatan yang digunakan. Buat uraian sedetail mungkin untuk setiap tahapan.

   3.3 Jadwal Pengerjaan

   Buat jadwal pengerjaan, termasuk estimasi waktu yang diperlukan setiap tahapan serta capaian. Buat dalam satuan yang berbeda (bulan, minggu, dan hari) agar lebih jelas.

4. Rencana Proses Teknis

   4.1 Metode, Alat, dan Teknik

   Tuliskan dan jelaskan semua metode, alat, dan teknik yang digunakan dalam pengembangan perangkat lunak ini, termasuk versi yang digunakan.

   4.2 Rencana Infrastruktur

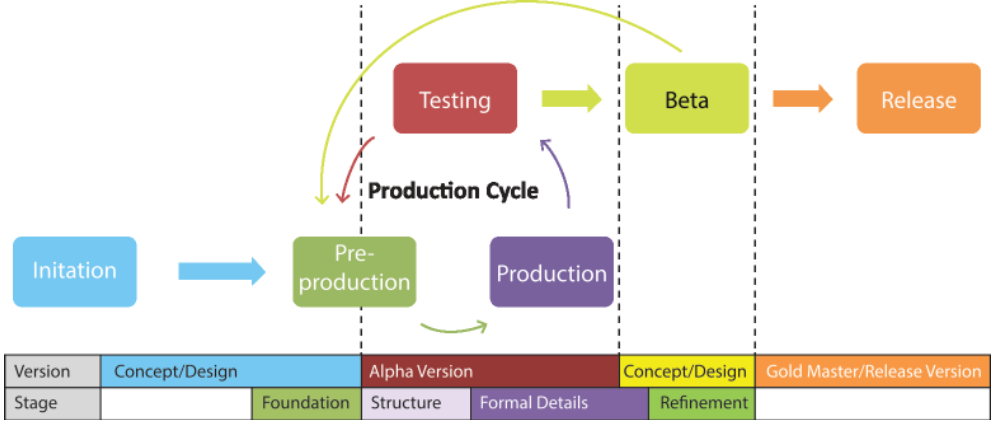
   Tuliskan rencana infrastruktur penunjang yang diperlukan untuk pengembangan perangkat lunak ini. Jika tidak memerlukan infrastruktur cukup tuliskan "Tidak memerlukan infrastruktur khusus".

2. Proses Pengembangan Gim

Meskipun sebelumnya sudah dibahas bagaimana mengembangkan Perangkat Lunak secara umum, kalian juga harus tahu bahwa setiap jenis perangkat lunak memerlukan pendekatan yang berbeda dalam mengembangkannya, demikian halnya dengan gim. Kalian dapat mengembangkan gim menggunakan serangkaian aktivitas yang dinamakan *Game Development Life Cycle* (GDLC) atau Siklus Hidup Pengembangan Gim. Apa itu GDLC dan bagaimana aktivitasnya, yuk simak bahasan berikut.

Sebenarnya mengembangkan gim tidak jauh berbeda dengan mengembangkan perangkat lunak dikarenakan gim masih termasuk salah satu jenis perangkat lunak. Meskipun gim itu sebuah "permainan"

yang digunakan untuk main-main, tetapi kompleksitanya tinggi, lho. Diperlukan beberapa penambahan dan penekanan aktivitas dalam mengembangkan gim karena kompleksitasnya. GDLC digunakan sebagai panduan tahapan pekerjaan bagi para pengembang dalam membangun sebuah gim.



Gambar 3.10 *Game development phase*
Sumber: Ramadan & Widyani/2013

Gambar 3.10 menunjukkan fase-fase pada pengembangan gim. Diagram pada bagian atas menunjukkan fase, sedangkan tabel pada bagian bawah menunjukkan perkembangan pembangunan gim.

a. Inisiasi

Memulai langkah memang bukan sesuatu yang mudah. Langkah awal pengembangan gim dimulai dari inisiasi. Pada inisiasi, yang harus dilakukan adalah memunculkan ide. Tentu ide yang dimunculkan bukan ide sembarangan. Ide akan sulit kita munculkan jika kalian kurang memiliki dasar pengetahuan dan pengalaman. *Brainstorming* adalah kuncinya, dan berikut adalah beberapa petunjuk yang berguna:

- Jenis permainan apa yang akan kalian buat?
- Bagaimana *gameplay*-nya?
- Siapa saja karakternya?
- Fitur apa yang dimilikinya?
- Apakah 2D atau 3D?
- Bagaimana cerita dimulai?
- Siapa audiens target kalian?

- Apa *platform* targetnya?
- Teknologi/mesin apa yang akan digunakan?

b. Pra-Produksi

Tahap kedua ini merupakan tahapan yang kritis. Setelah ide dimunculkan, tidak mungkin dibiarkan liar begitu saja, harus mulai direalisasikan. Pra produksi merupakan "*game design*" awal yang nanti akan disempurnakan pada tahap-tahap selanjutnya. Pada tahap ini mulai didefinisikan konsep gim, genre, karakter, fitur *gameplay*, mekanik gim, interaksi pemain, faktor kesenangan gim, dan teknik monetasi. Setelah desain gim didefinisikan, purwa-rupa pertama mulai dibuat. Purwa-rupa pertama ini digunakan untuk menunjukkan *gameplay* apa yang dimiliki. Setelah fondasi diterima, dimulai dengan menyempurnakan fondasi menjadi struktur, purwa-rupa yang disempurnakan dan lebih dapat dimainkan yang menunjukkan potensi menyenangkan dan fungsional.

c. Produksi

Proses inti dari siklus produksi adalah produksi. Ini adalah jantung dari proses pengembangan gim. Produksi terkait erat dengan pembuatan aset gim, kode sumber, dan integrasi kedua elemen. Penjelasannya memang sesederhana itu, tetapi realita kegiatannya tidak sesederhana itu. Kalian harus mengelola *milestone*, *deliverable*, dan memastikan bahwa ketika kedua elemen terintegrasi, itu akan bekerja dengan baik. Hasil produksi adalah gim yang dapat dimainkan. Dalam bentuk apa? Dalam bentuk *Formal Details Prototype* atau *Refinement Prototype*.

Formal Details adalah gim yang dapat dimainkan yang tidak hanya mencakup *gameplay* dan mekanik, tetapi juga aturan menang-kalah, hubungan antar fitur, dan yang paling penting: berjalan dengan baik. Itu harus dapat dimainkan setidaknya untuk menguji permainan apakah itu fungsional, lengkap, dan seimbang.

Refinement adalah purwa-rupa paling matang yang hanya membutuhkan beberapa pemolesan. Ini adalah penyempurnaan dari detail formal, lengkap, dapat dimainkan, dan hampir siap untuk dikirim. Tetapi, itu harus cukup intuitif untuk dimainkan oleh pemain.

d. Pengujian

Sesuai dengan namanya, pengujian berkisar pada evaluasi *build game*. Pengujian dilakukan untuk menguji fitur gim, nilai, konsep, desain, semuanya. Semuanya. Pengujian dilakukan oleh anggota tim internal untuk menilai fitur gim, modul, dan lain-lain. Menjelang akhir, tes digunakan untuk menilai gim secara keseluruhan. Pengujian membutuhkan rencana pengujian, skenario, dan beberapa diskusi antara sesama pengembang. Beberapa petunjuk berguna:

- Apakah gim masih *buggy*?
- Pernahkah kalian terjebak dan tidak bisa melanjutkan permainan?
- Apakah ada tanda-tanda eksploitasi/kesalahan?
- Apakah permainan terlalu mudah/terlalu sulit untuk dikalahkan?
- Apakah ada fitur yang seharusnya sudah ada tetapi belum ada?

Setelah pengujian dilakukan, akan dihasilkan laporan pengujian yang mencakup semua kesalahan dan beberapa hal yang harus dihilangkan, atau disertakan. Hasil pengujian akan berakhir dalam dua keputusan berbeda: mengulangi siklus produksi baru dari pra- produksi baru lalu menyempurnakan keseluruhan gim, atau maju ke langkah berikutnya, dalam konteks ini: Pengujian Beta.

e. Beta

Jika pengujian dilakukan oleh sesama pengembang, **beta** dilakukan oleh pihak ketiga, seperti penerbit, calon pembeli, pengulas gim, dan lain-lain. Tujuan Pengujian Beta adalah untuk menemukan *bug* yang mengintai dalam gim dan umpan balik pengguna terhadap *build game* saat ini. Hasilnya sama, yaitu laporan pengujian. Laporan tersebut digunakan untuk memutuskan apakah akan memperbaiki lagi gim (ulangi dari pra-produksi) atau gim sudah bagus dan siap dikirim, jadi lanjutkan ke rilis.

f. Rilis

Nah, apa lagi yang bisa dikatakan saat gim siap dikirim? Fase ini terkait dengan peluncuran gim, rilis ke pasar yang sesuai, dan penutupan proyek. Membuat buku pengetahuan, portofolio proyek, dan berbagi pengetahuan penting untuk mempersiapkan proyek berikutnya yang lebih baik.

3. Memasarkan Perangkat Lunak dan Gim

Aktivitas Belajar 3-3

Buatlah uraian singkat yang menggambarkan rangkaian aktivitas kalian, jika kalian akan mengembangkan sebuah gim sutten. Uraian tersebut meliputi:

1. Bagaimana *gameplay*-nya?
2. Karakter apa saja yang ada?

Berhasil membangun perangkat lunak atau gim bukanlah suatu prestasi yang biasa-biasa. Waktu yang dibutuhkannya pun tidak sedikit. Butuh sumber daya yang tidak sedikit pula untuk menyelesaikannya. Menyelesaikan pembangunan perangkat lunak atau gim bukan akhir dari sebuah perjalanan, justru sedang ditunggu oleh perjalanan berikutnya yang lebih sulit. Apakah tujuan dikembangkan perangkat lunak sudah tercapai? Belum.

Perangkat lunak maupun gim yang sudah jadi harus dipasarkan kepada target market kalian. Sangat disayangkan jika kalian memiliki produk yang bagus dan dapat menjawab kebutuhan tetapi tidak ada orang yang tahu. Lantas apa yang harusnya dilakukan?

Setelah kalian berhasil mengembangkan perangkat lunak atau gim, maka produk kalian harus dipasarkan. Sama halnya dengan produk-produk yang lainnya, pengguna tidak akan tahu kualitas produk kalian jika kalian tidak memasarkannya. Kalian harus merangcang strategi pemasaran yang tepat agar produk kalian diketahui dan digunakan oleh pengguna. Perlu kalian ingat bahwa pengembang perangkat lunak bukan hanya kalian. Pengembang- pengembang lain juga mungkin memiliki produk yang sejenis. Di sinilah kompetisi dimulai.

Untuk memasarkan produk perangkat lunak atau gim pada dasarnya sama seperti memasarkan produk lainnya. Apa yang harus kalian lakukan pertama kalinya? Ya, kalian harus melakukan riset pasar terlebih dahulu.

Pemasaran sebuah produk dapat menggunakan prinsip 4P menurut Kotler & Keller, yaitu:

a. *Product* (Produk)
b. *Price* (Harga)
c. *Promotion* (Promosi)
d. *Place* (Tempat)

Dalam memasarkan sebuah produk, kalian harus memastikan produk yang kalian pasarkan merupakan produk yang memiliki mutu baik. Itulah mengapa dalam proses pengembangan perangkat lunak kalian harus melakukan pengujian sebelum perangkat lunak atau gim diserahterimakan kepada pelanggan. Harga yang kompetitif juga menjadi hal yang tidak boleh dilupakan. Harga kompetitif tidak selalu harus murah ya tetapi sesuai dengan kualitas produk kalian. Jangan lupa melakukan promosi ya agar orang lain mengenal kalian dan produk-produk yang kalian hasilkan. Tempat yang strategis menjadi pertimbangan dalam membeli sebuah produk. Buat *landing page* yang baik seperti alamat *website* agar pelanggan percaya bahwa kalian memang benar-benar ada. Selain itu kalian juga harus melihat bagaimana sudut pandang sebagai konsumen. Apa yang harus diperhatikan dari konsumen? Beberapa hal yang kalian harus perhatikan:

a. Apa yang menjadi kebutuhan dan keinginan konsumen (*customer needs and wants*),
b. Biaya konsumen yang harus dikeluarkan oleh konsumen (*cost to the customer)*,
c. Kenyamanan konsumen pada saat membeli produk kalian (*convenience*), dan
d. Komunikasi yang baik dan nyaman (*comunication*).

Pemasaran merupakan upaya untuk meningkatkan daya saing kalian di pasaran. Pemasaran bisa dilakukan secara daring maupun luring. Meskipun saat ini teknologi informasi berkembang pesat, tetapi tidak semua konsumen tahu dan nyaman dengan pemasaran daring. Beberapa contoh pemasaran secara luring: pameran teknologi, pameran inovasi, jaringan, bertemu langsung.

Beberapa teknik yang bisa dilakukan pada pemasaran perangkat lunak dan gim antara lain:

a. Optimalkan penggunaan media sosial.
b. Promosi dari mulut ke mulut (*person to person*).
c. Branding produk.
d. Pemberian insentif (*cashback*, bonus, dan lain-lain).
e. Potongan harga atau diskon.

Aktivitas Belajar 3-4

Berdasarkan aktivitas 3-2 dan 3-3, buatlah deskripsi yang menarik tentang produk yang kalian hasilkan meliputi

- Apa kelebihan produk yang kalian kembangkan?
- Untuk siapa produk tersebut dikembangkan?
- Bagaimana bentuk promosi yang tepat untuk calon pelanggan kalian?

B. Penerapan Budaya Mutu

Aktivitas Belajar 3-5

Deskripsikan apa perbedaan antara produk perangkat lunak dengan produk manufaktur lainnya ditinjau dari:

1. Kompleksitas produk
2. Penampakan
3. Sifat pengembangan dan proses produksi

Pernahkah kalian melihat wujud fisik perangkat lunak atau gim? Ya, perangkat lunak merupakan hasil industri yang tidak kasat mata (tidak ada wujud). Karena tidak berwujud, maka kalian baru akan mengetahui ada cacat atau kesalahan saat perangkat lunak tersebut digunakan atau dijalankan. Oleh karena gim merupakan bagian dari perangkat lunak, maka gim juga memiliki sifat yang sama. Satu-satunya kesempatan untuk mengetahui cacat atau kesalahan adalah pada saat pengembangan.

Lantas bagaimana menjamin kualitas perangkat lunak itu? Sebelum bicara penjaminan kualitas perangkat lunak, kalian harus tahu dulu apa itu kualitas perangkat lunak. IEEE (*Institute of Electrical and Electronics Enginers*) mengemukakan definisi kualitas perangkat lunak sebagai ketercapaian sejauh mana sistem, komponen, atau proses memenuhi kebutuhan yang ditentukan serta memenuhi kebutuhan pelanggan atau pengguna (Galin, 2004). Kualitas perangkat lunak juga merupakan "proses perangkat lunak yang efektif diterapkan dengan

cara menciptakan produk yang berguna yang memberikan nilai terukur bagi mereka yang memproduksinya dan mereka yang menggunakannya" (Presman & Maxim, 2020).

Menjamin kualitas perangkat lunak tidaklah hanya sekadar memastikan hasil akhir seperti halnya produk yang berwujud. Mengendalikan kualitas produk biasanya dilakukan di akhir ketika suatu produk sudah dihasilkan. Menjamin kualitas berarti memastikan kualitas pada setiap prosesnya, berarti kalian harus melihat kembali proses pengembangan perangkat lunak serta memeriksa kembali keterpenuhan kebutuhan teknis. Kalian bisa melakukan pengujian berjenjang pada perangkat lunak meliputi pengujian unit, integrasi, validasi, dan sistem.

1. Strategi Pengujian Perangkat Lunak

Pengujian merupakan langkah awal dari penjaminan perangkat lunak atau gim. Pada awalnya pengujian hanya dilakukan pada tahap akhir pengembangan perangkat lunak, yaitu hanya pada kode program, tetapi mengingat pentingnya deteksi dini kerusakan perangkat lunak, maka pengujian dilakukan pada seluruh bagian dari perangkat lunak atau gim.

Pengujian perangkat lunak atau gim harus dilakukan secara formal karena merupakan bagian yang tidak dapat dipisahkan dari pengembangan perangkat lunak. Pengujian harus direncanakan, dijadwalkan, dan disepakati antara pengembang dan pelanggan. Setiap bagian dari perangkat lunak atau gim diperiksa dan dipastikan kesesuaiannya dengan menjalankan serangkaian skenario tertentu.

Pengujian juga tidak dilakukan oleh pemrogram atau pengembang sendiri tetapi dilakukan oleh sebuah tim independen. Hal ini dilakukan agar tidak terjadi konflik kepentingan dan menjamin pengujian berjalan secara efektif dan profesional. Tim penguji akan menggunakan serangkaian metode pengujian untuk memastikan bahwa perangkat lunak atau gim yang diuji telah sesuai dengan kualitas yang diharapkan.

Untuk merencanakan, menyusun serta menjalankan skenario pengujian, agar efektif dan profesional, maka diperlukan strategi pengujian yang tepat. Ada banyak strategi pengujian yang dapat digunakan pada pengujian perangkat lunak atau gim, tetapi secara umum terdapat dua strategi mendasar yaitu menguji secara keseluruhan setelah selesai, atau menguji secara bertahap mulai dari unit, modul, sampai sistem. Untuk melakukan pengujian bertahap, kalian dapat

melakukannya dari atas ke bawah (*top-down*) atau dari bawah ke atas (*bottom-up*).

2. Teknik Pengujian Perangkat Lunak

Untuk melaksanakan strategi pengujian maka diperlukan teknik pengujian yang sesuai. Ada berbagai macam teknik pengujian yang bisa digunakan. Pada dasarnya teknik tersebut dikelompokkan dalam pengujian struktur internal perangkat lunak dan hasil yang diharapkan. Dua teknik pengujian yang populer adalah *whitebox testing* dan *blackbox testing*.

Whitebox Testing merupakan teknik pengujian yang digunakan untuk memastikan struktur logika program berjalan dengan baik sesuai dengan spesifikasinya. Beberapa hal yang diuji dalam *Whitebox Testing* adalah:

- Memastikan bahwa semua jalur independen yang terdapat pada kode program benar-benar dijalankan, minmal satu kali jalan.
- Memastikan semua keputusan logika (kondisional) bekerja dengan baik pada keadaan bernilai benar (*true*) dan salah (*false*).
- Memastikan semua pengulangan bekerja dengan baik berdasarkan batas yang telah ditentukan (*boundary*).
- Memastikan semua struktur data internalnya valid

Blackbox Testing pada dasarnya merupakan komplemen dari *Whitebox Testing* dengan mengungkap sisi yang berbeda dalam pengujiannya. *Blackbox Testing* memfokuskan diri pada kebutuhan fungsional dan domain informasi perangkat lunak. Oleh karena sifatnya adalah komplemen, teknik ini digunakan setelah *Whitebox Testing* selesai dilakukan. Pengujian dilakukan dengan memberikan serangkaian masukan yang dikondisikan untuk mewakili keadaan masukan yang benar dan masukan yang salah pada setiap fungsional. Beberapa hal yang diuji pada *Blackbox Testing* adalah:

- Menemukan dan memperbaiki fungsi yang salah atau hilang.
- Menemukan dan memperbaiki kesalahan pada antarmuka.
- Menemukan dan memperbaiki kesalahan dalam struktur data atau akses basis data eksternal.

- Menemukan dan memperbaiki kesalahan perilaku atau kinerja.
- Menemukan dan memperbaiki inisialisasi dan kesalahan terminasi.

C. Manajemen Proyek Perangkat Lunak dan Gim

Mengembangkan perangkat lunak dan gim bukan hanya sekadar aktivitas biasa. Aktivitas ini melibatkan banyak sumber daya serta pemangku kepentingan. Risiko kegagalannya pun tidak kecil, oleh karena itu harus dikelola dengan sebaik mungkin sehingga tercapai tujuan yang diharapkan. Dalam mengembangkan perangkat lunak, biasanya kita mengenal istilah proyek. Pengembang mengistilahkan "mendapatkan proyek" untuk mengerjakannya. Tahukah kalian apa sebenarnya proyek itu? Bagaimana mengelola proyek yang benar? Apa saja yang harus dilakukan dalam mengelola sebuah proyek? Nah untuk mengetahui itu semua, kalian harus memahami *Project Management Body of Knowledge* (PMBOK) menjadi acuan pengelolaan sebuah proyek.

Project Manajemen Institute (PMI) mendefinisikan proyek sebagai "Proyek adalah usaha sementara yang dilakukan untuk menciptakan produk atau jasa yang unik" (*Project Management Institute*, 2013). Sementara yang dimaksud bisa dalam jangka waktu beberapa minggu, beberapa bulan, atau beberapa tahun. Sifat sementara menunjukkan bahwa proyek memiliki awal dan akhir yang pasti dan telah ditentukan. Akhir dari sebuah proyek adalah pada saat tujuan proyek telah tercapai. Sebuah proyek juga dapat dihentikan di tengah jalan jika klien berkeinginan menghentikan proyek tersebut.

Setiap proyek yang dikerjakan akan menghasilkan produk, layanan, atau hasil yang unik. Apa yang dihasilkan proyek mungkin dalam bentuk yang berwujud atau tidak berwujud. Kegiatan dalam sebuah proyek tidak jarang merupakan aktivitas yang berulang dari aktivitas proyek sebelumnya. Meski demikian, setiap proyek memiliki keunikan masing- masing. Mengapa bisa seperti itu? Setiap proyek memiliki permasalahan yang berbeda satu dengan yang lain. Lokasi yang berbeda, suasana yang berbeda, kebutuhan yang berbeda, desain yang berbeda, pemangku kepentingan yang berbeda, bahkan bisa jadi tim yang terlibat juga berbeda.

Untuk menjalankan proyek, perlu sebuah manajemen yang disebut Manajemen Proyek. Manajemen Proyek adalah "aplikasi pengetahuan, keterampilan, alat dan teknik untuk kegiatan proyek untuk memenuhi persyaratan proyek". (Project Management Institute, 2013). Manajemen Proyek Perangkat Lunak adalah seni untuk mendefinisikan, merencanakan, melaksanakan, dan

memantau aktivitas yang akan membawa produk perangkat lunak menjadi ada. Pimpinan proyek (Manager Proyek) berusaha untuk memenuhi batasan rangkap tiga dengan menyeimbangkan sasaran ruang lingkup, waktu, dan biaya proyek.

Apa yang dilakukan dalam proyek? Secara umum siklus hidup proyek terdiri dari:

- **Inisiasi** – Lingkup proyek; menyewa proyek, mengidentifikasi pemangku kepentingan.
- **Perencanaan** – Mengembangkan rencana proyek. Mengumpulkan kebutuhan, mengidentifikasi jadwal, menentukan ruang lingkup, mengestimasi biaya, menentukan standar kualitas, mengidentifikasi sumber daya manusia, mengidentifikasi risiko, dan merencanakan manajemen pengadaan.
- **Pelaksanaan** – Jalankan rencana. Mengarahkan dan mengelola pekerjaan proyek; melakukan penjaminan mutu; mengelola dan mengembangkan tim proyek; melakukan pengadaan.
- **Pemantauan dan Pengendalian** – Memantau kemajuan dari proyek. Memantau dan mengendalikan pekerjaan proyek, mengelola perubahan ruang lingkup jika terjadi, memantau dan memastikan jadwal, mengendalikan kualitas; mengendalikan risiko; pengendalian pengadaan
- **Penutupan** – Tutup proyek: Tutup proyek; tutup pengadaan.

Aktivitas Belajar 3-6

Lihat kembali Aktivitas 3-2, diskusikan dengan anggota kelompok kalian, buat uraian bagaimana mengatur dan mengendalikan rencana pembangunan perangkat lunak tersebut

D. Rekayasa Kebutuhan

Pernahkah kalian datang ke toko komputer untuk membeli laptop? Ketika kalian membeli laptop maka penjual akan bertanya kepada kalian, mencari laptop seperti apa? Mau digunakan untuk apa? Kemudian kalian akan menjelaskan seperti apa laptop yang kalian inginkan. Nah berarti kalian sudah menyampaikan kebutuhan kalian kepada penjual laptop tadi.

Dalam pengembangan perangkat lunak dan gim pun berlaku hal yang sama. Sebagai pengembang, kalian harus bisa memahami apa yang dibutuhkan oleh pelanggan. Ingat, pada bahasan sebelumnya kalian sudah mengetahui bahwa agar produk yang kalian hasilkan laku, maka harus sesuai dengan kebutuhan pelanggan.

Kebutuhan adalah ”atribut yang diperlukan dalam suatu sistem, pernyataan yang mengidentifikasi kemampuan, karakteristik, atau faktor kualitas suatu sistem agar memiliki nilai dan kegunaan bagi pelanggan atau pengguna” (Young, 2004). Kebutuhan sangat penting karena menyediakan hal dasar untuk semua pekerjaan pengembangan berikutnya. Setelah kebutuhan ditetapkan, pengembang dapat memulai pekerjaan teknis lainnya seperti: rancangan sistem, implementasi, pengujian, dan operasional.

Dalam banyak kasus, pengembang perangkat lunak dan gim seringkali melupakan kebutuhan. Ada kecenderungan bahwa Manajer Proyek (PM) beranggapan bahwa pekerjaan utama dari pengembangan perangkat lunak dan gim adalah aktivitas "***coding***". Berdasarkan pengalaman industri, keputusan yang lebih baik adalah meluangkan lebih banyak waktu dalam pengumpulan kebutuhan, analisis permasalahan, dan aktivitas manajemen, tidak terburu-buru untuk masuk pada aktivitas pengkodean. Jika kebutuhan telah didefinisikan dengan baik, maka tidak memerlukan waktu yang banyak untuk melakukan aktivitas berikutnya.

Berdasarkan peruntukannya, kebutuhan dapat digolongkan menjadi:

- Kebutuhan pengguna (*user requirements*), berupa pernyataan dalam bahasa sehari-hari yang mudah dipahami. Dapat juga ditambah dengan diagram yang menggambarkan layanan apa saja yang ada pada sistem serta bagaimana menjalankannya. Kebutuhan ini dibuat untuk berkomunikasi dengan pelanggan. Dalam beberapa pendekatan pengembangan perangkat lunak, biasa juga disebut dengan "*user story*”.
- Kebutuhan sistem (*system requirements*), berupa dokumen terstruktur yang mendeskripsikan secara terperinci mengenai layanan, fungsi, serta pengoperasian sistem. Dalam kebutuhan ini juga dideskripsikan apa saja yang harus dilaksanakan dan dapat dijadikan sebagai kontrak.

Berdasarkan bentuknya, kebutuhan perangkat lunak dapat dikelompokkan menjadi kebutuhan fungsional dan non fungsional (*funtional and non-functional requirements*).

- **Kebutuhan fungsional** merupakan pernyataan yang mendeskripsikan layanan-layanan apa saja yang harus disediakan oleh sistem, termasuk di dalamnya bagaimana sistem bereaksi terhadap sebuah masukan tertentu yang diberikan serta harus melakukan apa pada situasi tertentu. Kebutuhan ini erat kaitannya dengan fitur-fitur apa saja yang dimiliki oleh sistem tersebut dan berasal dari berbagai pemangku kepentingan.

- **Kebutuhan non-fungsional**, merupakan pernyataan kebutuhan berupa batasan-batasan (*constraints*) terhadap layanan yang diberikan (*fungsional*) oleh sistem meliputi kecepatan, waktu, keselamatan, keamanan, performa.

Untuk merekayasa kebutuhan kalian harus mulai dengan keluaran- keluaran yang dihasilkan oleh sistem. Keluaran sistem umumnya berupa informasi yang diperlukan oleh pengguna atau pemangku kepentingan lainnya. Berdasarkan keluaran tersebut, kalian dapat menemukan kebutuhan fungsional maupun non fungsional. Proses menemukan, mengumpulkan, menganalisis, memeriksa, dan menetapkan kebutuhan dinamakan Rekayasa Kebutuhan (*requirement engineering*).

Proses Rekayasa Kebutuhan meliputi:

1. Permulaan (*Inception*)

   Untuk memulai sebuah pengembangan perangkat lunak, kalian harus memahami terlebih dahulu permasalahannya. Apa alasan perangkat lunak ini dibuat, siapa saja pemangku kepentingan yang terlibat dalam solusi, kapan harus diselesaikan, bagaimana cara kerjanya, dan lain-lain, merupakan hal-hal yang harus kalian pahami terlebih dahulu. Komunikasi dengan semua pemangku kepentingan perlu dilakukan agar pengembangan nantinya berjalan efektif.

2. Elisitasi (*Elicitation*)

   Elisitasi adalah mentransfer ide dari pemangku kepentingan ke tim perangkat lunak dengan lancar dan tanpa penundaan. Sangat mungkin bahwa persyaratan baru akan terus muncul sebagai produk berulang perkembangan terjadi. Bagian penting dari elisitasi adalah memahami tujuan bisnis

3. Elaborasi (*Elaboration*)

   Tugas *elaboration* (elaborasi) berfokus pada pengembangan model kebutuhan yang mengidentifikasi berbagai aspek fungsi perangkat lunak, perilaku, dan informasi. Pada proses ini, kalian didorong untuk menyempurnakan apa yang menjadi kebutuhan pengguna termasuk bagaimana interaksi pengguna dengan perangkat lunak. Proses ini bisa dilakukan berulang-ulang untuk memastikan tidak ada hal yang terlewatkan pada kebutuhan sistem.

   Elaborasi adalah hal yang baik, tetapi kalian perlu tahu kapan harus berhenti. Kuncinya adalah untuk menggambarkan masalah dengan

cara yang membangun dasar yang kuat untuk desain dan kemudian melanjutkan. Jangan terobsesi dengan detail yang tidak perlu.

4. Negosiasi (*Negotiation*)

   Rasa tidak puas bagi pelanggan maupun pengguna tentu bukan hal yang baru. Pelanggan atau pengguna pasti akan meminta lebih dari apa yang telah dirumuskan sebelumnya. Tidak jarang juga muncul usulan kebutuhan baru yang bertentangan dengan kebutuhan sebelumnya dengan berbagai macam argumen. Perbedaan- perbedaan memandang masalah serta kebutuhan yang saling bertentangan tentu tidak boleh dibiarkan, harus dinegosiasikan agar bisa disepakati. Pada proses ini, pelanggan dan pengguna diminta untuk mengurutkan tingkat kepentingan dari kebutuhan tersebut serta mendiskusikannya dengan pemangku kepentingan yang lain. Pendekatan prioritas kepentingan inilah yang digunakan sebagai dasar penentuan akhir kebutuhan. Setiap kebutuhan ditinjau lagi dari risiko, biaya, dan sumber daya yang digunakan. Bisa saja suatu kebutuhan digabungkan dengan kebutuhan lainnya, dimodifikasi atau bahkan dihilangkan atas dasar pertimbangan-pertimbangan tersebut. Tidak boleh ada yang merasa menang dan kalah dalam proses ini.

5. Spesifikasi (*Specification*)

   Hasil kesepakatan kebutuhan harus dirumuskan dan didokumentasikan dalam sebuah dokumen yang berisi model grafis, model matematis, kumpulan skenario, dan hal-hal lain yang berkenaan dengan kebutuhan yang dinamakan Spesifikasi Kebutuhan Perangkat Lunak. Spesifikasi dirumuskan dalam suatu *template* tertentu yang disepakati bersama. Spesifikasi kebutuhan inilah yang nantinya digunakan sebagai dasar untuk menentukan tahapan-tahapan berikutnya.

6. Validasi (*Validation*)

   Apa yang dihasilkan selama rekayasa kebutuhan dinilai kualitasnya, terlebih pada konsistensi. Kalian dapat menggunakan model analisis untuk memastikannya. Pastikan juga bahwa semua kebutuhan telah ditulis dengan jelas, tidak ada kelalaian, dan sesuai dengan standar. Semua pemangku kepentingan dapat melakukan validasi dan memriksa spesifikasi, termasuk jika ada kebutuhan yang tidak realistis. Jika ditemukan kesalahan, kelalaian, informasi yang hilang, atau ketidaksesuaian dengan standar, lakukan perbaikan.

Aktivitas Belajar 3-7

Silakan berdiskusi dengan kelompokmu kemudian buatlah uraian singkat menggunakan 5W1H untuk mendeskripsikan kebutuhan perangkat lunak penjualan pada minimarket.

1. *What*: Deskripsikan perangkat lunak apa yang akan dibuat ini. Berisikan setidaknya fitur utama apa saja yang dimiliki oleh perangkat lunak ini.
2. *Why*: deskripsikan permasalahan apa yang mendasari fitur-fitur tersebut harus ada dalam perangkat lunak yang akan dikembangkan.
3. *How*: deskripsikan bagaimana cara kerja setiap fitur yang akan ada pada perangkat lunak yang akan dikembangkan.
4. *Who*: uraikan siapa saja yang berkepentingan dengan perangkat lunak tersebut dan masing-masing peran punya tugas dan tanggung jawab seperti apa.
5. *Where*: tentukan perangkat lunak ini akan diimplementasikan di mana.
6. *When*: kapan estimasi waktu untuk perangkat lunak ini rilis.

Uji Kompetensi

Dalam test ini kalian harus membaca dengan cermat dan teliti setiap butir soal di bawah ini. Kemudian berdasarkan uraian materi di atas tulislah jawabannya pada lembar jawaban yang telah disediakan.

1. Pada saat kalian mengembangkan sebuah perangkat lunak atau gim, ada serangkaian kegiatan atau langkah-langkah yang harus dilakukan secara berurutan maupun berulang. Uraikan dengan singkat, mengapa kita harus melakukan langkah-langkah tersebut?

   .....

   .....

   .....

2. Jika kalian mengembangkan sebuah perangkat lunak penjualan pada minimarket, tentu ada berbagai pemangku kepentingan yang berinteraksi secara langsung maupun tidak langsung dengan perangkat lunak tersebut. Ada berbagai kebutuhan juga yang muncul dari setiap pemangku kepentingan. Menurut kalian, siapa sajakah pemangku kepentingan yang terlibat? Apa kebutuhan setiap pemangku kepentingan terhadap perangkat lunak tersebut?

.....

.....

.....

3. Jika kalian mengembangkan gim (permainan) ular tangga, tentunya melibatkan serangkaian aktivitas pengembangan mulai dari inisiasi sampai dengan rilis. Setiap aktivitas memiliki tantangan tersendiri yang dapat menentukan keberhasilan pengembangannya. Menurut kalian, aktivitas mana yang paling menentukan dalam fase pengembangan gim?

   .....

   .....

   .....

4. Agar menjadi perangkat lunak dan gim yang berkualitas, perangkat lunak dan gim harus dilakukan penjaminan mutu. Salah satu kegiatan dalam penjaminan mutu perangkat lunak dan gim adalah melakukan pengujian. Diperlukan strategi dan teknik yang tepat agar pengujian berhasil dilakukan. Ada berbagai macam teknik pengujian yang bisa digunakan dengan kelebihannya masing-masing. Teknik pengujian apa saja yang umumnya digunakan pada pengujian perangkat lunak dan gim? Berikan penjelasan kelebihan masing-masing.

   .....

   .....

   .....

5. Mengembangkan perangkat lunak dan gim, berarti kalian melaksanakan sebuah proyek. Proyek pengembangan perangkat lunak dan gim harus dikelola dengan baik dan tidak asal-asalan agar menghasilkan perangkat lunak dan gim yang berkualitas. Ada berbagai hal yang harus diperhatikan ketika mengelola sebuah proyek perangkat lunak. Menurut kalian, bagaimana seharusnya sebuah proyek perangkat lunak dikelola?

   .....

   .....

   .....

Pengayaan

1. Buku

- Mohapatra, P. K. (2010). Software Engineering (A Lifecycle Approach). New Delhi: New Age International (P) Limited, Publishers.
- Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition. New York: McGraw Hill Education.
- Project Management Institute. (2013). A Guide To Project Management of Knowledge (PMBOK Guide)-Fifth Edition. Pennsylvania: Project Management Institute, Inc.
- Project Management Institute. (2013). Software Extension to the PMBOK Guide Fifth Edition. Pennsylvania: Project Management Institute, Inc.
- Ramadan, R., & Widyani, Y. (2013). Game Development Life Cycle Guidelines. 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 95-100). Sanur Bali, Indonesia: IEEE.
- Sommerville, I. (2016). Software Engineering. Tenth Edition. Harlow: Pearson Education Limited.
- Young, R. R. (2004). The Requirements Engineering Handbook. Norwood: Artech House Inc.

2. Artikel

- Devin Pickell. The 7 Stages of Game Development. https://www.g2.com/articles/stages-of-game-development. Akses terakhir 21 Oktober 2022.
- Juego Studios. What Is the Game Development Life Cycle?. https://www.gamedeveloper.com/business/what-is-the-game- development-life-cycle-. Akses terakhir 21 Oktober 2022
- Java Point. Requirement Engineering. https://www.javatpoint.com/software-engineering-requirement- engineering. Akses terakhir 21 Oktober 2022.
- Alex Melnichuk. Top 8 Software Development Models. https://ncube.com/blog/top-software-development-models. Akses terakhir 21 Oktober 2022.
- David Martin. Software Requirements Analysis — The Key to Developing Great Software. https://medium.com/illumination/software-requirements-analysis-the-key-to-developing-great-software-20162c5b0580. Akses terakhir 21 Oktober 2022.

Refleksi

Berilah tanda (√) pada kotak yang dianggap sesuai! Setelah mempelajari bab ini, bagaimanakah penguasaankalian terhadap materi-materi berikut?

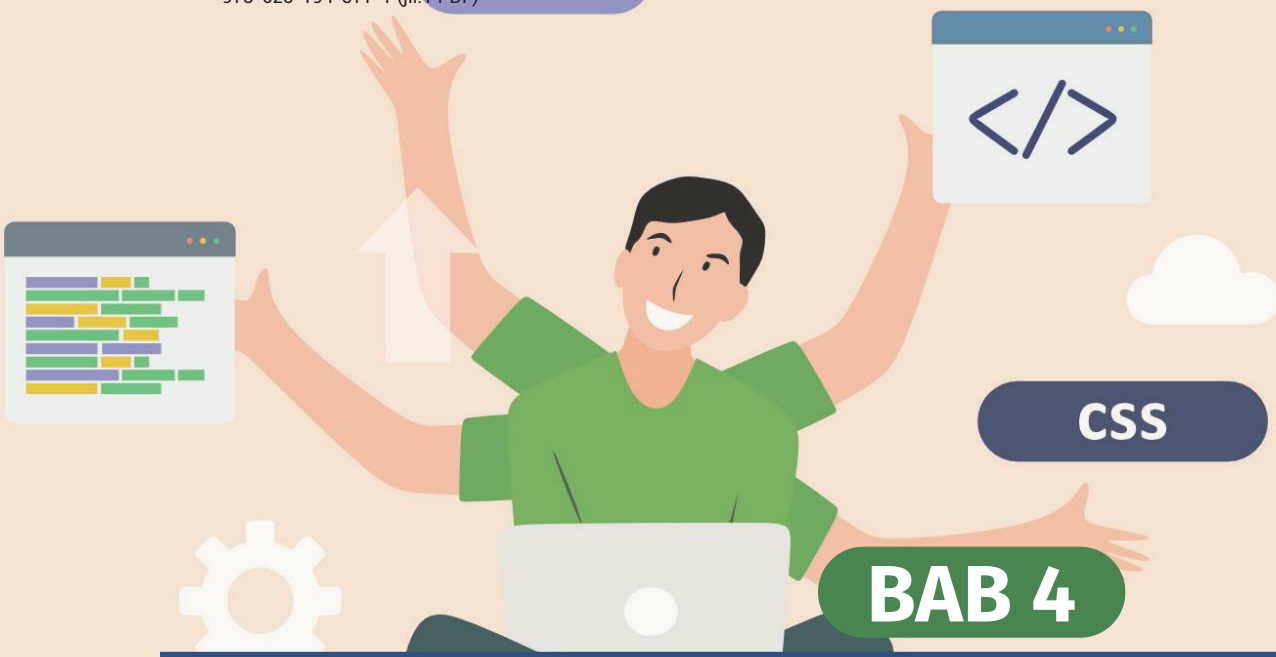
No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Proses Pengembangan, dan Pemasaran, Perangkat Lunak dan Gim			
2.	Penerapan Budaya Mutu			
3.	Penerapan Manajemen Proyek			
4.	Rekayasa Kebutuhan			

1. Dari materi-materi tersebut, bagian manakah yang paling kalian sukai? Mengapa?
2. Apa manfaat yang kalian dapatkan setelah mempelajari materi bab ini untuk kehidupan sehari-hari?
3. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim
untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)



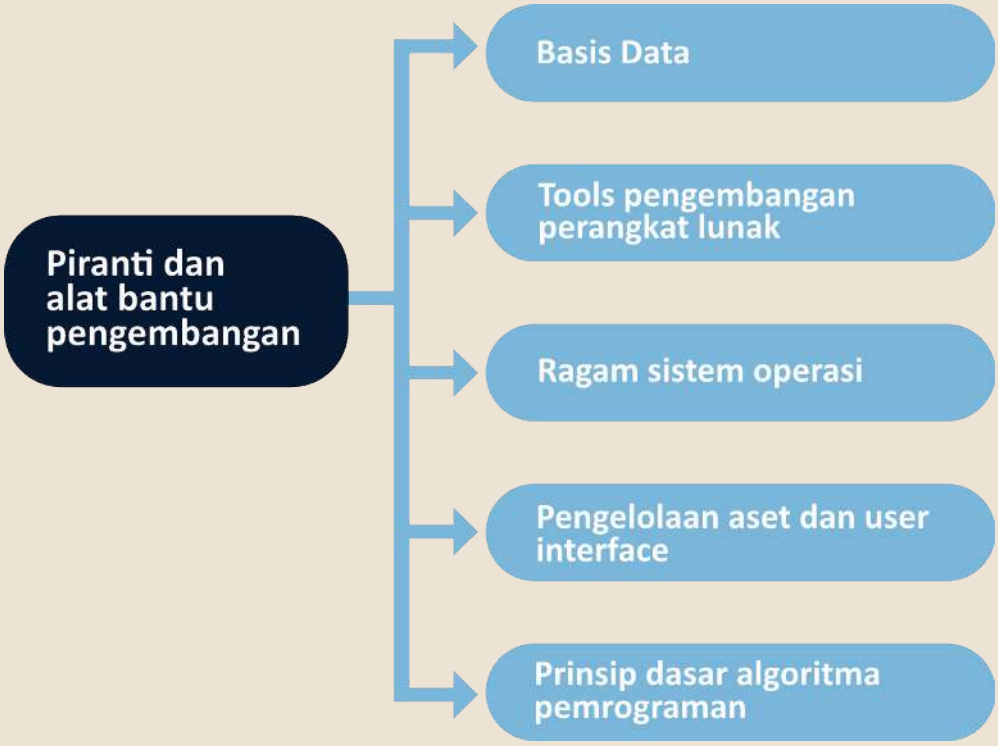
BAB 4

Piranti dan Alat Bantu Pengembangan

Tujuan Pembelajaran

Setelah mempelajari materi ini kalian diharapkan dapat menggunakan basis data, tools pengembangan perangkat lunak, ragam sistem operasi, pengelolaan aset dan user interface serta menerapkan prinsip dasar algoritma pemrograman sebagai perangkat dan aplikasi di bidang Perangkat Lunak dan Gim.

Peta Materi



Kata Kunci

♦ Algoritma ♦ Basis Data ♦ Manajemen Aset ♦ User Interface
♦ Sistem Operasi ♦ Alat Bantu.

Gambar 4.1 Proses membangun sebuah rumah
Sumber: Marwondo, Rini Melati, dan Dana R. N. Adnan/2023

Perhatikan gambar di atas, apa yang terjadi jika tidak ada peralatan tersebut?

Apersepsi

Pada Bab sebelumnya, kalian telah memahami dengan baik bagaimana sebuah perangkat lunak dan gim dikembangkan. Tentu saja kalian tidak akan bisa mengerjakan semua seorang diri, ada pihak-pihak yang terlibat di dalamnya. Nah, ibarat membangun rumah, tentu ada pihak-pihak yang dilibatkan. Mulai dari arsitek, mandor, tukang, kuli, tukang instalasi listrik, desain interior, dan pihak-pihak lainnya. Masing-masing pihak bekerja dengan alat bantu yang berbeda-beda, kan? Nah, begitu juga kalian dalam mengembangkan perangkat lunak dan gim. Ada berbagai alat bantu yang dapat digunakan untuk menunjang keberhasilan pembangunan perangkat lunak dan gim tersebut.

A. Basis Data

Secara umum, perangkat lunak bekerja dengan mengolah data menjadi informasi. Informasi menjadi sangat penting bagi sebagian besar organisasi. Data dan informasi dapat memiliki berbagai bentuk dapat berupa angka, teks, gambar, suara. Tentunya data dan informasi tidak boleh hanya sekadar numpang lewat, atau hanya masuk dan keluar begitu saja. Data atau informasi ada yang harus

disimpan untuk dapat digunakan kembali pada kepentingan yang lainnya. Nah, tahukah kalian, di mana data tersebut disimpan?

Data yang bersifat temporer (sementara) biasanya akan disimpan pada memori utama atau pada *buffer* yang akan menghilang saat perangkat lunak dihentikan atau ditutup (*volatile*). Sedangkan data yang bersifat jangka panjang akan disimpan pada *datastore*. Sekumpulan data dan informasi yang tersimpan bersama dengan redudansi terkontrol yang digunakan untuk melayani satu atau lebih aplikasi secara optimal membentuk pangkalan data atau biasa disebut dengan basis data (Gupta & Mittal/2017).

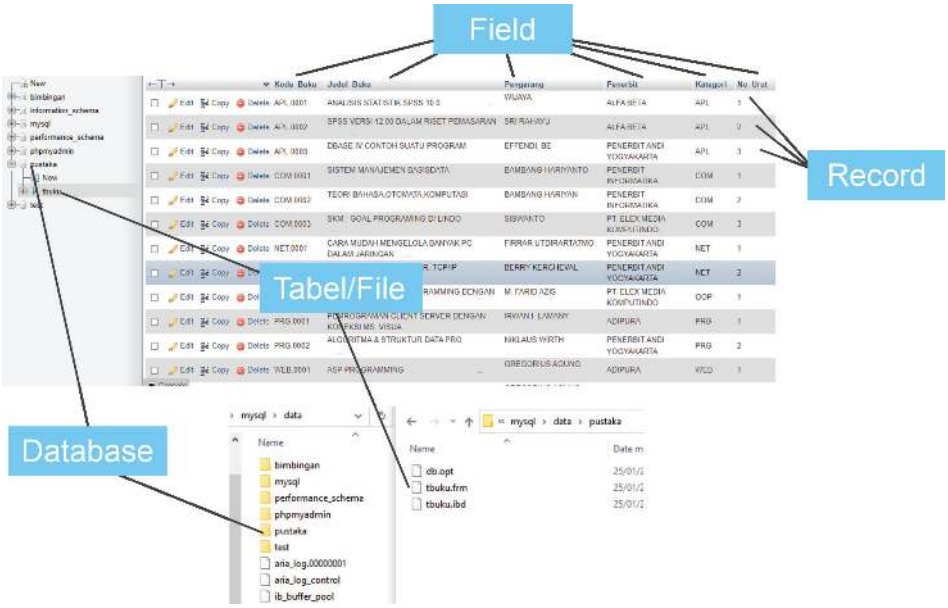
Data disimpan sedemikian rupa sehingga terlepas dari program yang digunakan oleh orang-orang untuk mengakses data. Pendekatan yang digunakan dalam menambah data baru, memodifikasi dan mengambil data yang ada dari basis data adalah pendekatan yang umum dan terkontrol.

Basis data sebenarnya mirip dengan lemari arsip yang di dalamnya tersimpan data-data berupa berkas-berkas yang dikelompokkan berdasarkan kategori tertentu. Keduanya memiliki tujuan yang sama yaitu pengaturan data/arsip sehingga memudahkan dalam pencarian ketika diperlukan (Fathansyah, 2015). Yang membedakan keduanya hanya pada media penyimpanan yang digunakan. Meskipun mirip dengan lemari arsip, akan tetapi basis data bukan hanya digunakan sebagai media penyimpanan. Basis data menekankan pada pengaturan, pemilahan, pengelompokan, dan pengorganisasian data yang akan disimpan dalam bentuk tabel-tabel terpisah atau kolom-kolom (*field*) dalam setiap tabelnya.

Sebuah basis data dibangun dari beberapa elemen yaitu *field*, *record*, dan *file*. Secara singkat elemen-lemen tersebut dapat dijelaskan sebagai berikut.

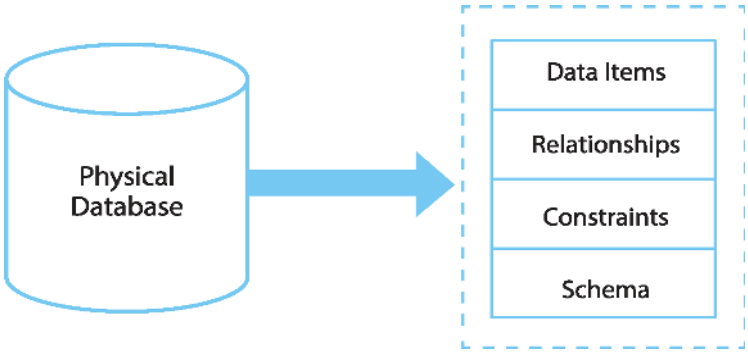
1. *Field*. Merupakan unit terkecil dari data yang biasa juga disebut dengan item data atau elemen data. Nama, Alamat, No HP, Jenis Kelamin merupakan contoh *field* yang masing-masing memiliki tipe data tertentu. *Field* inilah yang akan diberi nilai.
2. *Record*. Merupakan kumpulan *field* yang saling terhubung secara logis. *Record* biasa juga disebut dengan baris.
3. *File*. Merupakan kumpulan atau himpunan *record*. Biasanya mewakili sebuah tabel.

Untuk memperjelas elemen-elemen tersebut, perhatikan gambar 4.2



Gambar 4.2 Elemen-elemen basis data.
Sumber: Marwondo, Rini Melati/2023

Sebuah basis data tentu tidak berdiri sendiri, ada komponen-komponen yang menyusunnya. Komponen-komponen basis data dapat digambarkan seperti pada gambar 4.3.



Gambar 4.3 Komponen basis data.
Sumber: Gupta & Mittal/2017

1. *Data Items* (Item Data). Elemen-elemen informasi yang berhimpun dalam suatu entitas.
2. *Relationship* (Relasi). Hubungan antara satu entitas dengan entitas yang lainnya.
3. *Constraints* (Batasan). Status yang menyatakan kebenaran suatu basis data.

4. *Scheme* (Skema). Menggambarkan organisasi di dalam basis data, terdiri dari definisi berbagai jenis *record* dalam basis data, item data yang berhimpun di dalamnya serta pengaturan pengelompokannya. Struktur penyimpanan database dijelaskan oleh skema penyimpanan. Skema konseptual mendefinisikan struktur data yang disimpan. Skema eksternal mendefinisikan tampilan basis data untuk pengguna tertentu.

Operasi dasar yang dimiliki basis data meliputi (Fathansyah, 2015):

1. Penciptaan basis data (*Create Database*)
2. Penghapusan basis data (*Drop Database*)
3. Penciptaan tabel (*Create Table*)
4. Penghapusan tabel (*Drop Table*)
5. Pengisian data baru (*Insert*)
6. Pengambilan atau pencarian data (*Retrieve/Search*)
7. Pengkinian data (*Update*)
8. Penghapusan data (*Delete*)

1. Sistem Basis Data

Jika basis data merupakan sekumpulan data dan informasinya, maka bagaimana dia dikelola? Basis data dapat dikatakan hanyalah objek yang bersifat pasif (menunggu). Mengapa demikian? Basis data ada karena ada yang membuatnya, tetapi akan sangat tidak memiliki manfaat jika tidak ada pengelola dan penggeraknya. Untuk mengelola basis data diperlukan sebuah sistem yang terdiri dari program serta basis data itu sendiri. Sistem ini kemudian dikenal dengan **sistem basis data**. Sistem basis data merupakan sekumpulan tabel data yang saling berhubungan dan sekumpulan program yang memungkinkan beberapa pengguna atau program lainnya untuk mengakses dan memanipulasi tabel-tabel data tersebut.

Sebuah sistem basis data secara lengkap memiliki komponen-komponen sebagai berikut:

a. Perangkat keras. Berupa media penyimpanan sekunder yang digunakan untuk menyimpan data seperti hardisk, SSD, dan lain-lain serta perangkat lainnya yang mendukung dalam menjalankan DBMS.

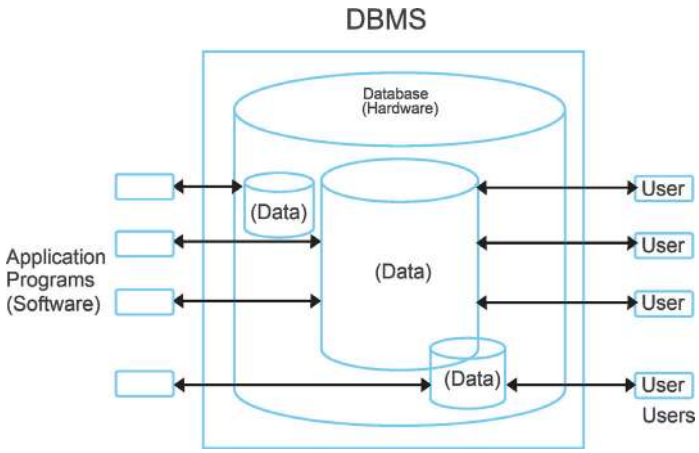
b. Sistem operasi. Sistem yang diperlukan untuk menyelaraskan semua sumber daya yang ada pada komputer baik perangkat keras maupun perangkat lunak.

c. Basis Data. Kumpulan data itu sendiri yang akan dikelola.

d. Sistem Pengelola Basis Data (DBMS). Perangkat lunak yang akan menentukan bagaimana basis data diorganisasi, disimpan, diperbarui, serta dipanggil kembali.

e. Pengguna. Orang-orang yang berinteraksi dengan sistem basis data baik secara langsung maupun tidak langsung. Secara umum ada empat jenis pengguna yang berinteraksi dengan sistem basis data yaitu pemrogram, pengguna langsung, pengguna akhir atau pengguna naif dan *Administrator Basis Data* (DBA).

f. Perangkat lunak lain. Basis data tidak hanya dikelola oleh DBMS, tetapi akan digunakan juga oleh perangkat lunak lain yang dibangun secara khusus melalui mekanisme tertentu.



Gambar 4.4 Sistem basis data.
Sumber: Gupta & Mittal/2017

2. *Database Management System* (DBMS)

Salah satu komponen penting dalam sistem basis data adalah DBMS (*Database Management System*). Apa sih sebenarnya DBMS itu? Seperti halnya yang sudah dijelaskan sebelumnya, basis data tidak akan bermanfaat jika tidak dikelola dan digerakkan. Nah, sesuatu yang digunakan untuk mengelola tersebut disebut dengan Sistem Manajemen

Basis Data atau Database Management System (DBMS). DBMS merupakan kumpulan data yang saling terkait dan satu set program

untuk mengakses data tersebut (Silberschatz, Korth, & Sudarshan, 2020). Beberapa contoh DBMS yang populer antara lain: MS Access, MySQL, MS SQLServer, Oracle Database, PostgreSQL, Sybase, FoxPro, Paradox, MongoDB, IBM Db2, Elasticsearch, Redis, SQLite, dan lain-lain.

Mengapa kalian perlu DBMS? Jika kalian telah memiliki arsip-arsip yang telah tersimpan di map-map, apakah map-map arsip tersebut dibiarkan berserakan begitu saja? Tentu Kalian butuh tempat menyimpannya bukan? Selain tempat, kalian juga pasti memerlukan bagaimana mekanisme untuk mencari arsip tersebut jika diperlukan, menyimpan arsip baru, mengelompokkan arsip yang ada, serta hal-hal lain yang berhubungan dengan pengelolaan arsip.

Tujuan utama keberadaan DBMS adalah untuk menyediakan cara menyimpan dan mengakses informasi yang ada dalam sebuah basis data secara nyaman dan efisien. Ada banyak keuntungan yang akan kalian dapatkan jika menggunakan DBMS, antara lain:

a. Mengendalikan duplikasi atau redudansi data. Dengan adanya integrasi berkas data, maka kecenderungan duplikasi dapat dikendalikan. Jika memang harus terjadi data yang berulang pada berkas yang berbeda, maka perubahan dapat dilakukan serentak melalui mekanisme tertentu.
b. Konsistensi data. Meskipun beberapa pengguna dapat melakukan perubahan data baik melalui mekanisme tambah, ubah, maupun hapus, pengguna yang lain juga akan mendapatkan perubahan data yang sama.
c. Integritas data. Dalam DBMS, terdapat mekanisme yang mengatur hak akses masing-masing pengguna sehingga integritas data terjaga dengan baik. Pengguna yang tidak berhak tidak akan dapat melakukan perubahan data.
d. Berbagi data. DBMS memungkinkan beberapa aplikasi mengakses data yang sama (*data sharing*). Setiap aplikasi yang dikembangkan tidak perlu harus menyediakan ulang data yang sudah pernah dibuat sebelumnya atau data yang sudah dikelola oleh aplikasi lainnya. Tentu mekanisme berbagi data harus diatur sedemikian rupa sehingga integritas dan keamanan data tetap terjaga.
e. Meningkatkan keamanan data. Pada DBMS data dapat diatur hak akses pengguna dan juga enkripsi terhadap data. *Administrator*

Basis Data dapat menetapkan skema autentifikasi yang tepat untuk mengakses basis data, jika diperlukan dapat juga menerapkan pemeriksaan tambahan untuk melindungi data. Tentunya tingkat keamanan dapat berbeda untuk berbagai jenis data dan operasi yang dimilikinya.

f. Meningkatkan produktivitas pemrogram. DBMS menyediakan berbagai fungsi baku yang dapat digunakan pada kode program. Hal ini membuat pemrogram lebih fokus untuk membangun fungsi yang diperlukan oleh pengguna.

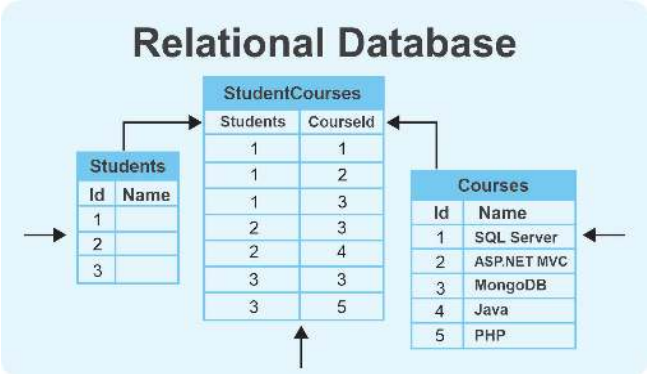
g. Memudahkan pemeliharaan. DBMS memiliki independensi data dengan penyediaan fungsi atau prosedur tertentu. Fungsi atau prosedur tersebut tidak perlu mekanismenya dituliskan pada kode program. Hal ini akan memudahkan jika terjadi perubahan mekanisme, tidak perlu mengubah kode program.

h. Memudahkan penelusuran dan pemulihan. Jika terjadi kegagalan perubahan data baik yang disebabkan oleh mekanisme program maupun faktor lainnya, DBMS dapat mengembalikan keadaan data seperti pada keadaan peristiwa kegagalan tersebut terjadi. DBMS memiliki mekanisme *backup* dan *restore* yang memungkinkan kalian membuat cadangan data serta mengembalikan jika diperlukan.

Berdasarkan sistem pemodelannya, DBMS dikelompokkan menjadi tiga jenis, yaitu:

a. *Basis Data Relasional* atau *Relational Database Management System* (RDBMS).

   Merupakan DBMS yang memiliki pola hubungan relasi. Data direpresentasikan dalam bentuk tabel dua dimensi yang saling dihubungkan satu sama lainnya. Setiap tabel terdiri dari kolom dan baris. Kolom atau biasa juga disebut dengan *field*, digunakan untuk mendefinisikan struktur, tipe, dan atribut data. Baris atau disebut juga dengan record digunakan untuk menyimpan isian data. RDBMS adalah jenis DBMS terpopuler saat ini. RDBMS memiliki antarmuka yang *user-friendly* dan untuk mengoperasikannya dapat menggunakan *Structured Query Language* (SQL).



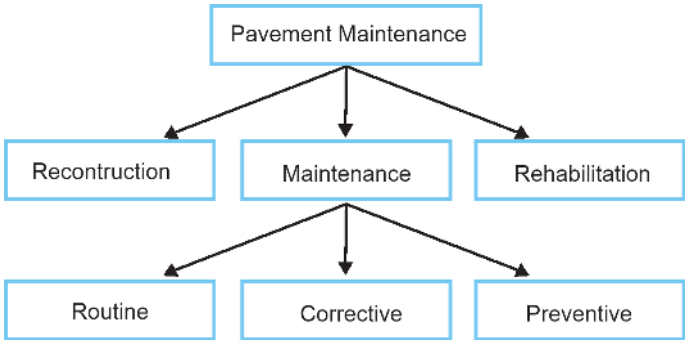
Gambar 4.5 Contoh RDBMS.
Sumber: Ariffudin/2022

Contoh RDBMS antara lain:

- MySQL
- MsSQLServer
- PostgreSQL
- Oracle DB

b. Basis Data Hierarki atau *Hierarchical Database Management System* (HDBMS)

Merupakan DBMS yang pengorganisasian datanya berbentuk hierarki seperti pohon. Data tersimpan dari atas ke bawah dan memiliki hubungan *parents – child*. Induk (*parent*) berada pada bagian atasnya sedangkan anak (*child*) berada pada bagian bawahnya. Perhatikan gambar 4.6 berikut.



Gambar 4.6 Contoh *hierarchical* DBMS.
Sumber: Ariffudin/2022

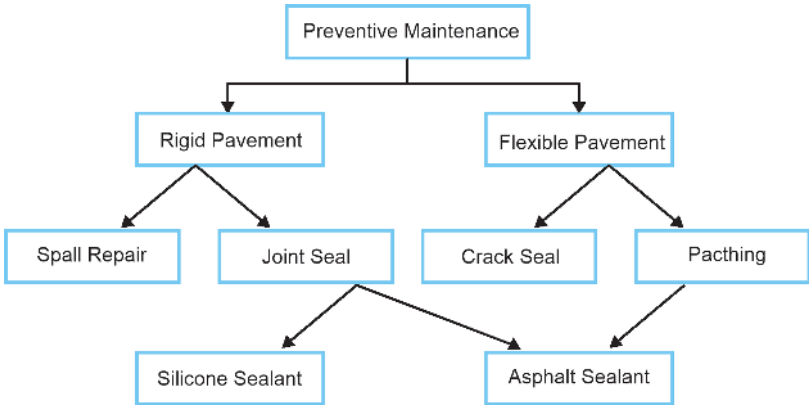
Contoh hierarchical DBMS antara lain:

- IBM Information Management System (IMS)
- RDM Mobile

- Windows Registry
- XML & XAML

c. Basis Data Jaringan atau *Network Database Management System* (NetDBMS)

Merupakan DBMS yang datanya disusun membentuk suatu jaringan. Secara teknis, NetDBMS merupakan bentuk turunan dari *hierarchical* DBMS. Sama halnya dengan *hierarchical* DBMS, pemodelan data di NetDBMS juga bersifat atas-bawah. Namun, jika pada *hierarchical* DBMS anak hanya berasal dari satu induk, pada NetDBMS, anak bisa berasal dari beberapa induk. Perhatikan gambar 4.7 berikut.



Gambar 4.7 Contoh *network* DBMS.
Sumber: Ariffudin/2022

Contoh network database antara lain:

- *Integrated Data Store* (IDS)
- *Integrated Database Management System* (IDMS)
- *Raima Database Manager*
- TurboIMAGE
- Univac DMS-1100

DBMS memiliki bahasa dan antarmuka tersendiri. Bahasa dan antarmuka yang disediakan berbeda untuk setiap kategori pengguna. Setiap DBMS mempunyai aturan tersendiri baik dari sintak maupun notasi pada penggunaan bahasanya. Namun, secara umum DBMS mengimplementasikan bahasa yang seragam yang disebut *Structured Query Language* (SQL) terutama pada RDBMS. SQL secara kegunaannya dikelompokkan menjadi:

a. *Data Definition Languange* (DDL). Merupakan kelompok perintah yang digunakan membangun kerangka sebuah basis data meliputi pendefinisian dan modifikasi atribut-atribut pada basis data, tabel, hubungan antar tabel, serta constraints. Yang termasuk dalam kelompok ini adalah *CREATE*, *ALTER*, *DROP*.

b. *Data Manipulation Language* (DML). Kelompok perintah yang digunakan untuk memanipulasi data-data yang ada dalam basis data termasuk di dalamnya menambahkan data baru, menyisipkan, memperbarui, menghapus, memanggil data, dan menyaring data tertentu. Yang termasuk dalam kelompok ini adalah *INSERT*, *SELECT*, *UPDATE*, *DELETE*.

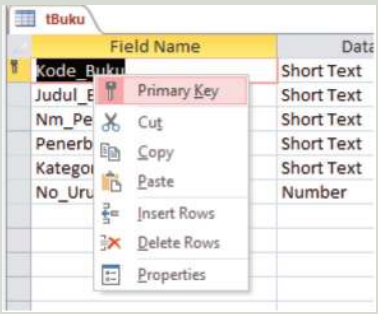
c. *Data Control Language* (DCL). Kelompok perintah yang digunakan untuk pengendalian akses ke server maupun data seperti pemberian akses dan *privillege* pengguna. Yang termasuk dalam kelompok ini adalah *GRANT*, *REVOKE*.

Aktivitas Belajar 4-1

1. Gunakan Microsoft Access dan buat sebuah basis data dengan nama Pustaka.accdb
2. Buat sebuah tabel dengan nama TBuku dengan struktur sebagai berikut.

Nama Field	Tipe Data	Ukuran
Kode_Buku	*Short Text*	8
Judul_Buku	*Short Text*	255
Nm_Pengarang	*Short Text*	50
Penerbit	*Short Text*	30
ID_Kategori	*Short Text*	3
No_Urut	*Number*	*Integer*

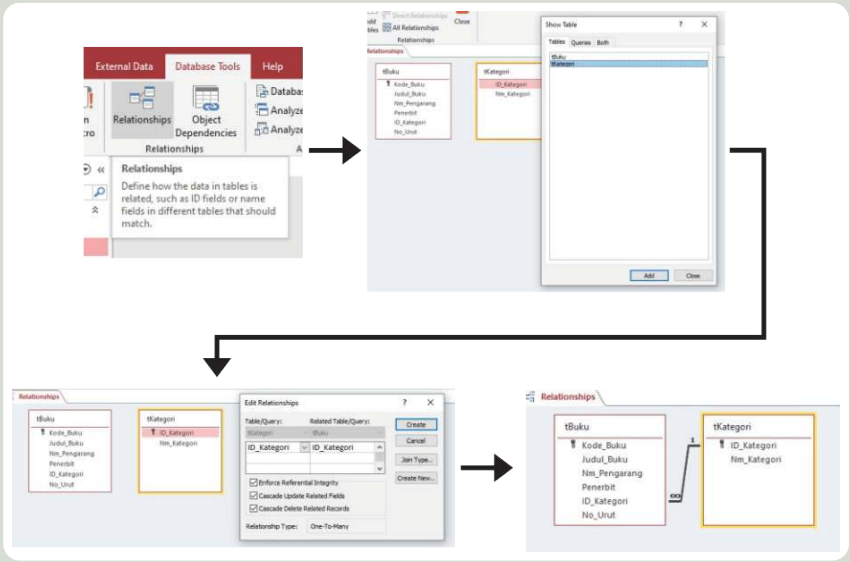
3. Atur Kode_Buku sebagai Primary Key



4. Tambahkan tabel baru dengan nama TKategori dengan struktur sebagai berikut.

Nama Field	Tipe Data	Ukuran	Keterangan
ID_Kategori	*Short Text*	3	*Primary Key*
Nm_Kategori	*Short Text*	30	

5. Buat relasi antar kedua tabel tersebut.



6. Isikan data berikut ke dalam tabel TBuku.

TBuku					
Kode Buku	Judul Buku	Nama Pengarang	Penerbit	ID Kategori	No Urut
APL.0001	ANALISIS STATISTIK SPSS 10.0	WIJAYA	ALFA BETA	APL	1
APL.0002	SPSS VERSI 12.00 DALAM RISET PEMASARAN	SRI RAHAYU	ALFA BETA	APL	2
APL.0003	DBASE IV CONTOH SUATU PROGRAM	EFFENDI, BE	PENERBIT ANDI YOGYAKARTA	APL	3
COM.0001	SISTEM MANAJEMEN BASISDATA	BAMBANG HARIYANTO	PENERBIT INFORMATIKA	COM	1
COM.0002	TEORI BAHASA, OTOMATA, KOMPUTASI	BAMBANG HARIYAN	PENERBIT INFORMATIKA	COM	2
COM.0003	SKM: GOAL PROGRAMING D/LINDO	SISWANTO	PT. ELEX MEDIA KOMPUTINDO	COM	3

TBuku					
Kode Buku	Judul Buku	Nama Pengarang	Penerbit	ID Kategori	No Urut
NET.0001	CARA MUDAH MENGELOLA BANYAK PC DALAM JARINGAN	FIRRAR UTDIRARTATMO	PENERBIT ANDI YOGYAKARTA	NET	1
NET.0002	DHCP PANDUAN KONF. JAR. TCP/IP	BERRY KERCHEVAL	PENERBIT ANDI YOGYAKARTA	NET	2
OOP.0001	OBJECT ORIENTED PROGRAMMING DENGAN PHP5	M. FARID AZIS	PT. ELEX MEDIA KOMPUTINDO	OOP	1
PRG.0001	PEMROGRAMAN CLIENT SERVER DENGAN KONEKSI MS. VISUAL FOXPRO &	IRWAN I. LAMANY	ADIPURA	PRG	1
PRG.0002	ALGORITMA & STRUKTUR DATA PRO	NIKLAUS WIRTH	PENERBIT ANDI YOGYAKARTA	PRG	2
WEB.0001	ASP PROGRAMMING	GREGORIUS AGUNG	ADIPURA	WEB	1
WEB.0002	WAP PROGRAMMING	GREGORIUS AGUNG	ADIPURA	WEB	2
WEB.0003	15 PROGRAM BANTU POPULER WEB	WAHANA	PENERBIT ANDI YOGYAKARTA	WEB	3

Apa yang terjadi pada saat kalian memasukkan data tersebut?

7. Sekarang, isikan data berikut ke tabel TKategori.

TKategori	
ID_Kategori	Nm_Kategori
APL	Aplikasi
MNJ	Manajemen
WEB	Web
PRG	Pemrograman
NET	Jaringan
COM	Komputer Umum
OOP	Pemrograman Berorientasi Objek

8. Ulangi langkah 6, apakah sekarang data tersebut dapat diisikan ke dalam tabel?
9. Diskusikan dengan teman, mengapa itu bisa terjadi.

B. Tools Pengembangan Perangkat Lunak

Mengembangkan perangkat lunak atau gim perlu kecermatan dan kehati- hatian dalam setiap tahapannya. Meskipun sudah mengikuti dengan baik langkah demi langkah pengembangan, tetapi ada kalanya bagian-bagian tertentu tidak sinkron dengan bagian lainnya atau malah terlewatkan. Hal ini tentu dapat mengurangi kualitas perangkat lunak atau gim yang dihasilkan.

Pada tahun 1980-an, mulai dikembangkanlah alat bantu sebagai tanggapan terhadap kebutuhan untuk menertibkan proyek pengembangan perangkat lunak besar, dan vendor mengklaim mereka akan meningkatkan produktivitas dan mengurangi kesalahan. Alat bantu inilah yang kemudian dikenal dengan istilah *Computer-Aided Software Engineering* (CASE).

CASE merupakan implementasi alat dan metode berbasis komputer dalam pengembangan perangkat lunak. CASE digunakan untuk memastikan perangkat lunak berkualitas tinggi dan bebas cacat. Pendekatan yang dianut CASE mencakup keseluruhan proses pengembangan perangkat lunak termasuk pembuatan kode program, repositori, pembuatan prototipe, pemodelan dalam diagram dan alat bantu lainnya yang diperlukan dalam pengembangan perangkat lunak. CASE mampu menyelesaikan proses pemodelan analisis, pemrograman, dan pengembangan sistem secara otomatis serta terarah. CASE juga dapat berfungsi sebagai penyimpanan (*store*) untuk berbagai dokumen seputar proyek pengembangan perangkat lunak seperti rencana bisnis, rencana pengembangan, kebutuhan, perancangan, serta pengujian.

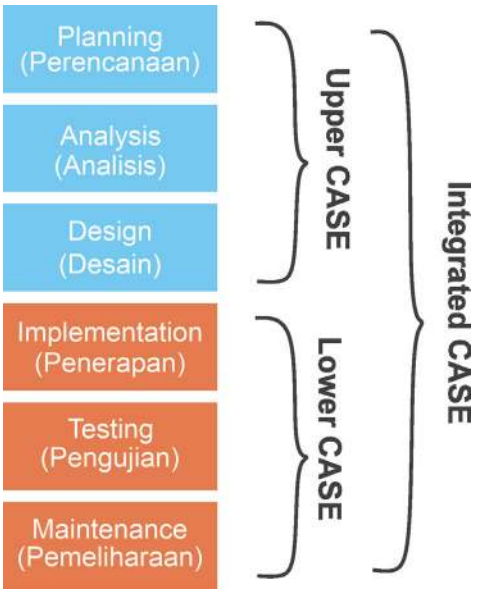
Secara umum, penggunaan CASE akan memberikan keunggulan kompetitif bagi kalian. Namun ada beberapa keuntungan lain yang kalian dapatkan juga ketika menggunakan CASE:

- Peningkatan produktivitas. Dengan CASE, kalian dapat meng- *generate* kode program secara otomatis berdasarkan desain yang sudah dibuat.
- Peningkatan kualitas program. Kode program yang dihasilkan lebih bebas *bug*. Proses otomatisasi generate kode program yang ada pada CASE menerapkan prinsip-prinsip standar yang seharusnya digunakan pada pemrograman sehingga kecil kemungkinan terjadi kesalahan.
- Peningkatan prosedur pengendalian. Pengendalian sistem, ukuran keamanan, keterauditan sistem, dan prosedur penanganan kesalahan dapat dideteksi lebih dini dalam proses desain.

- Dokumentasi yang disederhanakan. CASE dapat secara otomatis mendokumentasikan sistem dengan kemajuan perkembangannya, sehingga tidak perlu lagi harus membuat dokumentasi tersendiri.
- Penghematan biaya. Waktu yang lebih cepat, sumber daya lebih sedikit, kualitas lebih baik tentu akan menghemat biaya.

CASE tidak berdiri sendiri, tetapi didukung oleh berbagai alat bantu yang dinamakan *CASE Tools*. *CASE Tools* berdasarkan fase penggunaannya dikelompokkan menjadi:

- *Upper CASE Tools* – digunakan pada tahap perencanaan, analisis, dan desain.
- *Lower CASE Tools* – digunakan pada tahap implementasi, pengujian, dan pemeliharaan.
- *Integrated CASE Tools* – digunakan pada setiap tahap pengembangan.



Gambar 4.8 Pengelompokan *CASE Tools*.
Sumber: tutorialspoint.com/2022

Ada banyak *CASE Tools* yang bisa kalian gunakan dalam pengembangan perangkat lunak dan gim meliputi:

- **Diagram Tools**

  Alat bantu ini digunakan untuk membantu menyusun diagram-diagram yang diperlukan pada analisis, perancangan, *deployment*, maupun fase-fase lainnya. Contoh alat bantu ini antara lain: MsVisio, StarUML, VisualParadigm, Rational Rose, Sybase Power Designer, dan lain-lain.

- **Alat Bantu Pemodelan Proses**

  Alat bantu ini digunakan untuk membantu memodelkan proses pada perangkat lunak berdasarkan kebutuhan. Contoh alat bantu ini, EPF Composer.

- **Alat Bantu Manajemen Proyek**

  Alat bantu ini digunakan untuk membantu perencanaan proyek, estimasi biaya dan usaha, penjadwalan proyek dan perencanaan sumber daya. Alat bantu manajemen proyek membantu dalam menyimpan dan berbagi informasi proyek secara real-time di seluruh organisasi. Contoh alat bantu ini antara lain: MsProject, Creative Pro Office, Trac Project, dan Basecamp.

- **Alat Bantu Dokumentasi**

  Alat bantu ini dapat membantu menghasilkan dokumentasi baik teknis maupun untuk kebutuhan pengguna. Contoh alat bantu ini antara lain: Doxygen, DrExplain, dan Adobe RoboHelp.

- **Alat Bantu Analisis**

  Alat bantu ini digunakan untuk membantu memeriksa konsistensi kebutuhan, ketidakakuratan dalam diagram, redudansi data, atau kelalaian. Contoh alat bantu ini antara lain: Accept 360, Accompa, CaseComplete, dan Visible Analyst.

- **Alat Bantu Desain**

  Alat bantu ini membantu untuk merancang struktur blok pada perangkat lunak, yang selanjutnya dapat dipecah menjadi modul yang lebih kecil dengan menggunakan teknik penyempurnaan. Alat-alat ini memberikan perincian dari setiap modul dan interkoneksi antar modul. Contoh alat bantu ini antara lain: Animated Software Design.

- **Alat Bantu Manajemen Konfigurasi**

  Alat bantu ini membantu pelacakan perkembangan perangkat lunak secara otomatis termasuk manajemen versi serta manajemen rilis. Contoh alat bantu ini antara lain: Fossil, Git, dan Accu REV.

- **Alat Bantu Pengendali Perubahan**

  Alat bantu ini kadang juga dianggap bagian dari alat bantu manajemen konfigurasi. Alat bantu ini mengotomatisasi pelacakan perubahan pada kode program, manajemen *file*, manajemen kode.

- **Alat Bantu Pemrograman**

  Alat bantu ini memudahkan pemrogram untuk melakukan pemrograman secara konsisten dan lebih cepat. Umumnya berbentuk *Integrated Development Environment* (IDE) yang sudah berisi *library* bawaan, *emulator*, *debugging*, serta fitur-fitur lainnya. Contoh alat bantu ini antara lain: Netbeans, Eclipse, CScope, dan lain-lain.

- **Alat Bantu *Prototipe***

  Alat bantu ini dapat mempermudah pengembang untuk membuat *prototipe* (purwa-rupa) yang merupakan versi simulasi dari perangkat lunak. Contoh alat bantu ini antara lain: Serena dan Mockup Builder.

- **Alat Bantu Pengembangan Web**

  Alat bantu ini digunakan untuk pengembangan web yang terintegrasi sehingga baik dari sisi tampilan (*form*, teks, grafik, skrip) maupun komponen pendukung di dalamnya. Alat bantu ini juga dapat berbentuk *framework* serta menyediakan pratinjau laman web yang dibuat. Contoh alat bantu ini antara lain: Fontello, Adobe Edge Inspect, Foundation 3, Brackets, Code Ignition, Yi, Laravel, dan lain-lain.

- **Alat Bantu Penjaminan Kualitas**

  Alat bantu ini memudahkan seorang *Quality Manager* dalam melakukan penjaminan kualitas. Alat bantu ini terdiri dari konfigurasi, alat control perubahan, alat pengujian. Contoh alat bantu ini antara lain: SoapTest, AppsWatch, dan JMeter.

- **Alat Bantu Pemeliharaan**

  Alat bantu ini membantu dalam proses pemeliharaan perangkat lunak. Teknik pencatatan dan pelaporan kesalahan secara otomatis, menerbitkan tiket penanganan otomatis, serta analisis penyebab suatu kesalahan termasuk di dalamnya. Contoh alat bantu ini antara lain: Bugzilla dan HP Quality Center.

Selain banyak keuntungan yang didapat, penggunaan *CASE Tools* juga berpotensi membawa kerugian jika tidak dilakukan dengan tepat. Beberapa kerugian yang mungkin dapat timbul antara lain:

- Tidak kompatibel
- Biaya tinggi
- Harapan yang tidak terpenuhi/tidak sesuai ekspektasi

Agar tidak membawa kerugian, maka Kalian harus memperhatikan beberapa langkah berikut sebelum menggunakan *CASE Tools*.

1. Lakukan studi terhadap teknologi yang ada agar kalian bisa mempersiapkan dampak perubahan teknologi yang akan terjadi nantinya, sehingga model yang dibangun bisa fleksibel terhadap perubahan.
2. Evaluasi bagaimana jika organisasi yang sudah ada harus dibangun ulang agar bisa mengambil keuntungan dari teknologi baru.
3. Tetapkan suatu ketentuan untuk mengganti sistem yang lama dengan teknologi baru yang paling efektif.
4. Tentukan suatu metodologi pembangunan sistem.

Setelah melakukan tahapan-tahapan tersebut, barulah kalian bisa menentukan *CASE Tools* mana yang akan digunakan.

Aktivitas Belajar 4-2

Lakukan secara berkelompok, cari tahu sedikitnya 5 *CASE Tools* kemudian buat sebuah perbandingan sederhana mengikuti format tabel berikut.

No.	Nama *Case Tools*	Pembuat	Fase Pengembangan	Fitur-fitur	Kelebihan	Kekurangan
1						
2						
3						
4						
5						
...	...					

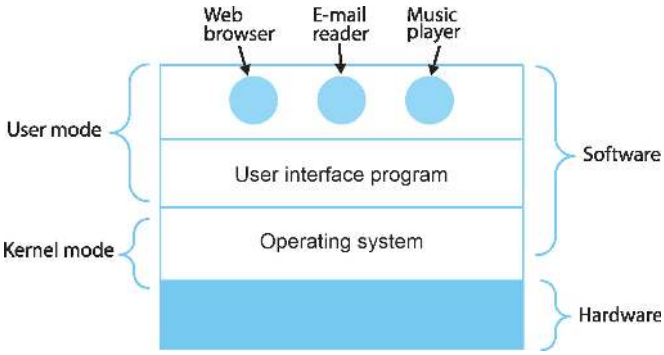
C. Ragam Sistem Operasi

Pernahkah kalian membayangkan bagaimana komputer bekerja tanpa sistem operasi? Sebelum adanya sistem operasi, untuk mengoperasikan komputer hanya digunakan sinyal analog maupun digital yang menyebabkan kerumitan dalam pengoperasiannya. Sistem operasi merupakan sebuah penghubung antara pengguna, perangkat lunak, dan perangkat keras komputer. Tugas utama sistem operasi adalah menyediakan program bagi pengguna dengan model komputer yang lebih baik, lebih sederhana, lebih bersih, dan menangani pengelolaan semua sumber daya (Tanenbaum & Bos/2015).

Seiring dengan berkembangnya pengetahuan dan teknologi, pada saat ini terdapat berbagai sistem operasi dengan keunggulan masing-masing. Ada

sistem operasi yang berjalan pada *platform* komputer personal, *workstation*, server, piranti bergerak bahkan pada otomotif. Ada pula sistem operasi yang interaksinya menggunakan perintah baris program, ada juga yang sudah berbasis grafis. Untuk lebih memahami sistem operasi maka sebaiknya perlu diketahui terlebih dahulu beberapa konsep dasar serta ragam sistem operasi itu sendiri.

Secara umum posisi sistem operasi dalam sistem komputer digambarkan pada gambar 4.9 berikut.



Gambar 4.9 Posisi sistem operasi pada sistem komputer.
Sumber: Tanenbaum & Bos/2015

Perangkat keras berada pada lapisan paling bawah dan di atasnya adalah perangkat lunak. Secara umum, sebagian besar perangkat lunak pada sistem komputer berjalan dengan dua mode, yaitu mode pengguna dan mode *kernel*. Sistem operasi merupakan perangkat lunak yang esensial (mendasar) yang bekerja pada mode *kernel* sedangkan perangkat lunak yang lain berjalan pada mode pengguna. Sistem operasi memiliki akses ke semua perangkat keras dan dapat menjalankan instruksi apapun yang diberikan oleh mesin. Mode pengguna tidak diperbolehkan mengakses langsung ke perangkat keras, apalagi menjalankan perintah langsung yang berkenaan dengan piranti masukan dan keluaran.

Sistem operasi berbeda dengan program pengguna, selain posisinya pada sistem, kerumitan dan kompleksitasnya juga tinggi. Jumlah baris kode program juga tidak sedikit. Kode program utama yang menjadi jantung (*kernel*) pada Windows dan Linux misalkan, bisa mencapai 5 juta baris lebih. Bisa kalian bayangkan, seberapa tebal halamannya jika dicetak menjadi buku?

Sistem operasi terus berkembang seiring juga dengan perkembangan perangkat keras. Dulu, untuk menjalankan sistem operasi harus mengetikkan serangkaian baris perintah karena antarmukanya masih berbasis teks (*shell*). Saat ini hampir semua sistem operasi sudah bekerja dengan antarmuka GUI yang lebih menarik dan mudah dioperasikan.

Berikut akan dibahas beberapa ragam sistem operasi yang berkembang saat ini.

1. Microsoft Windows

Microsoft Windows atau biasa disebut Windows merupakan sistem operasi. Dikembangkan oleh Microsoft Corporation sejak 1985 menggunakan antarmuka berbasis GUI dan merupakan kelanjutan dari MsDOS (*Microsoft Disk Operating System*) yang berbasis teks.

Gambar 4.10 Logo Windows dari masa ke masa.
Sumber: dailysocial.id/Arnetta

Sampai saat buku ini ditulis, ada banyak versi Windows yang sudah diluncurkan. Versi terbaru dinamakan Windows 11 yang dirilis pada tahun 2021. Sebelumnya juga ada Windows 10 (dirilis tahun 2015), Windows 8 (2012), Windows 7 (2009), Windows Vista (2007), dan versi-versi sebelumnya. Selain untuk kebutuhan personal komputer, Windows juga diluncurkan untuk memenuhi kebutuhan server (Windows Server) berbasis *kernel* Windows NT, piranti bergerak (Windows Mobile), bahkan sampai ke bidang otomotif dengan Windows Automotive.

2. Unix

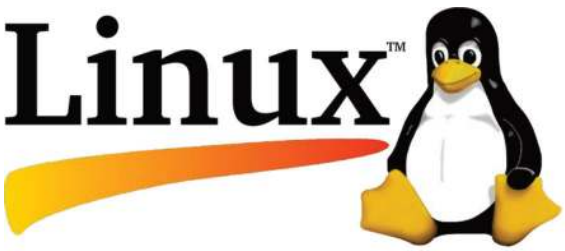
Unix adalah sistem operasi komputer *multiuser* dan *multitasking*, yakni memungkinkan pengguna untuk melakukan penggunaan aplikasi atau program dalam waktu yang bersamaan. Unix diciptakan oleh Ken Thompson dan Dennis Ritchie, dikembangkan oleh AT&T Bell Labs. Unix mengkhususkan diri pada *workstation* dan server. Sistem Unix dibangun di sekitar *kernel* inti yang mengelola sistem dan proses lainnya. Subsistem *kernel* dapat mencakup manajemen proses, manajemen file, manajemen memori, manajemen jaringan, dan lain-lain.

Gambar 4.11 Logo sistem operasi Unix
Sumber: jurnalponsel.com

Unix dibangun dengan Bahasa C dan menggunakan antarmuka berbasis teks. Unix merupakan sistem operasi yang memiliki tingkat keamanan tinggi. Setiap *file*, direktori, pengguna, dan grup diatur sedemikian rupa sehingga memiliki pengaturan izin akses tersendiri. Unix dapat digunakan, disalin, dan dimodifikasi berdasarkan kebutuhan yang berbeda oleh banyak pihak. Hal inilah yang menyebabkan Unix memiliki banyak sekali varian. Salah satu varian Unix yang berkembang pesat adalah Linux.

3. Linux

Linux pada awalnya dikembangkan oleh Linus Torvalds yang merupakan kloningan dari MINIX (salah satu varian UNIX) dan peralatan sistem serta *library*-nya secara umum berasal dari Sistem Operasi GNU (*GNU's Not UNIX*). Linux memiliki banyak desain yang berasal dari desain dasar Unix, Linux menggunakan Kernel Monilitik yaitu *kernel* yang menangani kontrol proses, jaringan, periferal, dan pengaksesan sistem berkas. Sama seperti Unix, Linux pun dapat dikendalikan oleh satu atau lebih antarmuka baris perintah (*Command Line Interface*/CLI) berbasis teks, antarmuka pengguna grafis (*Graphical User Interface*/GUI) seperti GNOME, KDE, dan Xfce untuk versi *desktop*.



Gambar 4.12 Logo Linux
Sumber: freepnglogos.com

Linux merupakan sistem operasi yang dibuat dalam bentuk sumber terbuka (*open source*). Artinya siapapun dapat memodifikasi kode program pada Linux untuk disesuaikan kebutuhannya serta mendistribusikannya. Keadaan inilah yang menjadikan Linux tersedia dalam banyak versi *open source* (distro) yang berbeda serta dapat diunduh secara gratis. Ada beberapa macam distro Linux, seperti: Debian, Lycoris, Xandros, Lindows, Linare, Linux-Mandrake, Red Hat Linux, Slackware, Knoppix, Fedora, Suse, Ubuntu.

Jika kalian merupakan orang yang mengetahui cara menyesuaikan dan bekerja dengan sistem operasi, serta pengkodean dan pekerjaan back- end semacam ini menarik, sistem ini pilihan yang ideal untuk kalian. Kalian dapat menggunakan sistem operasi ini dan mulai memodifikasinya. Selain itu, sistem ini terkenal kebal akan virus, tetapi untuk perangkat lunak tambahan masih rentan terhadap *malware* pada web server. Bahasa yang kerap digunakan dalam pengembangan Linux merupakan C dan Assembly.

4. MacOS

MacOS atau *Macintosh Operating System* merupakan sistem operasi yang dikembangkan oleh Apple Computer khusus untuk komputer Apple dan tidak kompatibel dengan komputer berbasis IBM PC. MacOS adalah sistem operasi pertama yang memiliki antarmuka berbasis GUI. Sistem operasi ini banyak disukai karena kemudahan pengoperasian, kecepatan pemrosesan, serta pembaruan akan *user experience* yang cepat. MacOS identik dengan kebutuhan multimedia karena tampilannya yang menarik. Selain itu MacOS jarang ditemukan bug dan memiliki keamanan vang baik.

Gambar 4.13 Logo MacOS
Sumber: pngegg.com

MacOS dibagi menjadi 2 jenis:

a. MacOS Klasik

Tidak memiliki sembarang *Command Line* (baris perintah), menggunakan *User Interface* (UI) sepenuhnya dan menggunakan *Cooperative Multitasking*.

b. MacOS X

MacOS X adalah penerus dari MacOS (Klasik). MacOS X dibangun di atas XNU *kernel*, dengan fasilitas standar Unix tersedia dari

antarmuka baris perintah. MacOS X memasukkan unsur-unsur BSD Unix, One Step, dan MacOS X memiliki memori ala-Unix dan Pre-Emptive Multitasking. Kelebihan MacOS X:

- Stabil, karena menggunakan UNIX.
- Multitasking
- Tampilan (UI) sangat bagus.
- Aman dari *malware*.

5. Chrome OS

Chrome OS merupakan sistem operasi yang dikembangkan oleh Google yang diperkenalkan pada tahun 2009 dan dirilis resmi pada tahun 2011. Sistem ini mengandalkan antarmuka berbasis peramban Google Chrome dengan basis Linux Gentoo sebagai sistem utamanya. Sistem operasi ini tidak dapat berjalan di sembarang perangkat, hanya bisa digunakan pada perangkat mitra manufaktur Google dalam bentuk Chromebook.

Gambar 4.14 Logo Chrome OS
Sumber: pngwing.com

Chrome OS dirancang untuk pengguna yang memang banyak menghabiskan waktu dengan internet. Selain peramban Chrome yang disertakan, Chrome OS hanya menambahkan pemutar multimedia dan manajemen file. Inilah salah satu hal yang membuat sistem operasi ini menjadi lebih ringan. Aplikasi pendukung kerja lainnya dapat kalian tambahkan melalui Chrome Web Store.

Chrome OS juga terhubung dengan akun Google sehingga dapat memanfaatkan aplikasi yang ada di Google Workspace seperti Google Docs, Google Sheet, Google Presentation, termasuk penyimpanan awan pada Google Drive. Tidak semua aplikasi pada Chrome OS harus dijalankan secara daring, ada beberapa aplikasi juga yang dapat dijalankan secara luring. Karena Chrome OS dan Android merupakan produk Google, sebagian besar aplikasi yang ada Android juga dapat dijalankan di sistem ini.

Chrome OS dirancang untuk kesederhanaan, kecepatan, dan keamanan yang baik, oleh karena itu *firmware* OS ini dibuat seringan mungkin dengan menghilangkan beberapa fitur perangkat keras seperti yang selama ini ditemukan pada BIOS. Proses *booting* menjadi jauh lebih cepat dibandingkan dengan sistem operasi lainnya. Chrome OS juga mengadopsi *sandbox* yang merupakan salah satu teknik keamanan dengan mengisolasi program, mencegah program jahat atau penyerang mengintip, menguasai dan merusak seluruh komputer. Sistem memeriksa integritas sistem operasi pada saat boot dan memperbaiki dirinya sendiri jika mendeteksi adanya modifikasi pada *file* sistem. Firmware Chrome OS juga akan mendeteksi jika OS hilang atau rusak saat memulai, dan meminta pengguna untuk menjalankan proses pemulihan.

Chrome OS adalah *open source* dan gratis untuk digunakan. Selain itu tidak memerlukan perangkat keras kelas atas seperti media penyimpanan yang kapasitasnya besar mengingat penyimpanan dilakukan secara awan pada akun Google Drive.

Seiring dengan perkembangan piranti bergerak, terutama telepon pintar, maka sistem operasi juga memiliki peran yang tidak sedikit. Jika sebelumnya Kalian sudah mengenal beberapa sistem operasi yang berjalan pada piranti komputer persomal, berikut dibahas sistem operasi yang berjalan pada telepon pintar.

Gambar 4.15 Logo iOS dari masa ke masa

Sumber: 1000logos.net

6. iOS

iOS merupakan sistem operasi untuk seluler yang dibangun oleh Apple Inc. berdasarkan MacOS X. iOS pada dasarnya adalah akronim dari iPhone Operating System. iOS diluncurkan pada 2007 awalnya hanya untuk iPhone, tetapi saat ini telah merambah ke piranti Apple lainnya seperti iPad, iPod Touch. iOS merupakan sistem operasi tertutup yang hanya dapat digunakan pada piranti produk Apple.

Antarmuka iOS dibangun berdasarkan prinsip *direct manipulation* (objek yang menarik, respon yang cepat, serta umpan balik ke pengguna) dan multitouch (lebih dari satu titik). iOS juga menyediakan aksesabilitas yang baik sehingga memudahkan pengguna dengan kekurangan fungsi penglihatan maupun pendengaran dapat menggunakan iOS dengan

baik. Aplikasi yang dapat dijalankan pada sistem operasi ini dapat kalian unduh di Apple App Store.

7. Android

Gambar 4.16 Logo sistem operasi Android.
Sumber: freepnglogos.com

Android merupakan sistem operasi berbasis Linux yang digunakan untuk seluler terutama telepon pintar dan tablet. Android pada awalnya dikembangkan oleh Andorid Inc. berbasis Palo Alto pada tahun 2003 yang kemudian diakuisisi oleh Google pada tahun 2005. Penamaan sistem operasi ini terbilang unik karena mengikuti tema hidangan penutup seperti Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, Kitkat, Lollipop, Marshmallow, dan lain-lain. Google beralasan tema ini digunakan "Karena perangkat ini membuat hidup kita begitu manis, setiap versi Android diberi nama makanan penutup.

Awalnya Android dikembangkan untuk sistem operasi pada kamera digital yang kemudian berkembang luas untuk berbagai piranti. Android saat ini tidak hanya digunakan pada telepon pintar, tetapi juga sudah digunakan pada jam tangan, TV, pemutar lagu, *head* unit mobil, sampai dengan piranti elektronik kebutuhan rumah tangga. Android memungkinkan pengguna menggunakan aplikasi pihak ketiga yang diunduh dari web maupun layanan resmi Google Play Store. Kalian pun dapat membangun perangkat lunak atau gim yang diunggah ke Play Store agar para pengguna dapat menggunakannya dengan mudah. Android didukung oleh komunitas besar pengembang dan pengguna, termasuk ketika pengguna ingin memeriksa keaslian sebuah aplikasi. Pengguna dapat memeriksa ulasan yang diberikan oleh pengguna yang ada di Play Store.

Sama halnya dengan sistem operasi Google lainnya, Android juga terintegrasi dengan layanan Google termasuk penyimpanan awan dan lainnya. Meskipun setiap piranti telah dilengkapi dengan media penyimpanan, pengguna juga dapat menyimpannya pada penyimpanan awan Google Drive.

Selain beberapa sistem operasi yang telah dibahas sebelumnya, masih banyak sistem operasi lain yang pernah ada maupun masih

berjalan seperti MS DOS, IBM OS/360, IBM OS/2, Palm OS, Blackberry OS, Symbian OS, Windows Phone, Firefox OS, MeeGo, WebOS, BADA, dan masih banyak lagi.

Aktivitas Belajar 4-3

Lakukan secara berkelompok, cari tahu dan diskusikan sedikitnya 5 sistem operasi selain yang telah dibahas sebelumnya. Buat ringkasan dari kelima sistem operasi tersebut yang berisi:

1. Sejarah
2. Teknologi dasar
3. Piranti yang didukung
4. Kelebihan
5. Kekurangan

Presentasikan ringkasan dan hasil diskusi kalian!

D. Pengelolaan Aset

Dalam mengembangkan gim, ada banyak karakter yang harus dibuat. Karakter-karakter tersebut ada yang digunakan berulang-ulang. Apakah setiap karakter yang akan digunakan harus dibuat ulang? Bukan hanya karakter, dalam gim juga terdapat objek-objek yang menyertainya, termasuk suara, gambar, video, animasi, dan objek-objek lainnya. Objek, karakter, latar, suara, gerakan, animasi, tekstur, dan segala sesuatu yang digunakan pada gim itulah yang disebut dengan aset. Agar aset gim dapat digunakan dengan tepat dan tidak tumpang tindih, maka diperlukan suatu pengelolaan yang baik yang disebut dengan manajemen aset.

Proses menyiapkan, membuat, bahkan mengimpor, dan mengelola aset gim bisa jadi rumit, lama dan juga membosankan. Aset gim dapat dibuat secara manual menggunakan perangkat lunak desain grafis seperti Corel Draw, Adobe Photoshop, atau perangkat lunak pemodelan 3D seperti Maya, 3DS Max Design, Modo, dan Zbrush, perangkat lunak pengolah suara seperti Audacity, dan perangkat lunak pembuat animasi serta perangkat lainnya. Tentu pembuatan aset secara manual tidak secepat menggunakan perangkat lunak manajemen aset gim. Perangkat lunak ini membantu pengembang gim mengelola berbagai aset dalam gim dengan baik. Menggunakan perangkat lunak seperti ioMoVo, echo3D, Unity Asset Store, Unreal Engine 4, Game Maker Studio, dan lain-lain memungkinkan pembuatan purwa-rupa gim dengan cepat.

Selain hal di atas, ada berbagai alasan mengapa perlu menggunakan perangkat lunak manajemen aset gim. Alasan paling umum yang ada adalah (I/O Software Inc., 2022):

1. **Kecepatan:** Perangkat lunak manajemen aset dapat membantu kalian membuat dan mengelola aset dengan cepat. Ini penting jika kalian ingin membuat aset baru atau jika kalian perlu memperbarui aset yang ada dengan cepat.
2. **Kenyamanan:** Perangkat lunak manajemen aset dapat mempermudah pembuatan dan pengelolaan aset karena perangkat lunak menyediakan akses mudah ke semua aset kalian, serta fitur untuk membantu kalian mengatur aset.
3. **Pengorganisasian;** perangkat lunak manajemen aset gim akan membantu kalian mempermudah pengorganisaian. Kalian akan lebih mudah melakukan pelacakan aset serta lokasi. Ini dapat berguna saat kalian perlu menemukan di mana aset untuk gim disimpan dan digunakan atau saat kalian perlu membuat aset baru.

Selain alasan-alasan tersebut, perangkat lunak manajemen aset memiliki manfaat yang tidak sedikit. Perangkat lunak manajemen aset membantu pengembang gim merampingkan proses pembuatan dan pengelolaan aset. Piranti ini dapat membuat pengembang juga lebih mudah untuk bekerja dengan lebih banyak file dengan lebih sedikit kesalahan. Dengan demikian dapat mengurangi waktu yang dihabiskan untuk pembuatan dan pengelolaan aset, meningkatkan akurasi saat menyulap beberapa aset, dan memfasilitasi proses pelacakan, perubahan, dan pembaruan aset gim.

Berbagai pilihan perangkat lunak manajemen aset tersedia, tinggal disesuaikan dengan kebutuhan kalian. Beberapa pilihan perangkat lunak manamen aset gim populer antara lain GameMaker Studio 2's Asset Store, Unity's Assets Store, Unreal Engine 4's Marketplace, dan RPG Maker MV's File System. Pertimbangkan dengan seksama fitur setiap piranti yang tersedia. Beberapa fitur umum yang menjadi bahan pertimbangan mencakup opsi penyimpanan aset (seperti penyimpanan lokal atau *online*), fungsi pencadangan/pemulihan otomatis, alat kontrol versi, dan dukungan untuk mengimpor/mengekspor *file*.

Aktivitas Belajar 4-4

Dengan mengunakan salah satu perangkat lunak manajemen aset gim, buatlah aset-aset yang diperlukan untuk sebuah permainan (gim) *sutten* sebagaimana yang telah direncakan pada aktivitas belajar 3-3.

E. Antarmuka Pengguna (*User Interface*)

Pernahkah kalian menemukan situs, gim, atau aplikasi yang tampilannya kurang enak dilihat? Atau sulit sekali memahami alur kerjanya? Bahkan antara tulisan dan latar belakangnya seperti menjadi satu? Tentu membuat kalian tidak nyaman dan mungkin merasa sebal, kan? Nah, itulah salah satu akibat dari pengembang yang tidak memperhatikan antarmuka pengguna atau *user interface* (UI).

Antarmuka pengguna merupakan tampilan visual yang pertama kali bersentuhan langsung dengan pengguna. Ibarat bangunan, orang akan melihat tampilan depannya terlebih dahulu. Di situ akan terlihat di mana letak pintu masuk, bagaimana bentuk jendelanya, jalan menuju pintu mudah dilalui atau tidak, dan sebagainya. Jika semuanya enak dilihat, dan mudah dicapai, pasti orang akan tertarik untuk memasuki bangunan tersebut dan mencari tahu lebih dalam lagi apa yang tersimpan dalam bangunan tersebut.

Dalam perancangan antarmuka perangkat lunak, Deborah J Mayhew (Mayhew, 2008) mengemukakan beberapa prinsip yang harus diperhatikan oleh kalian di antaranya adalah sebagai berikut:

1. Kesesuaian dengan pengguna
2. Kesesuaian dengan produk-produk sebelumnya
3. Kesesuaian dengan tugas yang dilaksanakan *user*
4. Kesesuaian dengan alur pekerjaan
5. Kosistensi
6. Keakraban
7. Kesederhanaan
8. Manipulasi langsung
9. Kontrol oleh pengguna
10. WYSIWYG (*What You See is What You Get*)
11. Fleksibel
12. Responsif
13. Kerumitan teknologi tersembunyi
14. Tangguh
15. Proteksi dari kesalahan manusia
16. Mudah dipelajari dan mudah digunakan

Dalam penggunaannya, antarmuka pengguna terbagi menjadi beberapa jenis yaitu:

1. ***Graphical User Interface (GUI)***; Jenis ini merupakan yang paling banyak digunakan untuk aplikasi dan web. Biasanya mencakup tombol, menu, taskbar yang direpresentasikan pada panel digital dan diakses menggunakan pointer mouse. Contoh GUI yakni *taskbar*, menu *start*, *notification area* yang ada di sistem Windows.
2. ***Command Line Interface* (CLI)**; Merupakan antarmuka berbentuk baris perintah pengguna. Pengguna harus memasukkan serangkaian baris perintah agar suatu fitur dapat digunakan. Jenis antarmuka ini lebih cepat dibandingkan dengan antarmuka jenis lainnya. Contoh CLI adalah MS-DOS di Windows dan terminal di MAC atau Linux.
3. ***Menu-Driven User Interface***; merupakan antarmuka pengguna berbasis menu. Pengguna diberikan pilihan terbatas pada opsi yang ditampilkan. Pengguna tidak perlu mengetahui tentang sistem, cukup memahami apa tugas yang harus dilakukan. Contoh antarmuka ini adalah menu pada ATM, USSD pada telepon genggam.
4. ***Touch User Interface***; merupakan antarmuka berbasis sentuhan. Antarmuka ini tidak lepas dari teknologi perangkat keras *display* yang memungkinkan pengguna memilih dengan langsung menyentuh layar. Contoh antarmuka ini adalah tampilan pada telepon cerdas.
5. ***Voice User Interface (VUI)***; merupakan antarmuka pengguna berbasis suara. Dengan teknik pengenalan suara, pengguna dapat mengakses suatu aplikasi dengan memberikan perintah melalui suara. Pengguna dengan fungsi penglihatan terbatas atau fungsi gerak terbatas dapat menggunakan antarmuka ini. Contoh penggunaan *Voice User Interface*, yaitu perangkat asisten virtual, *talk-to-text*, dan GPS.
6. ***Form-Based User Interface***; merupakan antarmuka pengguna yang berbasis formulir. Biasanya digunakan untuk memasukkan data ke dalam aplikasi dengan menawarkan pilihan-pilihan yang terbatas. Contoh *Form-Based Interface* yang biasa ditemui oleh pengguna, yakni halaman formulir kontak bisnis yang ada di situs web. Bisa juga kamu temukan dari Google Formulir yang digunakan untuk mengumpulkan data.
7. ***Natural Language User Interface***; antarmuka ini memungkinkan pengguna berinteraksi menggunakan bahasa alami atau bahasa sehari-hari. Pengguna memberikan masukan dalam bahasa alami manusia melalui baik tulisan, suara, atau metode lain. Sistem akan memberikan tanggapan yang disampaikan melalui ucapan, teks, atau modalitas lain yang sesuai. Misalnya, Google Search, ChatGPT, chatBot.

Secara teknis, komponen dasar pembentuk antarmuka pengguna adalah:

1. Tata letak – penempatan elemen antarmuka yang digunakan.
2. Warna – pewarnaan dari desain antar muka secara umum baik *foreground* maupun *background.*
3. Tipografi – kombinasi huruf yang digunakan di antarmuka.
4. Grafik – ikon yang digunakan sebagai ilustrasi penggunaan sistem.

1. Tata Letak

Apa kesan yang kalian dapatkan ketika memasuki sebuah ruangan yang berantakan? Kesal, tidak nyaman, bahkan mungkin marah bisa kalian rasakan. Namun bagaimana juga kesannya jika kalian memasuki ruangan yang tertata dengan rapi, indah, nyaman, bersih? Tentu kalian akan betah berlama-lama di ruangan itu. Demikian juga dengan tata letak pada antarmuka perangkat lunak dan gim.

Tata letak merupakan struktur yang mendukung komponen visual pada sebuah tampilan antarmuka. Tata letak berfungsi membuka jalur agar penglihatan pengguna dapat dengan mudah beralih dari bagian ke bagian lainnya dan memahami informasi, termasuk menyorot informasi yang paling penting pada sebuah halaman. Desain tata letak yang baik tercermin dalam pengalaman pengguna. Kalian dapat memperhatikan beberapa hal berikut untuk mendesain tata letak (Moreno, 2020).

a. Ukuran

Ukuran tampilan menjadi hal pertama yang harus diperhatikan. Hierarki ukuran tampilan dapat memfasilitasi pengguna membuat keputusan untuk melanjutkan penelusuran informasi berikutnya atau tidak. Berdasarkan ukuran inilah pengembang atau desainer menentukan bagian mana yang merupakan informasi penting atau utama. Perhatikan juga kesesuaian dengan piranti yang digunakan. Bagian yang memiliki informasi utama tidak boleh menghilang pada saat berganti piranti.

b. Perjalanan Visual

Memandu tampilan yang disarankan jauh lebih ramah daripada memaksa item untuk ditelusuri. Menjadikan semua item menarik dalam sebuah tampilan bukanlah hal yang baik. Item-item semestinya didesain sesuai urutan prioritas. Kenali dengan

baik informasi apa yang hendak disampaikan, pahami isi, tujuan serta elemen apa yang akan dimasukkan ke dalam tampilan.

c. Asimetris

Tata letak tampilan tidak selalu harus simetris. Kalian perlu mempertimbangkan tata letak yang asimetris. Tata letak asimetris dapat mengekspresikan lebih banyak gerakan dan dinamisme. Asimetri cenderung lebih menarik perhatian mata karena hukum komposisi dan keseimbangan, memfasilitasi hierarki elemen dan memberi mereka gerakan. Tata letak asimetris justru menyeimbangkan elemen antarmuka sedemikian rupa sehingga satu bagian tidak lebih berat dari yang lain, tetapi tanpa terlalu jelas seperti dalam desain simetri. Ini akan memberi ruang bagi Kalian untuk menentukan hierarki dalam tata letak dan bermain dengan elemen seperti teks, ikon, gambar, ruang, dan warna dengan cara yang lebih kreatif dan tetap.

d. Ruang

Spasi atau ruang yang benar akan menentukan tata letak. Hal ini dikarenakan tata letak ditunjukkan oleh jarak antar elemen atau ruang negatif, bentuknya tergantung pada seberapa dekat beberapa elemen dari yang lain. Ruang-ruang ini berisi komposisi dan menghubungkan atau memutuskan elemen sedemikian rupa sehingga asosiasi tercipta di antara elemen-elemen tersebut. Asosiasi visual utama ditentukan oleh tata letak, sehingga harus dirancang untuk tujuan ini sejak awal. Kalian juga harus ingat bahwa jarak antara satu elemen dan elemen lainnya harus menjadi cerminan dari apa yang ada dalam pikiran pengguna sebagai terkait dan tidak terkait, dan mengisolasi sekelompok elemen memberikan lebih banyak keterbacaan dan kepentingan. Tata letak desain UI akhirnya merupakan organisasi ruang.

e. Patahkan

Keluar dari skema tradisional dapat menarik perhatian pengguna. Pengulangan bentuk cenderung membuat pikiran pengguna merasa jenuh sehingga setiap perubahan skema akan selalu menarik perhatian pengguna untuk mencobanya. Berfokuslah pada bagaimana menarik perhatian. Sering kali

mendesain tata letak fungsional dasar mengurangi rasa ingin tahu desainer untuk menemukan cara baru agar membaca informasi lebih menyenangkan. Namun menjaga perhatian pengguna dan membuat pengalaman menjadi menyenangkan sangat berkaitan dengan pergerakan, skala, kekusutan, dan pelanggaran beberapa aturan. Yang biasa memang nyaman, tetapi inovasinya jadi berkesan.

f. Superposisi

Tumpang tindih satu elemen dengan yang lain menciptakan kedalaman. Meskipun tata letaknya hidup dalam dunia dua dimensi, menempatkan satu elemen di atas elemen lainnya memperluas kedalamannya memberikan realisme pada komposisi. Menggunakan superposisi dalam desain UI menciptakan persepsi pengguna bahwa ada panggung nyata di layar. Penggunaan lapisan yang berbeda menunjukkan bahwa desain berlanjut, meluas ke ruang, yang tidak hanya terasa lebih realistis dan menyerap tetapi juga lebih menarik. Salah satu aturan terpenting saat teks tumpang tindih adalah harus ada cukup kontras dengan latar belakang untuk menghindari masalah keterbacaan. Jika penglihatan tidak dapat mengenali karakter dengan lancar, pengalaman itu akan membuat stres.

g. *Grid*

Penggunaan *grid* secara matematis enak dipandang. Meskipun tidak terlihat dalam desain, grid memudahkan penempatan elemen, menciptakan distribusi yang proporsional, dan memberikan konsistensi yang tinggi pada antarmuka. Mendesain dengan penspasian yang benar adalah tantangan nyata, tetapi mempertahankan konsistensi penspasian ini di seluruh aliran UI bahkan lebih menantang. *Grid* adalah solusi standar untuk ini. Caranya memuat elemen akan memberikan proporsi yang tepat, tetapi kita harus ingat bahwa tata letak dapat sangat bervariasi, bergantung pada gaya perancang dan bagaimana ia ingin menampilkan informasi. Praktek sistematis adalah cara terbaik untuk menguasai penggunaan *grid*.

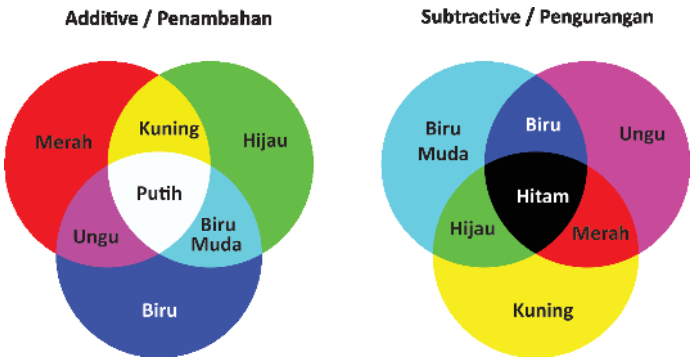
Meskipun banyak variasi tata letak antarmuka, yang paling penting adalah membuatnya tetap sederhana dan sesuai dengan tujuan antarmuka tersebut. Bukanlah tugas yang mudah untuk dapat

menggabungkan prinsip estetika dan ilmu proses kognitif, penggunaan yang tepat dari keduanya sangat penting untuk desain UI profesional. Jika Kalian akan mengerjakan proyek baru, ingatlah untuk memberikan diri Kalian kebebasan kreatif yang diperlukan, tetapi selalu berdasarkan pada struktur yang diakui pengguna, dan yang telah terbukti berhasil.

2. Warna

Warna merupakan elemen antarmuka yang tidak bisa lepas dari pengguna. Dalam kehidupan nyata, manusia mengenal warna primer yaitu Merah, Kuning, dan Biru. Warna-warna lain terbentuk berdasarkan perpaduan ketiga warna primer tadi. Namun berbeda dalam dunia digital. Warna primer digital tidaklah sama dengan warna primer dalam dunia nyata. Manusia dalam melihat warna pada dasarnya didasarkan pada prinsip penambahan (*Additive*) dan pengurangan (*Substractive*). Kedua prinsip inilah yang kemudian melahirkan sistem warna digital yang biasa dikenal seperti RGB dan CMYK.

Warna *additive*, biasanya digunakan untuk tampilan layar. Hal ini didasarkan pada penglihatan manusia yang dipengaruhi panjang gelombang warna tertentu yang didasarkan pada "*trichromatic theory*". Semua warna bisa dihasilkan dengan mencampur 3 (tiga) buah warna utama atau warna dasar. Yang termasuk ke dalam sistem ini adalah RGB, Lab Color, HLS. Gim, tampilan web, wallpaper, tampilan aplikasi, dan video menggunakan sistem ini.



Gambar 4.17 Sistem warna *additive* dan *subtractive.*
Sumber: Stephanus Eko Wahyudi/informatika.uc.ac.id

Warna *substractive* pada dasarnya merupakan perpaduan dari tiga warna dasar. Sistem ini biasanya digunakan untuk keperluan cetak. Konsep *substractive* ini didasarkan pada penglihatan manusia

yang jarang sekali melihat langsung pada sumber cahaya. Warna yang dilihat merupakan hasil pantulan cahaya dari benda. Jenis warna ini digunakan apabila akan dicetak pada media kertas atau media lainnya. Yang termasuk sistem ini adalah CMYK yang memiliki tiga warna dasar yaitu *cyan*, magenta, *yellow*.

3. Tipografi

Elemen lain yang erat dengan pengguna adalah penggunaan jenis huruf atau biasa dikenal dengan istilah tipografi. Tipografi merupakan suatu ilmu dalam memilih dan menata huruf dengan pengaturan penyebarannya pada ruang-ruang yang tersedia, untuk menciptakan kesan tertentu, sehingga dapat menolong pengguna untuk mendapatkan kenyamanan membaca semaksimal mungkin.Tipografi dapat dikatakan sebagai bahasa visual karena merupakan media berkomunikasi antara pengguna dan aplikasi atau gim. Tipografi sendiri merupakan seni mendesain dengan tulisan. Secara luas cakupannya tidak hanya huruf tetapi juga mencakup penataan dan pola halaman tampilan, penataan dan berbagai hal bertalian pengaturan baris-baris susunan huruf (*typeset*).

Setiap jenis huruf memiliki karakter yang berbeda sehingga perlu pengelolaan dan penanganan yang berbeda. Dalam penggunaannya, elemen tipografi terbagi menjadi dua yaitu:

a. Huruf Teks (*Text Type*), yaitu huruf yang tersaji untuk naskah. Jenis huruf teks merupakan huruf dengan *body* yang tidak terlalu tebal dan tidak terlalu banyak lekukan agar tingkat keterbacaannya tinggi dan nyaman.
b. Huruf Judul (*Display Type*), yaitu huruf yang digunakan untuk penekanan judul suatu halaman atau tampilan. Penggunaan huruf lebih fleksibel, tidak kaku yang penting unsur keterbacaan dan keefektifan penyampaian pesan dapat terkemas dengan rapi dan nyaman, maka unsur penerapan dalam desain grafis telah terpenuhi. Penggunaan huruf juga tergantung pada tema desain antarmuka.

Pada sistem komputer saat ini terdapat berbagai jenis huruf, dari mulai yang klasik sampai yang modern. Kalian dapat menerapkan penggunaan berbagai macam huruf dalam mengembangkan perangkat lunak atau gim. Hal yang harus diperhatikan pada penggunaan huruf ada pada keterbacaan (*readability*) dan kejelasan (*legibility*). Keterbacaan

merupakan tingkat kenyamanan atau kemudahan suatu susunan huruf saat dibaca oleh pengguna. Keterbacaan biasanya dipengaruhi oleh jenis huruf, ukuran, kontras warna belakang, pengaturan (termasuk spasi, *kerning*/jarak antar huruf, perataan, kemiringan, ketebalan, dan lain-lain).

Kejelasan merupakan tingkat kemudahan mata pembaca untuk mengenali rupa huruf tanpa harus bersusah payah. Biasanya kejelasan dipengaruhi oleh kerumitan desain huruf, penggunaan warna, frekuensi kemunculan huruf tersebut sehari-hari. Namun, dalam pengembangan perangkat lunak dan gim, adakalanya kejelasan ini sengaja diabaikan untuk keadaan penyembunyian informasi. Kalian pernah menemukan captcha yang sulit dibaca, kan? Ya, itu dibuat karena kerahasiaannya dari mesin robot.

Pada perancangan antarmuka pengguna, hal yang harus benar-benar diperhatikan adalah antarmuka sangat erat kaitannya dengan pengalaman pengguna (*user experience*/UX). Oleh karena itu, desain antarmuka harus benar-benar memberikan kemudahan, kepuasan, dan pengalaman pada pengguna. Riset atau penelitian akan profil pengguna perlu dilakukan agar benar-benar memahami apa yang diharapkan oleh pengguna.

Aktivitas Belajar 4-5

Buatlah sketsa tampilan awal untuk sebuah halaman web yang menampilkan tempat-tempat wisata di daerahmu dalam versi personal komputer dan *mobile*.

F. Dasar Algoritma Pemrograman

Aktivitas Belajar 4-6

Jika kalian kalian memiliki dua buah gelas yang berisi cairan berwarna, misalkan gelas A berwarna abu dan gelas B berwarna biru, tuliskan langkah-langkah yang kalian lakukan agar isi gelas A berpindah ke gelas B dan sebaliknya.

Kalian tidak akan bisa langsung memerintahkan komputer untuk melakukan pekerjaan tanpa dituliskan baris perintah dalam bahasa yang dimengerti oleh mesin. Manusia dan mesin memiliki perbedaan bahasa. Sama seperti kalian yang memiliki keberagaman bahasa antar suku bangsa satu dengan lainnya. Agar mesin memahami apa yang diinginkan manusia, maka dituliskanlah dalam

bahasa pemrograman. Tentu pada saat kalian menuliskan perintah pada bahasa pemrograman harus dibuat runut (sistematis) agar mesin dapat mengerjakan dengan baik perintah tadi. Runutan langkah-langkah penyelesaian masalah inilah yang dinamakan dengan algoritma.

Algoritma merupakan solusi dalam memecahkan masalah ke dalam program. Algoritma juga merupakan kumpulan perintah untuk menyelesaikan masalah (Maulana, 2017). Algoritma adalah urutan logis pengambilan keputusan pada pemecahan masalah dengan bantuan komputer. Ini berarti nilai kebenaran harus dapat ditentukan sebab langkah-langkah yang tidak benar dapat memberikan hasil yang salah.

Ciri-ciri algoritma yang baik adalah:

1. Memiliki logika atau metode yang tepat.
2. Menghasilkan keluaran yang tepat dalam waktu yang cepat.
3. Ditulis dengan sistematis dan tidak menimbulkan makna ganda.
4. Ditulis dengan format yang mudah dipahami dan mudah diimplementasikan ke dalam bahasa pemrograman.
5. Semua operasi yang dibutuhkan harus terdefinisi dengan jelas.
6. Semua proses harus berakhir setelah sejumlah langkahtertentu dilakukan.

Menurut Donald E. Knuth (Knuth, 2013), algoritma memiliki fitur-fitur:

1. Keterbatasan (*Finitness*), algoritma harus berakhir atau diakhiri jika telah melakukan serangkaian langkah.
2. Kepastian (*Definiteness*), setiap langkah pada algoritma harus dijelaskan dengan tepat dan tidak menimbulkan arti ganda.
3. Masukan (*Input*), algoritma dapat memiliki nol atau beberapa nilai masukan.
4. Keluaran (*Output*), algoritma harus memiliki keluaran baik satu atau beberapa.
5. Efektivitas (*Effectiveness*), langkah-langkah pada algoritma harus dibuat sederhana sehingga dapat dilakukan dengan tepat dan dalam waktu yang cepat.

1. Notasi Algoritma

Kalian pernah tahu sandi morse, kan? Ya, sandi morse merupakan salah satu contoh bagaimana berkomunikasi melalui suatu notasi

tertentu. Notasi pada sandi morse berlaku universal. Artinya notasi sandi morse dibuat sederhana dan tidak menimbukan arti ganda, serta bisa berlaku di mana saja sehingga sandi morse bisa diterapkan pada bentuk-bentuk yang lain, misalkan bunyi bel, lampu sorot, sinyal, dan lain-lain. Demikian juga pada algoritma, algoritma adalah independen terhadap bahasa pemrograman. artinya algoritma yang telah dibuat dapat diterapkan pada berbagai bahasa pemrograman atau tidak mengacu pada salah satu bahasa pemrograman saja. Sebagaimana telah diuraikan sebelumnya, algoritma ditulis dengan format yang sederhana dan mudah dipahami, serta disusun secara sistematis dan tidak menimbulkan arti ganda. Oleh karenanya, algoritma harus dibuat notasi-notasi yang bersifat umum dengan memenuhi kaidah atau standar tertentu agar mudah dimengerti, dipahami, dan diterjemahkan oleh analis maupun *programmer* ke dalam kode program pada bahasa pemrograman yang diinginkan. Secara umum, algoritma dituliskan dalam bentuk: deklaratif, *pseudocode*, dan *flowchart*.

a. Deskriptif

Algoritma ditulis dalam bahasa manusia sehari-hari (misalkan bahasa Indonesia, bahasa Inggris, dan lain-lain) dan dibuat dalam kalimat- kalimat deskriptif. Setiap langkah diterangkan dalam satu atau beberapa kalimat. Perhatikan contoh berikut

Algoritma menentukan_terbesar_tiga_bilangan	
1	Baca masukan tiga bilangan (bilangan 1, bilangan 2, bilangan 3)
2	Jika bilangan 1 lebih besar dari bilangan 2 dan bilangan 3, maka bilangan 1 merupakan bilangan terbesar.
3	Jika tidak, bilangan 1 berarti bukan yang terbesar.
4	Bandingkan bilangan 2 dan bilangan 3. Jika bilangan 2 lebih besar dari bilangan 3, maka bliangan 2 merupakan yang terbesar. Jika sebaliknya, maka bilangan 3 merupakan yang terbesar.
5	Selesai

b. *Pseudocode* (Kode Semu)

Algoritma dituliskan menyerupai bahasa tingkat tinggi tertentu (misalkan bahas C, Pascal, Java, dan lain-lain). Pada dasarnya, *pseudocode* merupakan suatu penulisan yang memungkinkan kalian berpikir terhadap suatu permasalahan

tanpa mempedulikan *syntax* dari bahasa pemrograman. Tidak ada aturan *syntax* yang digunakan pada *pseudocode*, tetapi yang duliskan adalah urutan logika program tanpa memandang bahasa pemrogramannya. Perhatikan contoh berikut.

Algoritma menentukan_terbesar_tiga_bilangan.

```
Deklarasi:
    bil1, bil2, bil2: integer;
    terbesar:integer;
Deskripsi:
    Read (a, b,c)
    if(bil1>bil2) and (bil1>bil3) then
        terbesar ← bil1
    else
        if (bil2>bil3)
            terbesar ← bil2
        else
            terbesar ← bil3
        endif
    endif
    write(terbesar)
```

Tidak ada notasi yang baku dalam pendeskripsian algoritma seperti halnya notasi baku yang dimiliki bahasa pemrograman. Meskipun tidak ada yang baku, tetapi sebaiknya notasi algoritma yang digunakan memiliki korepondensi dengan bahas pemrograman. Perhatikan contoh berikut:

```
Baca masukan nilai a dan b
```

Dalam notasi algoritma dapat ditulis:

```
Read(a,b)
```

`Read` berarti meminta inputan dari piranti masukan untuk kemudian dimasukkan ke dalam nilai a dan nilai b. Dalam bahasa C akan diterjemahkan menjadi `scanf`, dalan C++ menjadi `cin`, dalam pascal menjadi read.

Contoh lain:

```
Isikan nilai bilangan 2 kedalam terbesar
```

Dalam notasi algoritma dapat ditulis:

```
Terbesar ← bilangan 2
```

Notasi ”←” berarti *assign* (mengisi nilai). Dalam bahasa pemrograman dapat ditulis:

```
Terbesar:=bilangan_2;   (bahasa   Pascal)
Terbesar=bilangan_2; (bahasa C dan C++)
```

Agar algoritma mudah dipahami, maka harus ditulis secara teratur dan sistematis. Algoritma dibagi-bagi ke dalam beberapa bagian secara sistematis. Sistematika yang sering digunakan adalah:

1) Kepala (*header*)

Berisi nama dan deskripsi atau penjelasan singkat dari algoritma. Nama algoritma biasanya diawali dengan kata ”algoritma” diikuti dengan nama algoritma. Penamaan algoritma hendaknya dibuat singkat tetapi mencerminkan isi dari algoritma. Beberapa contoh kepala algoritma.

Algoritma hitung_luas_segiEmpat
{menentukan luas bangun segiempat berdasarkan panjang dan lebar yang diperoleh dari masukan pengguna. rumus luas=panjang dikali lebar}

Algoritma menentukan_bilangan_prima
{menentukan apakah bilangan yang dimasukkan oleh pengguna adalah termasuk bilangan prima atau tidak. Algoritma akan menampilkan ”prima” di layar jika bilangan tersebut termasuk bilangan prima dan ”non prima” jika bukan.}

2) Deklarasi

Berisi semua pengenal yang digunakan pada algoritma baik yang bersifat global maupun lokal. Pengenal yang dimaksud dapat berupa peubah (variabel), tetapan (konstanta), tipe data, prosedur, maupun fungsi. Contoh penggunaan dapat dilihat berikut.

Deklarasi	
02	{nama konstanta}
03	const Npeg = 100 {jumlah pegawai}
04	const phi = 3.14 {nilai phi}
05	

```
06  {nama tipe}
07  type TTitik: record {tipe koordinat bidang kartesius}
08                  < x,
09                  y: integer
10                  >
11
12  {nama peubah}
13  c: char {karakter yang dibaca}
14  Q: TTitik {titik dalam koordinat kartesius}
15  ketemu: boolean {keadaan hasil pencarian}
16
17  function IsPrima(input x:integer) boolean
18  {mengembalikan nilai true bila x adalah prima, atau false
19          bila x adalah komposit}
20
21  procedure Tukar(input/output a,b: integer)
22  {mempertukarkan isi variabel a dan b}
```

3) Deskripsi

Berisi langkah-langkah penyelesaian, termasuk meminta masukan, menampilkan keluaran, melakukan pemilihan, dan sebagainya.

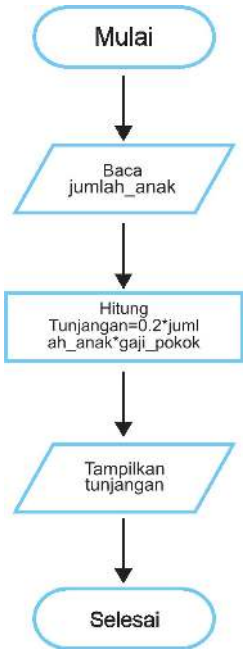
Secara utuh dapat digambarkan sistematikanya sebagai berikut

```
Algoritma nama_algoritma
    {deskripsi algoritma}
Deklarasi
    {deklarasi tipe}
    {deklarasi peubah}
    {deklarasi konstanta}
    {deklarasi prosedur}
    {deklarasi fungsi}
Deskripsi
    {baca masukan}
    {melakukan proses}
    {menampilkan keluaran}
```

Berikut adalah contoh algoritma menghitung luas segi empat secara lengkap.

Algoritma LuasSegiEmpat	
02	{menentukan luas bangun segiempat berdasarkan
03	panjang dan lebar yang diperoleh dari masukan
04	pengguna. rumus luas=panjang dikali lebar}
Deklarasi	
06	{deklarasi peubah}
07	panjang, lebar: **integer**
08	luas:**long**
Deskripsi	
10	{baca masukan}
11	**read**(panjang,lebar)
12	{melakukan proses}
13	luas←panjang*lebar
14	{menampilkan keluaran}
15	**write**(luas)

c. *Flowchart* (Bagan Alir)



Gambar 4.18 Bagan alir hitung tunjangan.

Sumber: Marwondo, Rini Melati, Dana R. N. Adnan/2023

Menyusun algoritma dengan deskriptif atau *pseudocode* masih memiliki kelemahan yaitu dipengaruhi oleh tata bahasa penyusunnya, sehingga kadang-kadang ada yang kesulitan dalam memahaminya. Untuk mengatasinya, maka digunakanlah cara mendeskripsikan algoritma menggunakan simbol-simbol tertentu yang mudah dipahami yang dikenal dengan istilah *flowchart* atau bagan alir.

Bagan alir merupakan suatu bagan yang menggambarkan arus logika dari mulai masuk, diproses, sampai dengan menjadi keluaran. Bagan alir terdiri dari simbol-simbol yang mewakili fungsi-fungsi tertentu dalam program dan garis yang menunjukkan arah langkah. Bagan alir juga merupakan alat grafis yang secara diagram menggambarkan langkah-langkah dan struktur suatu algoritma atau program (Furman, 2010). Perhatikan contoh di samping:

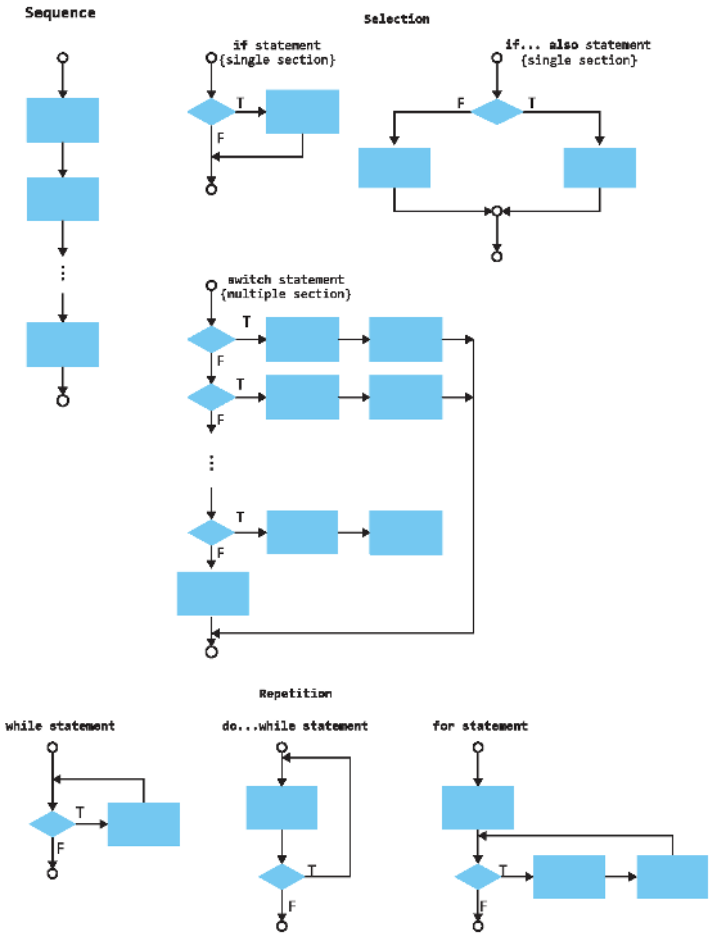
Berikut daftar simbol bagan alir yang umum digunakan

Simbol	Kegunaan
	Terminal. Menyatakan awal dan akhir.
	Data. Menunjukkan operasi masukan dan keluaran.
	Proses. Digunakan untuk menggambarkan proses yang terjadi.
	*Decision*. Menunjukkan pilihan berdasarkan kondisi tertentu.
	*Predefined process*. Menunjukkan proses yang masih perlu didefinisikan terpisah, misalnya pada *sub routine* (prosedur atau fungsi)
	*Off page connector*. Menghubungkandengan bagian dari *flowchart* yang terpisah halaman.
	Konektor. Menghubungkan bagian lain dari *flowchart* pada halaman yang sama.
	Garis Aliran. Arah aliran proses.

Tabel 4.1 Daftar simbol bagan alir
Sumber: Burford J. Furman/2010

Beberapa aturan penggunaan bagan alir yang perlu diperhatikan:

1) Semua simbol yang digunakan pada bagan alir dihubungkan oleh garis aliran, artinya ada arah panah yang muncul.
2) Garis aliran (*flowlines*) masuk dari bagian atas simbol dan keluar dari bagian bawah simbol, kecuali untuk simbol *Decision* dapat keluar dari bawah ataupun samping.
3) Bagan alir secara umum digambar dari atas ke bawah.
4) Awal dan akhir bagan alir ditunjukkan dengan simbol terminal. Contoh penerapan bagan alir pada beberapa keadaan dapat dilihat pada gambar 4.3 berikut.

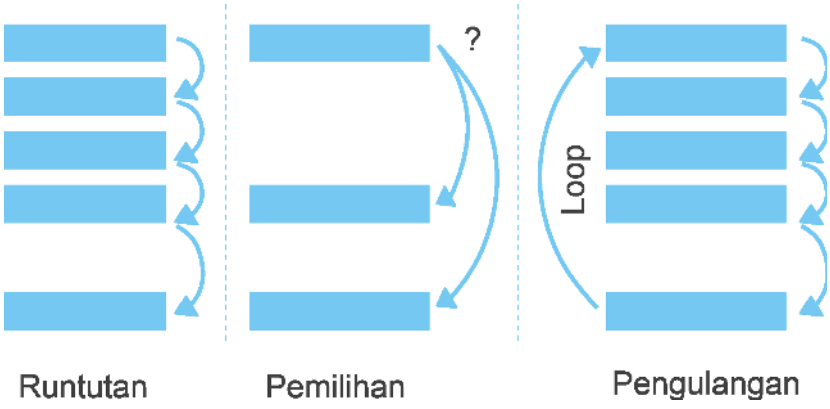


Gambar 4.19 Contoh penggunaan bagan alir
Sumber: Burford J. Furman/2010

2. Konstruksi Dasar Algoritma

Kalian tentu tahu bahwa sebuah gedung yag dibangun tidak langsung jadi begitu saja. Membangun gedung dimulai dengan membangun konstruksi terlebih dahulu. Konstruksi bangunan ini yang akan menentukan wujud bangunan itu akan seperti apa ketika jadi. Demikian juga sebuah algoritma dapat dipandang sebagai sebuah bangunan yang wujudnya akan ditentukan oleh konstruksinya atau struktur dasarnya.

Tahukah kalian, ada berapa struktur dasar algoritma? Ya betul, algoritma pada dasarnya memiliki tiga struktur dasar (Munir & Lidya, 2016), yaitu *sequence* (runtutan), *selection* (pemilihan), dan *repetition* (pengulangan). Sebagai gambaran perbedaan masing-masing, perhatikan gambar 4.4 berikut.



Gambar 4.20 Konstruksi dasar algoritma
Sumber: Marwondo, Rini Melati, Dana R. N. Adnan/2023

a. Runtutan (*Sequence*)

   Struktur ini menunjukkan bahwa deretan program dikerjakan secara berurutan sesuai dengan baris yang dituliskan. Program dieksekusi dari baris pertama sampai baris akhir tanpa melewatkan satu baris pun.

b. Pemilihan (*Selection*)

   Tidak setiap baris akan dieksekusi, melainkan baris yang memenuhi syarat atau kondisi saja yang dieksekusi. Struktur ini akan melakukan pengujian terlebih dahulu terhadap suatu kondisi untuk mengambil suatu keputusan apakah suatu baris atau blok program dieksekusi atau tidak.

c. Pengulangan (*Looping*)

   Struktur ini memungkinkan suatu baris atau blok program dilakukan berulang-ulang tanpa menuliskannya berkali-kali.

Lebih lanjut untuk memahami ketiga struktur tersebut akan dibahas pada bagian lain dari unit pembelajaran ini.

3. Tipe Data, Variabel, dan Konstanta

Pada dasarnya program komputer mengolah data menjadi informasi dengan memanipulasi data yang tersimpan di dalam memori. Setiap data memiliki tipe tersendiri yang dapat menentukan nilai dan operasi yang dapat dilakukan terhadapnya. Tipe data secara umum dikelompokkan menjadi **tipe dasar** dan **tipe bentukan**. Tipe dasar atau disebut juga

dengan tipe generik merupakan tipe data yang langsung dapat digunakan sedangkan tipe bentukan merupakan tipe yang berasal dari tipe dasar atau tipe bentukan lainnya.

Setiap data direpresentasikan pada sebuah variabel (peubah) atau kontanta yang dapat memiliki nilai tertentu sesuai dengan tipenya. Variabel merupakan objek yang nilainya dapat berubah-ubah sedangkan konstanta bernilai tetap.

a. Tipe Data Dasar

Tipe data dasar merupakan tipe data yang sudah kalian kenal dalam kehidupan sehari-hari. Dalam keseharian kalian tidak lepas dari angka-angka serta karakter-karakter tertentu. Tipe data dasar merupakan tipe data yang sudah disediakan oleh pemrograman. Secara umum tipe data dasar terdiri dari: bilangan bulat, bilangan riil, bilangan logika, karakter, dan string (Munir & Lidya, 2016).

Bilangan Bulat, bilangan yang tidak mengandung pecahan desimal. Dalam pemrograman tipe ini dinamakan **integer**. Secara teoritis, bilangan bulat memiliki rentang nilai yang tak terbatas, dari minus tak hingga sampai plus tak hingga. Dalam bahasa pemrograman tipe ini dibagi-bagi lagi ke dalam beberapa jenis berdasarkan rentang nilainya. Hal ini dilakukan untuk mengefisienkan penggunaan memori. Contoh data 25, 11, 0, -12, -1718, 1945, 28032008, 121415161718, dan seterusnya.

Operasi yang dapat dilakukan pada data ini yaitu operasi **aritmatika** dan operasi **perbandingan**. Operasi artimatika yang dilakukan pada bilangan bulat akan **menghasilkan** bilangan bulat juga. Operasi aritmatika yang dapat dilakukan pada bilangan bulat adalah:

Tabel 4.2 Operator aritmatika pada bilangan bulat

Operator	Arti
*	(kali)
div	(bagi)
mod	(sisa bagi)
+	(tambah)
-	(kurang)

Setiap bahasa pemrograman memiliki cara yang berbeda untuk menuliskan operator aritmatika terutama pada ”*div*” dan ”*mod*” serta jumlah operator yang berbeda pula. Contoh penggunaan operasi pada bilangan bulat:

Tabel 4.3 Contoh operasi bilangan bulat

Operasi	Hasil
3*2	(hasil: 6)
5+10	(hasil: 15)
37-3	(hasil: 34)
25 div 5	(hasil: 5)
10 div 3	(hasil: 3)
10 mod 3	(hasil: 1)
25 mod 5	(hasil: 0)

Ingat, operasi pada bilangan bulat menghasilkan bilangan bulat, perhatikan hasil ”*div*”.

Operasi perbandingan pada bilangan bulat dilakukan dengan membandingkan dua buah nilai dan menghasilkan nilai ”benar” (*true*) dan ”salah” (*false*). Operator perbandingan yang dapat digunakan pada bilangan bulat yaitu:

Tabel 4.4 Operator perbandingan

Operator	Arti
=	(sama dengan)
≠	(tidak sama dengan)
<	(lebih kecil dari)
≤	(lebih kecil atau sama dengan)
>	(lebih besar dari)
≥	(lebih besar atau sama dengan)

Contoh penggunaan operasi perbandingan pada bilangan bulat:

Tabel 4.5 Contoh operasi perbandingan

Operasi	Hasil
3<2	(*false*)
5<10	(*true*)
37>5	(*true*)
8≤8	(*true*)
(10 div 3)≠5	(*true*)

10=10	(*true*)
17>17	(*false*)

Bilangan Riil, bilangan yang mengandung pecahan seperti 0.5, 3.14, 0.003, 2.27e10, 3.5e-5 dan lainnya. Bilangan ini biasanya ditandai dengan titik (”.”). Nilai 25 akan dianggap sebagai bilangan bulat sedangkan 25.0 dianggap sebagai bilangan riil. Seperti halnya bilangan bulat, secara teoritis memiliki nilai tak terbatas. Dalam pemrograman, tipe ini dinamakan ***real*** atau ***float***. Sama halnya dengan bilangan bulat, pada pemrograman tipe ini juga memiliki banyak ragam sesuai dengan rentang nilai yang dimiliki.

Operasi yang dikenakan juga **hampir sama** dengan bilangan bulat yaitu operator aritmatika dan perbandingan. Untuk operator aritmatika yang digunakan yaitu:

Tabel 4.6 Operator aritmatika pada bilangan riil

Operator	Arti
*	(kali)
/	(bagi)
+	(tambah)
-	(kurang)

”/” berbeda dengan ”*div*”. Jika ”*div*” yang dihasilkannya adalah bilangan bulat (pembulatan terkecil), maka ”/” menghasilkan pecahan (riil). Contoh, 10 div 3 akan menghasilkan 3, sedangkan 10/3 akan menghasilkan 3.3333.

Sebagian besar bahasa pemrograman memperbolehkan operasi campuran antara bilangan bulat dan bilangan riil. Bilangan bulat jika dikonversikan ke dalam bilangan riil, hasilnya akan merupakan bilangan riil. Operator perbandingan yang digunakan pun hampir sama dengan bilangan bulat, tetapi minus ”=”. Tipe data bilangan bulat dan bilangan riil bisa juga disebut dengan tipe data **numerik**.

Bilangan Logika, merupakan hasil perbandingan logika yang bernilai ”benar” (*true*) dan ”salah” (*false*). Dalam pemrograman, tipe ini dinamakan ***boolean***. Operasi yang dapat dilakukan adalah operasi- operasi logika, yaitu: AND, OR, XOR, dan NOT. Perhatikan tabel perbandingan berikut.

Tabel 4.7 Perbandingan operator logika

x	y	NOT x	NOT y	x AND y	x OR y	x XOR y
TRUE	*TRUE*	*FALSE*	*FALSE*	*TRUE*	*TRUE*	*FALSE*
TRUE	*FALSE*	*FALSE*	*TRUE*	*FALSE*	*TRUE*	*TRUE*
FALSE	*TRUE*	*TRUE*	*FALSE*	*FALSE*	*TRUE*	*TRUE*
FALSE	*FALSE*	*TRUE*	*TRUE*	*FALSE*	*FALSE*	*FALSE*

Karakter, adalah semua karakter yang ada di kehidupan sehari-hari termasuk semua abjad (a-z), semua angka (0-9), karakter khusus ('&', '@', dan sebagainya), serta karakter ***Null***. Dalam pemrograman, tipe ini dinamakan dengan ***char***. Rentang nilai tipe ini meliputi semua jenis karakter. Standar karakter dapat dilihat pada ASCII. Dalam penggunaannya, nilai karakter diapit dengan kutip tunggal ' '. Contoh: 'Y', 'c', ':', '/'. Dalam bahasa pemrograman penamaan tipe ini mengikuti isi dari karakter yang dapat ditampung oleh memori.

Operasi yang dapat dilakukan terhadap tipe ini adalah operator perbandingan seperti pada bilangan bulat. Komputer memiliki sistem pengkodean tertentu semisal ASCII untuk menentukan urutan penomoran karakter. Hal inilah yang memungkinkan kalian dapat membandingkan "<", ">", "≤", "≥" pada karakter. Dalam sistem pengkodean juga memisahkan antara huruf kecil dan huruf besar, lho, jadi jika kalian membandingkan huruf kecil dan huruf besar maka dianggap karakter yang berbeda. 'A' tidak sama dengan 'a'. Perhatikan contoh berikut.

Tabel 4.8 Contoh operasi karakter

Operasi	Hasil
'A' = 'a'	(*false*)
'b' = 'b'	(*true*)
'm'<'z'	(*true*)
'X' > 'Y'	(*false*)
'2' < '5'	(*true*)
'c' ≠ 'C'	(*true*)

String, merupakan untaian karakter dengan panjang tertentu. Sebenarnya tipe ini merupakan tipe data yang tersusun dari elemen- elemen karakter sehingga dapat juga dikatakan tidak murni sebagai tipe dasar. Namun tipe ini sering digunakan

dalam pemrograman, sehingga dapat dimasukkan dalam kategori tipe dasar (Munir & Lidya, 2016).

Dalam pemrograman, tipe ini dinamakan ***string***. Dalam bahasa pemrograman ada juga yang menyebutnya dengan *NumChar*, *VarChar,* dan lain-lain. Nilai pada tipe ini diapit oleh kutip ganda " ". Contoh tipe data ini: "Indonesia", "Perangkat Lunak", "Gim", "Budaya", "Laki-laki", "Perempuan". Operasi yang bisa dilakukan pada tipe ini yaitu penyambungan (*concate*) dan perbandingan. Operasi penggabungan dilakukan dengan operator "+". Jika pada numerik "+" berarti mengakumulasikan nilai kedua data, maka pada string kedua nilai digabungkan. Misalkan a=1 dan b=1 maka pada tipe data numerik a+b=2 tetapi pada string a+b="11". Perhatikan contoh lain berikut ini.

"pengembang" + "gim" ="pengembang gim"
"Indonesia" +"Raya" = "Indonesia Raya"
"aku" + "bahagia" = "aku bahagia"

Operator perbandingan yang digunakan sama seperti pada tipe karakter. Bila dua string dibandingkan, maka perbandingan dilakukan masing-masing elemen yang menyusunnya. Contoh:

"abcd"="abc" (hasil: *false*)
"aku"<"AKU" (hasil: *true*)

b. Tipe Data Bentukan

Tipe data ini merupakan tipe data yang didefinisikan sendiri oleh pemrogram sesuai dengan kebutuhan atau tipe tersebut tidak tersedia dalam pemrograman. Tipe data bentukan dapat dibentuk oleh satu oleh lebih tipe data dasar. Dalam pemrograman, tipe data baru dideklarasikan pada bagian deklarasi dengan kata kunci ***type***.

Secara umum tipe data bentukan dapat berupa:

1) Tipe data dasar

Tipe data dasar yang diubah menjadi tipe baru, biasanya digunakan untuk memudahkan pemrogram mengingat dan menjaga konsistensi peubah. Pada deklarasi dapat kalian tulis:

Deklarasi	
	type *namatipe*: **TipeDasar**

Misalkan kalian ingin membuat *type* "BilanganBulat" dari tipe integer, kalian dapat menuliskannya sebagai berikut.

```
Deklarasi
02  type BilanganBulat: integer
03  x:BilanganBulat
```

x merupakan peubah dengan tipe BilanganBulat yang sebenarnya memiliki tipe integer.

2) Tipe terstruktur

Tipe ini merupakan tipe data yang berisi data terstruktur berupa rekaman yang disusun oleh beberapa *field*. Tiap *field* memiliki tipe data tertentu sesuai dengan yang diinginkan. Tipe data ini ditulis sebagai berikut:

```
Deklarasi
02  type namatipe: record <field1:tipe,
03  field2:tipe,..., fieldn:tipe>
```

Misalkan sebuah titik pada koordinat kartesius dinyatakan dengan (x,y) maka kalian dapat

```
Deklarasi
02  type Titik: record <x:real,
03  y:real>
04  p:Titik
```

Untuk memanggil elemen dari tipe data tersebut dapat dituliskan:

```
peubah.field
```

Contoh jika p adalah sebuah titik dengan nilai x = 2.35 dan y = 4.72, maka untuk mengisi nilai serta memanggil nilai-nilai tersebut dapat dituliskan:

```
p.x=2.35
p.y=4.72
write (p.x)
write (p.y)
```

3) Variabel dan Konstanta

Saat melakukan pemrograman, kalian harus menyimpan data sementara dulu ke dalam sebuah penampungan.

Penampung data sementara inilah yang dinamakan dengan variabel (peubah) dan konstanta. Variabel maupun konstanta harus dideklarasikan dan diinisialisasi terlebih dahulu sebelum digunakan. Deklarasi variabel dan konstanta ditulis pada bagian Deklarasi dengan format nama variabel diikuti tanda ":" dan tipedata-nya sedangkan konstanta ditulis dengan format ***const*** diikuti dengan *nama konstanta* tanda "=", dan *nilai*-nya.

```
Deklarasi
02  variabel: tipeData
03  const namaKonstanta =nilai
```

Variabel merupakan objek pada pemrograman yang nilainya dapat diubah-ubah melalui perintah yang diberikan dalam pemrograman. Sebuah variabel mewakili suatu data tertentu, oleh karena itu harus didefinisikan tipe datanya baik berupa tipe data dasar maupun tipe data bentukan.

```
Deklarasi
02  type Siswa: record <NIS:string,
03                      Nama:string,
04                      tahun_masuk:integer>
05  x,y: integer
06  z: integer sw: Siswa
07  sw: Siswa
```

Variabel yang digunakan untuk kelompok tujuan yang sama dan memiliki tipe data yang sama dapat dituliskan pada satu baris dengan menggunakan koma ",". Pada contoh di atas dapat dilihat pada "x" dan "y". Penerapan cara ini berbeda untuk variabel yang menggunakan tipe data bentukan. Tipe data bentukan harus didefinisikan terlebih dahulu sebelum digunakan. Perhatikan variabel "sw" yang menggunakan tipe "Siswa".

Untuk memberi nilai awal (inisialisasi) dapat dilakukan pada bagian deskripsi, begitu pun untuk mengubah nilai atau memanggil nilai.

```
Algoritma PenjumlahanDuaBilangan
02  {menjumlahkan dua bilangan berdasarkan
03  masukan dan menampilkan hasilnya pada layar}
```

```
Deklarasi
05  {deklarasi peubah}
06  a,b: integer
07  c:integer
Deskripsi
09  {memberi nilai pada a dan b}
10  a←3
11  b←4
12  {memberi nilai pada c dari operasi a dan b}
13  c←a+b
14  {menampilkan nilai a, b, dan c}
15  write (”Bilangan pertama =”,a)
16  write (”Bilangan kedua =”,b)
17  write (”Hasil penjumlahan =”,c)
```

Konstanta merupakan objek dalam pemrograman yang nilainya tidak berubah-ubah atau tetap. Konstanta bisa juga digunakan untuk menyatakan batasan-batasan tertentu misalkan nilai maksimal, batas maksimal pengulangan, jumlah elemen, dan sebagainya. Berbeda dengan variabel, konstanta tidak perlu menyebutkan tipe datanya, tetapi menyebutkan nilainya.

```
Deklarasi
02  const pi=3.14
03  const Nmaks=100
04  const key=”1234567890”
```

Menggunakan konstanta pada dasarnya sama dengan menggunakan variabel, perhatikan contoh berikut.

```
Algoritma LuasLingkaran
02  {menghitung luas lingkaran}
Deklarasi
04  {deklarasi peubah}
05  const pi=3.14
06  r:integer {jari-jari}
08  luas:real
Deskripsi
09  {meminta masukan nilai r}
10  read(r)
11  luas←pi*r*r
```

```
12  write(”panjang jari-jari =”,r)
13  write(”Luas lingkaran =”, luas)
```

Beberapa aturan penamaan variabel dan konstanta yang harus kalian perhatikan:

a) Dimulai dengan huruf alfabet, tidak boleh diawali dengan angka atau simbol.

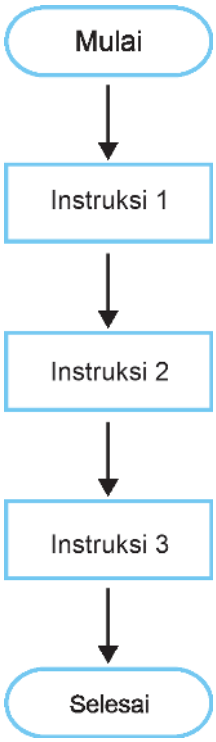
b) Tidak menggunakan spasi atau karakter khusus lainnya kecuali garis bawah ”_”.

c) Tidak mengandung operator aritmatika, logika, atau perbandingan.

d) Tidak menggunakan *reserved word* (kata-kata yang sudah didefinisikan pada bahasa pemrograman).

e) Mewakili isi data dan tidak terlalu panjang.

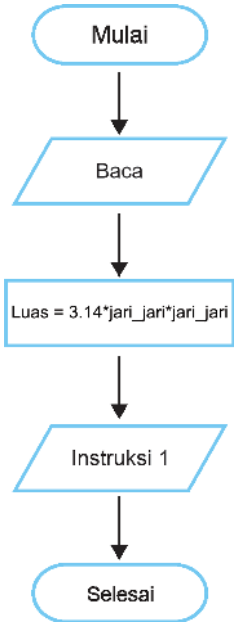
4. Runtutan (*sequence*)



Gambar 4.21 Contoh struktur sekuen
Sumber: Marwondo & Rini Melati/2022

Struktur runtutan merupakan struktur yang langkah-langkah pengerjaannya (instruksi) dilakukan berurutan dari awal hingga akhir. Setiap instruksi dikerjakan secara berurutan (sekuensial) sesuai dengan urutan yang ada dalam algoritmanya. Instruksi sebelumnya selesai dikerjakan terlebih dahulu sebelum mengerjakan instruksi berikutnya. Perhatikan bagan alir di samping!

Dari gambar 4.21 dapat kalian lihat bahwa algoritma memiliki tiga instruksi yang tersusun berurutan. Program akan mengeksekusi terlebih dahulu instruksi 1. Setelah instruksi 1 selesai maka dilanjutkan ke instruksi 2 dan diteruskan ke instruksi 3 sampai selesai. Berikut contoh struktur runtutan pada algoritma menghitung luas lingkaran.



Gambar 4.22 Bagan alir menghitung luar lingkaran
Sumber: Marwondo & Rini Melati/2022

Dari gambar 4.22 kalian bisa melihat bahwa program hitung luas lingkaran memiliki tiga langkah, yaitu:

1) Langkah 1: membaca masukan jari-jari.
2) Langkah 2: menghitung luas lingkaran.
3) Langkah 3: mencetak atau menampilkan luas di layar.

Langkah 2 tidak dilakukan jika langkah 1 belum dilakukan, demikian juga langkah 3 tidak dilakukan jika langkah 2 belum dilakukan.

Dalam struktur runtutan, urutan instruksi yang disusun menggambarkan logika berpikir penulis algoritma. Kesalahan urutan penulisan bisa menyebabkan solusi yang dibuat juga salah. Misalkan pada algoritma pertukaran gelas, sebelumnya logika yang ditulis seperti berikut.

Algoritma menukar_isi_gelas	
1.	Tuangkan larutan dari gelas A (Abu) ke gelas C (cadangan)
2.	Tuangkan larutan dari gelas B (Biru) ke gelas A (Abu)
3.	Tuangkan larutan dari gelas C (Cadangan) ke gelas B (Biru)

Jika algoritma tersebut diubah menjadi seperti berikut, apa yang terjadi?

Algoritma menukar_isi_gelas	
1.	Tuangkan larutan dari gelas A (Abu) ke gelas C (Cadangan)
2.	Tuangkan larutan dari gelas C (Cadangan) ke gelas B (Biru)
3.	Tuangkan larutan dari gelas B (Biru) ke gelas A (Abu)

Hasilnya salah, bukan? Ini yang dinamakan dengan kesalahan logika pada pemrograman. Urutan logika sangat penting pada sebuah pemrograman. Kesalahan logika berdampak pada kesalahan hasil.

5. Pemilihan (*selection*)

Dalam kehidupan sehari-hari, tentu ada kegiatan yang kalian lakukan berdasarkan situasi dan kondisinya. Contoh sederhana adalah ketika kalian mengendarai kendaraan dan sampai pada persimpangan yang memiliki ***traffic light***. Kalian akan berhenti **jika** lampu berwarna merah dan berjalan **jika** lampu berwarna hijau. Begitu pun dalam algoritma, tidak selalu setiap baris akan dieksekusi secara berurutan, adakalanya pada kondisi tertentu baris tersebut harus dilewat menuju baris lainnya.

Bentuk instruksi seperti inilah yang dinamakan dengan struktur pemilihan atau percabangan. Algoritma pemilihan atau percabangan memerlukan syarat tertentu untuk dapat dieksekusi. Struktur ini akan menguji, apakah baris tersebut memenuhi syarat atau tidak. Secara umum bentuk percabangan dituliskan dengan:

a. **Jika** kondisi **maka** aksi (dalam deskriptif).
b. **If** *kondisi* **then** *a*ksi (dalam pseudocode).
c. Simbol ”♦” (bagan alir).

Kondisi merupakan syarat yang diminta dan bernilai ”benar” atau ”salah” sedangkan aksi adalah kegiatan yang dilakukan jika kondisi bernilai ”benar”.

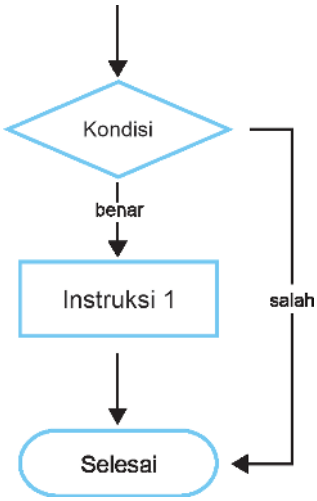
Dalam realita permasalahan pemrograman, kalian tidak hanya menghadapi satu kondisi saja, tetapi juga ada beberapa kondisi yang harus terpenuhi. Algoritma pemilihan menyediakan kemungkinan untuk menangani lebih dari satu kasus.

a. Satu Kasus

Notasi konstruksi algoritma untuk pemilihan dengan satu kasus ditulis dengan menggunakan if-then sebagai berikut.

if kondisi **then**
aksi
endif

Bentuk instruksi tersebut menandakan bahwa aksi hanya dilakukan jika kondisi bernilai ”benar”. Jika kondisi bernilai ”salah”, tidak ada aksi apapun yang dilakukan. Perhatikan bagan alir pada gambar 4.23 berikut untuk lebih jelasnya.



Gambar 4.23 Bagan alir pemilihan satu kasus
Sumber: Marwondo & Rini Melati/2022

Misalkan kalian diminta membuat algoritma untuk menerima masukan angka dan menampilkan pesan “ganjil” jika bilangan tersebut adalah ganjil, maka kalian dapat menuliskannya seperti berikut.

```
Algoritma BilanganGanjil
02  {mencetak pesan ”Ganjil” jika bilangan bulat
03  yang dimasukkan adalah bilangan ganjil}
Deklarasi
05  {deklarasi peubah}
06  x: integer
Deskripsi
08  {baca masukan}
09  read(x)
10  {melakukan proses}
```

```
11  if x mod 2 = 1 then
12        write (”Ganjil”)
13  endif
```

Contoh lain misalkan kalian diminta menampilkan pesan bahwa karakter yang dimasukkan adalah huruf vokal, kalian dapat menuliskan algoritmanya seperti berikut.

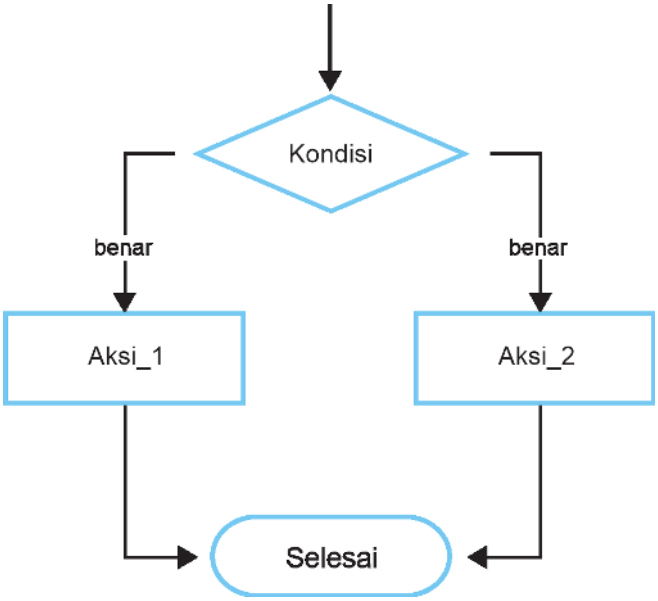
```
Algoritma HurufVokal
02  {mencetak pesan ”Vokal” jika karakter yang
03  dimasukkan termasuk a, i, u, e, o}
Deklarasi
05  {deklarasi peubah}
06  c: char
Deskripsi
08  {baca masukan}
09  read(c)
10  {melakukan proses}
11  if c=’a’ or c=’i’ or c=’u’ or c=’e’ or c=’o’ then
12        write (”Vokal”)
13  endif
```

b. Dua Kasus

Bentuk **if-then** yang sudah dijelaskan sebelumnya hanya dapat mengakomodir satu alternatif keadaan, tetapi dalam kenyataannya keadaan yang dipersyaratkan tidak hanya satu. Suatu saat kalian akan dihadapkan pada dua pilihan yang hanya boleh dipilih salah satunya. Nah, untuk menyelesaikan masalah tersebut, maka digunakanlah struktur **if-then-else**. Berikut bentuk notasi penggunaannya.

```
if kondisi then
    Aksi_1
else
    Aksi_2
endif
```

Aksi_1 dikerjakan jika kondisi ”benar” dan aksi_2 dikerjakan jika kondisi ”salah”. Perhatikan bagan alir pada gambar 4.24 berikut untuk memperjelas.



Gambar 4.24 Bagan alir pemilihan dua kondisi
Sumber: Marwondo & Rini Melati/2022

Perhatikan kembali algoritma bilangan ganjil yang sebelumnya dicontohkan. Sebuah bilangan bulat yang dimasukkan ke dalam program harusnya akan dibaca "ganjil" atau "genap". Jika bilangan tersebut habis dibagi 2, maka bilangan tersebut adalah bilangan genap, tetapi jika tidak habis dibagi 2, maka bilangan tersebut merupakan bilangan ganjil. Kasus tersebut dapat kalian gambarkan algoritmanya menjadi seperti berikut.

```
Algoritma BilanganGanjilGenap
{mencetak pesan "Ganjil" jika bilangan bulat
yang dimasukkan adalah bilangan ganjil dan
mencetak "Genap" jika bilangan tersebut adalah
bilangan genap}
Deklarasi
{deklarasi peubah}
x: integer
Deskripsi
{baca masukan}
read(x)
{melakukan proses}
if x mod 2 = 1 then
     write ("Ganjil")
```

```
15  else
16        write (”Genap”)
17  endif
```

Contoh lain misalkan kalian diminta untuk membandingkan dua bilangan. Dari kedua bilangan tersebut, mana yang lebih besar. Kalian bisa menuliskan algoritmanya seperti berikut.

```
Algoritma BilanganTerbesar
02  {mencetak bilangan terbesar diantara dua
03  bilangan yang dimasukkan}
Deklarasi
05  {deklarasi peubah}
06  x,y: integer
Deskripsi
08  {baca masukan}
09  read(x,y)
10  {melakukan proses}
11  if x > y then
12        write (”Bilangan terbesar adalah ”, x)
13  else
14        write (”Bilangan terbesar adalah ”, y)
15  endif
```

Aktivitas Belajar 4-7

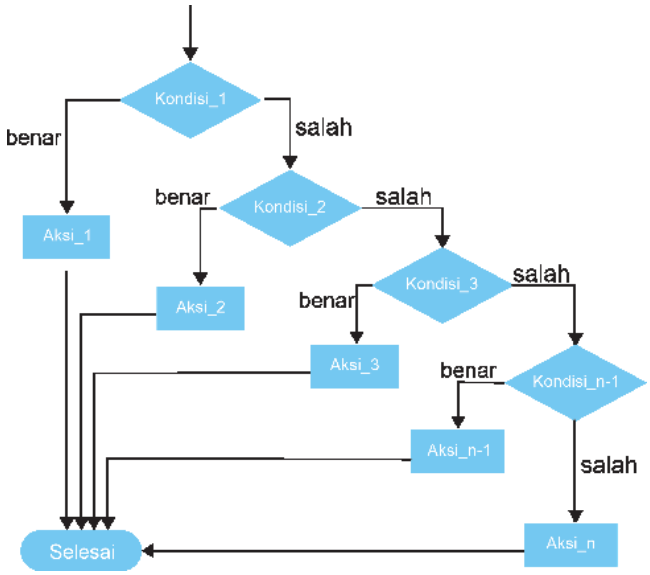
Buatlah algoritma (deskriptif atau pseudocode atau bagan alir) untuk menentukan tahun kabisat. Tahun kabisat adalah tahun yang habis dibagi 4.

c. Tiga Kasus atau Lebih

Bagaimana kalian akan mendefinisikan algoritmanya jika kalian diminta untuk menentukan sebuah bilangan bulat yang dimasukkan ke dalam program apakah termasuk bilangan bulat positif, bilangan bulat negatif, atau nol? Ya, tentu tidak lagi menggunakan dua kondisi, kan? Nah, pada algoritma, kalian bisa menggunakan struktur **if-then-else if- then** jika menemukan kondisi lebih dari dua. Struktur ini biasa juga dikenal dengan istilah ”**if** bertingkat” atau ”**if** bersarang”. Berikut notasi yang digunakan.

```
if kondisi_1 then
        Aksi_1
else if kondisi_2 then
        Aksi_2
else if kondisi_3 then
        Aksi_3
...
else if kondisi_n-1
        Aksi_n-1
else
        Aksi_n
endif
```

n menandakan jumlah keadaan atau kondisi serta aksi yang akan dilakukan. Nah, perhatikan bagan alir pada gambar 4.25 berikut.



Gambar 4.25 Bagan alir pemilihan lebih dari dua kondisi
Sumber: Marwondo & Rini Melati/2022

Jadi, bagaimana algoritma untuk menentukan bilangan bulat negatif, positif atau nol? Perhatikan contoh berikut.

```
Algoritma BilanganBulat
02  {menentukan bilangan bulat positif, negatif,
03  atau nol}
Deklarasi
05  {deklarasi peubah}
06  b: integer
Deskripsi
08  {baca masukan}
09  read(b)
10  {melakukan proses}
```

```
11  if b > 0 then
12     write (”Bilangan bulat positif”)
13  else if b<0 then
14     write (”Bilangan bulat negatif”)
15  else
16     write (”Nol”)
17  endif
```

Agar kalian bisa lebih memahami, perhatikan masalah berikut. Kalian diminta untuk menentukan sebuah titik (x,y) pada koordinat kartesius. Berdasarkan nllai x dan y-nya, tentukan titik tersebut ada pada kuadran berapa. Ketentuan kuadran didefinisikan sebagai berikut:

Kuadran I: x>0 dan y>0
Kuadran II: x<0 dan y>0
Kuadran III: x<0 dan y<0
Kuadran IV: x>0 dan y<0

Selain keempat kuadran tersebut, berarti titik tersebut ada di titik pusat. Algoritma dari masalah tersebut dapat kalian tuliskan seperti berikut.

```
Algoritma KuadranTitik
02  {menentukan kuadran dari sebuah titik pada
03  koordinat tertentu}
Deklarasi
05  {deklarasi peubah}
06  x,y: integer"
Deskripsi
08  {baca masukan}
09  read(x,y)
10  {melakukan proses}
11  if x > 0 and y>0 then
12     write (”Titik berada di Kuadran I”)
13  else if x<0 and y>0 then
14     write (”Titik berada di Kuadran II”)
15  else if x<0 and y<0 then
16     write (”Titik berada di Kuadran III”)
17  else if x>0 and y<0 then
18     write (”Titik berada di Kuadran IV”)
19  else
20     write (”Titik Pusat”)
21  endif
```

Aktivitas Belajar 4-8

Buatlah algoritma (deskriptif atau *pseudocode* atau bagan alir) untuk menentukan nilai mutu sebuah mata pelajaran berdasarkan nilai akhir yang didapat. Nilai mutu ditentukan sebagai berikut.

Nilai Mutu	Nilai Akhir
A	>85
A-	≤85 dan >80
B+	≤80 dan >75
B	≤75 dan >70
B-	≤70 dan >65
C	≤65 dan >60
D	≤60 dan >55
E	≤55

d. Konstruksi Case

Jika permasalahan yang ditemukan memiliki lebih dari dua kasus dan tidak memerlukan pemeriksaan berjenjang, maka kalian dapat menyelesaikannya dengan pemilihan berbentuk *case*. "**case**" secara umum dapat dituliskan sebagai berikut.

```
case (ekspresi)
   kasus_1: aksi_1
   kasus_2: aksi_2
   kasus_3: aksi_3
   ...
   kasus_n: aksi_n
   otherwise : aksi_x
end case
```

Ekspresi adalah sesuatu yang menghasilkan nilai, kasus adalah nilai yang dihasilkan oleh ekspresi. Pada dasarnya *case* memiliki fungsi yang sama dengan struktur *if*, tetapi pada *case* aksi tidak perlu menunggu kondisi sebelumnya tidak terpenuhi, program bisa langsung melakukan aksi pada kondisi tersebut secara langsung, sedangkan struktur *if* harus menunggu kondisi sebelumnya "salah" baru akan dijalankan. Misalkan dalam permasalahan kuadran kalian memiliki 4 kondisi dan 4 aksi. Jika ingin melakukan aksi_4, maka kondisi_1 sampai kondisi_3 harus bernilai "salah" sehingga aksi_4 bisa dijalankan. Hal ini berbeda dengan *case* yang bisa langsung menuju kasus_4 tanpa harus melalui kasus 1 sampai kasus 3.

Case memeriksa apakah nilai yang dihasilkan dari ekspresi sama dengan salah satu kasus, jika sama maka aksi pada kasus tersebut akan dilakukan. Sayangnya, tidak semua bahasa pemrograman menyediakan struktur ini, tetapi dapat kalian ganti dengan struktur *if* yang ekuivalen.

Misalkan kalian diminta membuat algoritma untuk mengubah angka 1- 9 menjadi kata, kalian dapat menuliskan struktur if sebagai berikut.

```
if angka=1 then
   write ("Satu")
else if angka=2 then
   write ("Dua")
else if angka=3 then
   write ("Tiga")
else if angka=4 then
   write ("Empat")
else if angka=5 then
   write ("Lima")
else if angka=6 then
   write ("Enam")
else if angka=7 then
   write ("Tujuh")
else if angka=8 then
   write ("Delapan")
else if angka=9 then
   write ("Sembilan")
else
   write ("Tidak termasuk")
endif
```

dengan struktur case, kalian dapat menuliskannya sebagai berikut.

```
case (angka):
   1: write ("Satu")
   2: write ("Dua")
   3: write ("Tiga")
   4: write ("Empat")
   5: write ("Lima")
   6: write ("Enam")
   7: write ("Tujuh")
   8: write ("Delapan")
   9: write ("Sembilan")
   otherwise write ("Tidak termasuk")
end case
```

Aktivitas Belajar 4-9

Buatlah algoritma (deskriptif atau *pseudocode* atau bagan alir) untuk menentukan jumlah hari pada bulan tertentu. Misalkan Januari 2008 jumlah hari 31, Februari 2008 jumlah hari 29.

6. Pengulangan (*Looping*)

Pernahkah kalian mengerjakan sesuatu yang diulang-ulang? Ya, pasti pernah melakukannya. Kalian bisa melakukan aktivitas yang sama yang dilakukan setiap hari. Bangun tidur-mandi-sarapan-berangkat ke sekolah- belajar-pulang-istrihat begitu seterusnya aktivitas ini kalian lakukan setiap hari. Suatu permasalahan bisa saja diselesaikan secara berulang-ulang. Pada saat membangun sebuah rumah, misalkan, memasang genting atau memasang bata merupakan kegiatan yang dilakukan berulang-ulang.

Apa yang akan kalian lakukan jika diminta mencetak angka dari 1 sampai 100? Tentunya kalian tidak akan menuliskan `write(1)`, `write(2)` dan seterusnya, kan? Ya, kalian bisa menggunakan struktur pengulangan untuk melakukan hal tersebut.

Struktur pengulangan merupakan perintah yang digunakan pada deretan aksi yang menunjukkan pola berulang. Struktur ini dalam pemrograman biasa disebut *looping* atau *repetition*. Secara umum struktur pengulangan terdiri dari dua bangian utama yaitu (Munir & Lidya, 2016):

a. Kondisi pengulangan, kondisi yang harus dipenuhi untuk melaksanakan pengulangan dan harus bernilai ”benar” atau ”salah”.
b. Badan pengulangan, sekumpulan aksi yang akan dilakukan secara berulang.

Dalam pemrograman, ada berbagai jenis pengulangan yang dapat digunakan. Beberapa jenis pengulangan dapat digunakan menyelesaikan masalah yang sama, akan tetapi jika kalian salah menggunakan jenis pengulangan maka hasilnya pun akan salah. Setiap bahasa pemrograman memiliki cara yang berbeda dalam mendefinisikan jenis pengulangan. Secara umum struktur pengulangan dibagi menjadi struktur **dengan kondisi** dan **tanpa kondisi**.

a. Struktur Tanpa Kondisi

   Struktur tanpa kondisi merupakan struktur pengulangan yang instruksi-instruksi di dalamnya dilakukan pengulangan sejumlah n kali sehingga jumlah pengulangan sudah diketahui sebelumnya. Pada pemrograman struktur ini menggunakan perintah ”**for**”.

Untuk menghitung (mencacah) berapa kali pengulangan dilakukan, kalian perlu variabel pencacah (*counter*) yang memiliki *predecessor* dan *successor*. Biasanya dalam bentuk integer. Bentuk umum **for** ditulis sebagai berikut:

```
for pencacah ← nilai_awal to nilai_akhir do
   aksi
end for
```

Contoh pada kasus menuliskan angka 1 sampai 100, kalian dapat tuliskan menjadi berikut

```
for i ← 1 to 100 do
   write (i)
end for
```

Jumlah atau batas pengulangan juga bisa kalian tentukan pada sebuah variabel lain, lho. Misalkan kalian diminta menghitung rata-rata sejumlah bilangan yang dimasukkan. Kalian dapat menuliskan algoritmanya sebagai berikut.

```
Algoritma HitungRata2
02 {menghitung rata-rata sejumlah bilangan
03 tertentu}
Deklarasi
05 {deklarasi peubah}
06 n: integer {banyak data yang dimasukkan}
07 x: integer {bilangan yang dimasukkan}
08 i: integer {pencacah}
09 jml: integer {pencatat jumlah data}
10 rerata: real {nilai rata-rata}
Deskripsi
12 {baca masukan}
13 read(n) {jumlah pengulangan}
14 jml←0 {inisialisasi awal pencatat jumlah}
15 {melakukan proses}
16 for i=1 to n do
17    read(x) {baca bilangan}
18    jml←jml+x
19 end for
20 rerata←jml/n
21 write (rerata)
```

Aktivitas Belajar 4-10

Buatlah algoritma (deskriptif atau *pseudocode* atau bagan alir) untuk menampilkan bilangan prima antara 1-100. Bilangan prima adalah bilangan yang hanya habis dibagi oleh dirinya sendiri.

b. Struktur dengan Kondisi

Pada beberapa masalah, jumlah pengulangan tidak dapat diketahui secara pasti sebelumnya, tetapi pengulangan tersebut dapat mulai atau berhenti pada keadaan tertentu. Instruksi-instruksi pada pengulangan akan terus dilakukan sampai dengan kondisi berhenti terpenuhi. Struktur ini direpresentasikan dengan perintah **while** dan **repeat**.

Secara umum struktur **while** dituliskan sebagai berikut:

```
while kondisi do
   aksi
end while
```

Untuk melakukan pengulangan, kondisi harus terpenuhi terlebih dahulu dan berhenti jika kondisi tidak terpenuhi. Yang harus kalian perhatikan dengan baik adalah pastikan bahwa kondisi yang kalian buat suatu saat akan bernilai ”salah” karena jika tidak, maka pengulangan tidak pernah berhenti. Hal ini dapat menyebabkan komputer hang. Berikan instruksi pada badan pengulangan yang akan mengubah kondisi.

Perhatikan contoh kasus berikut. Jika kalian mencetak angka 1 sampai 100, maka kalian juga bisa menuliskannya seperti berikut.

```
while i≤100 do
   write (i)
   i←i+1
end while
```

Sekilas antara *while* dan *for* mirip, tetapi jika data yang kalian olah tidak diketahui pasti maka *while* lebih tepat digunakan. Misalkan kalian diminta menghitung rata-rata nilai siswa di kelas, sedangkan data yang dimasukkan tidak diketahui pasti jumlahnya sampai dipilih tidak mengisi lagi dengan memasukkan nilai minus (-). Kalian bisa menuliskannya seperti berikut.

```
Algoritma HitungRata2While
{menghitung rata-rata sejumlah bilangan
tertentu menggunakan while}
Deklarasi
{deklarasi peubah}
n: integer {banyaknya data yang dimasukkan}
x: integer {bilangan yang dimasukkan}
i: integer {pencacah}
jml: integer {pencatat jumlah data}
rerata: real {nilai rata-rata}
Deskripsi
{baca masukan}
read(x) {nilai yang dihitung}
jml←0 {inisialisasi awal pencatat jumlah}
i←0 {inisialisasi awal pencacah}
{melakukan proses}
while i<n do
   read (x)
   jml←jml+x
   i←i+1
end while
if n≠0 then
   rerata←jml/n
   write (rerata)
else
   write (”Tidak ada data”)
endif
```

Selain *while*, struktur lain pada bagian ini adalah **repeat**. Berbeda dengan *while* yang memeriksa kondisi di awal, **repeat** melakukan pemeriksaan kondisi di akhir. Bentuk umum repeat dituliskan sebagai berikut:

```
repeat
   aksi
until kondisi
```

Pada struktur ini, aksi dilakukan minimal satu kali. Jika kalian memilih menu, apakah menunya tampil dulu sampai kita pilih tutup? Jika kita tidak memilih tutup atau keluar maka menu akan terus ditampilkan. Nah, keadaan seperti ini memerlukan struktur **repeat...until**. dalam beberapa bahasa

pemrograman, struktur ini ada yang direpresentasikan dalam bentuk **do...while** atau **do...loop** atau **do..until**.

Misalkan kalian ingin membuat program yang menampilkan menu kemudian berdasarkan menu yang dipilih maka akan dilakukan aksinya. Kalian dapat menuliskannya sebagai berikut.

```
Algoritma TampilMenu
{menampilkan menu, membaca pilihan, dan
menampilkan nomor menu sesuai dengan pilihan}
Deklarasi
{deklarasi peubah}
pilihan: integer
Deskripsi
repeat
   {cetak menu}
   write ("Menu")
   write ("1. Baca Data")
   write ("2. Cetak Data")
   write ("3. Ubah Data")
   write ("4. Hapus Data")
   write ("5. Keluar")
   write ("Masukkan pilihan anda (1/2/3/4/5")

   read(pilihan)

   case (pilihan):
      1:write("Anda memilih menu ke-1")
      2:write("Anda memilih menu ke-2")
      3:write("Anda memilih menu ke-3")
      4:write("Anda memilih menu ke-4")
      5:write("Keluar program")
      otherwise:write("tidak ada menu ini")
   end case
until pilihan=5
```

Aktivitas Belajar 4-11

Lakukan modifikasi pada algoritma menentukan kuadran sebuah titik pada koordinat kartesius sebelumnya dengan menampilkan pilihan berulang sampai memilih selesai.

7. Prosedur dan Fungsi

Apa yang yang terlintas di pikiran kalian jika mendengar kata prosedur? Ya, kata ini pasti tidak asing di telinga kalian. Di sekolah, kalian berhadapan dengan prosedur, di tempat layanan umum juga berhubungan dengan prosedur. Prosedur meminjam buku, prosedur mendaftar, prosedur mengganti buku tabungan, prosedur memesan barang dan masih banyak lagi yang lainnya. Prosedur merupakan serangkaian kegiatan yang dilakukan satu per satu.

Dalam pemrograman juga berlaku hal yang sama. Prosedur merupakan modul dalam pemrograman yang mengerjakan tugas tertentu dengan langkah-langkah tertentu yang nantinya dapat digunakan pada banyak aktivitas yang sama. Misalkan kalian harus mengurutkan data di beberapa bagian program, kalian tidak harus menuliskan algoritma pengurutan pada setiap bagian. Kalian dapat membuat sebuah prosedur pengurutan yang bisa digunakan di beberapa tempat.

Secara umum struktur **prosedur** mirip dengan struktur program, yaitu terdiri dari nama prosedur, deklarasi, dan deskripsi. Deklarasi prosedur ditandai dengan kata kunci "**procedure**" dengan struktur yang dituliskan sebagai berikut:

procedure nama_prosedur(parameter input)
{deskripsi prosedur}
Deklarasi
{deklarasi tipe} {deklarasi peubah} {deklarasi konstanta}
Deskripsi
{baca masukan} {melakukan proses} {menampilkan keluaran}

Dalam algoritma, prosedur dideskripsikan pada bagian deklarasi, tetapi dapat juga hanya dideklarasikan sedangkan deskripsi dibuat setelah deskripsi algoritma. Sebagai contoh misalkan pada algoritma segiempat, menghitung luas dan keliling dibuat menjadi prosedur. Jika deklarasi dan deskripsi dipisah, maka algoritmanya dapat ditulis seperti berikut.

```
Algoritma SegiEmpat
02  {menghitung luas dan keliling segi empat dalam
03  prosedur}
Deklarasi
05  {deklarasi peubah}
06  panjang, lebar: integer
07  {deklarasi prosedur}
08  procedure hitungLuas(p:integer,l:integer)
09  procedure hitungKeliling(p:integer,l:integer)
Deskripsi
11  {baca masukan}
12  read(panjang,lebar)
13  {melakukan proses}
14  hitungLuas(panjang,lebar)
15  hitungKeliling(panjang,lebar)
16
procedure hitungLuas(p:integer, l:integer)
18  {menghitus luas segi empat}
19  {parameter p digunakan untuk mewakili panjang,
20  sedangkan parameter l digunakan untuk mewakili
21  lebar}
Deklarasi
23  Ls:integer {variabel lokal untuk luas}
Deskripsi
25  Ls←p*l
26  write (Ls)
27
procedure hitungKeliling (p:integer, l:integer)
29  {menghitus luas segi empat}
30  {parameter p digunakan untuk mewakili panjang,
31  sedangkan parameter l digunakan untuk mewakili
32  lebar}
Deklarasi
34  Kll:integer {variabel lokal untuk keliling}
Deskripsi
36  Kll←2*(p+l)
37  write (Ls)
```

Selain bentuk penulisan tersebut, prosedur juga dapat langsung dideskripsikan tanpa dideklarasikan sebelumnya.

```
Algoritma SegiEmpat
02  {menghitung luas dan keliling segi empat dalam
03  prosedur}
Deklarasi
05  {deklarasi peubah}
06  panjang, lebar: integer
07  {deklarasi prosedur}
08  procedure hitungLuas(p:integer,l:integer)
09  deklarasi
10     Ls:integer
11  deskripsi
12     Ls←p*l
13     write(Ls)
14
15  procedure hitungKeliling(p:integer,l:integer)
16  deklarasi
17     Kll:integer
18  deskripsi
19     Kll←2*(p+l)
20     write(Kll)
Deskripsi
22  {baca masukan}
23  read(panjang,lebar)
24  {melakukan proses}
25  hitungLuas(panjang,lebar)
26  hitungKeliling(panjang,lebar
```

Dalam pemrograman, rangkaian kegiatan aksi-aksi yang terpisah juga dapat diwujudkan dalam **fungsi**. Pada dasarnya fungsi memiliki kesamaan dengan prosedur tetapi menghasilkan nilai keluaran. Bahkan dalam beberapa bahasa pemrograman hanya memiliki fungsi, tidak memiliki prosedur. Prosedur dianggap sebagai fungsi yang tidak menghasilkan nilai (*void*).

Struktur penulisan fungsi juga sama dengan prosedur, hanya saja kata kunci yang digunakan menjadi **function** dan di akhir ditambahkan pernyataan **return** untuk mengembalikan nilai. Secara lengkap dapat dituliskan seperti berikut.

```
function nama_fungsi(parameter input)→tipe luaran
    {deskripsi fungsi}
```

Deklarasi	
	{deklarasi tipe}
	{deklarasi peubah}
	{deklarasi konstanta}
Deskripsi	
	{baca masukan}
	{melakukan proses}
	return nilai

Cara menuliskan fungsi dalam algoritma juga sama dengan prosedur yaitu dapat dideklarasikan pada bagian deklarasi dan dideskripsikan setelah deskripsi algoritma selesai atau langsung dideskripsikan pada bagian deklarasi algoritma. Jika algoritma segi empat yang sebelumnya diubah menjadi fungsi, maka dapat dituliskan sebagai berikut.

```
Algoritma SegiEmpat
02  {menghitung luas dan keliling segi empat dalam
03  prosedur}
Deklarasi
05  {deklarasi peubah}
06  panjang, lebar, luas, keliling: integer
07  {deklarasi prosedur}
08  function hitungLuas(p:integer,l:integer):integer
09  function hitungKeliling(p:integer,l:integer):integer
Deskripsi
11  {baca masukan}
12  read(panjang,lebar)
13  {melakukan proses}
14  luas←hitungLuas(panjang,lebar)
15  keliling←hitungKeliling(panjang,lebar)
16  write(luas)
17  write(keliling)
18
function hitungLuas(p:integer, l:integer):integer
20  {menghitus luas segi empat}
21  {parameter p digunakan untuk mewakili panjang,
22  sedangkan parameter l digunakan untuk mewakili lebar}
Deklarasi
24  {tidak memerlukan}
Deskripsi
26  return p*l
27
```

```
function hitungKeliling(p:integer, l:integer):integer
29  {menghitus luas segi empat}
30  {parameter p digunakan untuk mewakili panjang,
31  sedangkan parameter l digunakan untuk mewakili lebar}
Deklarasi
33  {tidak memerlukan}
Deskripsi
35  return 2*(p+l)
```

Prosedur dan fungsi dalam pemrograman lebih banyak dipisahkan pada blok tersendiri sebagai sub program (*sub routine*). Hal ini dilakukan untuk memecah program yang rumit menjadi bagian-bagian yang sederhana. Prosedur dan fungsi juga bisa digunakan untuk menyembunyikan detail program dan memudahkan perubahan jika terjadi kesalahan.

Oleh karena prosedur dan fungsi merupakan sub program, maka penggunaan variabel juga menjadi berbeda. Ada variabel yang bisa digunakan pada seluruh bagian dari program (variabel global), ada juga yang hanya bisa digunakan pada prosedur atau fungsi saja (variabel lokal). Umumnya variabel global dideklarasikan pada algoritma sedangkan variabel lokal dideklarasikan pada bagian prosedur atau fungsi.

Aktivitas Belajar 4-12

Buatlah algoritma (*pseudocode*) untuk menghitung luas beberapa bangun datar (persegi panjang, bujur sangkar, segitiga, lingkaran) dan volume beberapa bangun ruang (balok, kubus, limas, kerucut). Tambahkan menu untuk memanggil masing-masing bangun.

Uji Kompetensi

Dalam uji kompetensi ini kalian harus membaca dengan cermat dan teliti setiap butir soal di bawah ini. Kerjakan sesuai dengan apa yang diminta pada soal. Lingkari jawaban yang dianggap paling benar!

1. Berikut ini adalah salah satu tujuan atau manfaat penggunaan DBMS, kecuali ....
   a. Akses data lebih mudah dan cepat
   b. Dapat menangani data-data dalam jumlah besar

c. Menghilangkan hasil duplikasi maupun inkonsistensi data
d. Membuat pekerjaan menjadi kompleks
e. Meningkatkan keamanan data

2. Perintah SQL yang digunakan untuk memodifikasi struktur tabel termasuk ke dalam ....
   a. *Data Control Language*
   b. *Data Manipulation Language*
   c. *Data Definition Language*
   d. NoSQL
   e. Tidak termasuk ke semuanya
3. Berikut ini yang bukan termasuk keuntungan penggunaan CASE adalah ....
   a. Peningkatan prosedur pengendalian
   b. Hemat biaya
   c. Peningkatan produktivitas
   d. Administrasi kompleks
   e. Peningkatan kualitas
4. Di bawah ini yang bukan merupakan fungsi Sistem Operasi adalah ....
   a. Mengatur sistem kerja pada komputer
   b. Mengatur penyimpanan
   c. Menjembatani antara pengguna dan program aplikasi
   d. Tempat meletakkan aplikasi
   e. Pusat pemrosesan seluruh perintah
5. Berikut ini yang bukan termasuk ke dalam Sistem Operasi *stand alone* adalah ....
   a. LINUX
   b. Windows
   c. Chrome OS
   d. FireFox OS
   e. MeeGO
6. Alasan umum menggunakan perangkat lunak manajemen aset adalah ....
   a. Kecepatan
   b. Kenyamanan

c. Pengorganisasian
d. Kepastian
e. A, B, dan C benar

7. Berikut yang termasuk perangkat lunak manajemen aset adalah ....
   a. Game Maker Studio, echo3D, Adobe Photoshop, Unreal Engine 4
   b. Unity Asset Store, Unreal 4's, RPG Maker MV's File System, ioMoVo
   c. Game Maker Studio, ioMoVo, Modo, 3DS Max Design
   d. Modo, 3DS Max Design, RPG Maker MV's File System, Game Maker Studio
   e. Unreal Engine 4, Game Maker Studio, Maya, Audacity
8. Berikut ini yang bukan termasuk jenis antarmuka adalah ....
   a. *Direct Manipulation*
   b. *Touch User Interface*
   c. *Natural Language User Interface*
   d. *Menu-Driven User Interface*
   e. *Command Line Interface*
9. Warna yang digunakan untuk tampilan berjenis ....
   a. *Additional*
   b. *Addictive*
   c. *Additive*
   d. *Substractive*
   e. *Substantive*
10. Kejelasan (*legibility*) biasanya dipengaruhi oleh ....
    a. Jenis huruf, warna, ukuran
    b. Warna latar belakang, kerumitan desain, jenis huruf
    c. Frekuensi kemunculan, warna, kerumitan desain
    d. Frekuensi, ukuran, kerumitan
    e. Latar belakang, warna, ukuran
11. Algoritma dapat dinotasikan dalam bentuk ....
    a. Cerita, gambar, nada
    b. Cerita, kode program, *flowchart*
    c. *Flowchart*, deskriptif, *pseudocode*
    d. *Pseudocode*, *flowchart*, cerita
    e. Deskriptif, gambar, cerita

12. Struktur dasar algoritma terdiri dari ....
    a. Runtutan, pemilihan, pengulangan
    b. *Header*, deklarasi, deskripsi
    c. *Flowchart*, deskriptif, *pseudocode*
    d. Tipe Data, Variable, Konstanta
    e. *Input*, process, *output*
13. Perhatikan algoritma berikut ini:
    1) Buka aplikasi ojek online
    2) Mengaktifkan GPS
    3) Input lokasi penjemputan
    4) Input lokasi tujuan
    5) Memilih metode pembayaran
    6) Pesan lalu menunggu hingga ojol tiba
    7) Menumpang ojol
    8) Mengendarai ojol sampai di tujuan
    9) Turun
    10) Membayar layanan ojol

    Kegiatan pada poin (8) jika dalam algoritma termasuk dalam ....
    a. Inisialisasi
    b. Proses
    c. Pemilihan
    d. *Input*
    e. *Output*
14. Perhatikan alur memilah sampah berikut
    1) Memilah sampah sesuai jenisnya.
    2) Pertama, organik.
    3) Sampah organik diolah jadi pupuk.
    4) Kedua, sampah yang masih bisa digunakan lagi.
    5) Sampah ini bisa dimanfaatkan untuk kegunaan lain.
    6) Ketiga, sampah yang bisa didaur ulang.
    7) Sampah ini bisa dimanfaatkan dan diolah jadi benda lain.
    8) Bila sampah tidak masuk kategori organik, bisa digunakan lagi, dan tidak dapat didaur ulang, maka buanglah ke TPA.

Struktur pemilihan ditunjukkan pada langkah nomor ....

a. 1, 3, 5, 7
b. 2, 4, 6
c. 2, 4, 6, 8
d. 3, 5, 7, 8
e. Tidak ada pemilihan

15. Diketahui sebuah persamaan dalam algoritma C=A+B*2/4, jika A=6, B=8, berapakah nilai C?

a. 7
b. 10
c. 14
d. 6
e. 8

Pengayaan

1. Buku

- Fatansyah. (2015). Basis Data. Revisi Kedua. Bandung: Informatika Bandung.
- Furman, B. J. (2010). *Lecture Note on Algorithms, Pseudocode, and Flowcharts*. San Jose: SAN JOSÉ STATE UNIVERSITY.
- Gupta, S. B., & Mittal, A. (2017). *Introduction To Database Managenent System, Second Edition*. New Delhi: Laxmi Publications (P) Ltd.
- Knuth, D. E. (2013). *The Art of Computer Programmning. Volume 1. Fundamental Algorithms. Third Edition*. Digital Release. Boston: Addison Wesley.
- Mayhew, D. J. (2008). *Principles and Guidelines in Software User Interface Design*. Pearson.
- Mayhew, D. J. (2008). *User Eficiency: Evaluation And Design*. West Tisbury, MA, USA: Deborah J. Mayhew & Associate.
- Munir, R., & Lidya, L. (2016). Algoritma dan Pemrograman Dalam Bahasa Pascal, C, C++. Edisi Keenam. Bandung: Informatika Bandung.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concept, Seventh Edition*. New York: McGrawHill.

- Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating System*. Fourth Edition. New Jersey: Pearson Education, Inc.
- White, A. W. (2011). *The Elements of Graphic Design, Second Edition*. Allworth.
- Wirth, N. (2004). *Algorithms and Data Structures. Oberon version*. Updated 2012. New Jersey: Prentice Hall, Inc.

2. Artikel

- Aprilia, P. (2022, September 6). Mengenal User Interface: Pengertian, Kegunaan, dan Contohnya. https://www.niagahoster.co.id/blog/user-interface/.
- Ariffudin, M. (2022, April 21). Apa itu DBMS? Pengertian, Fungsi, Kelebihan, Macam-macam DBMS. https://www.niagahoster.co.id/blog/dbms-adalah/.
- Guntoro. (2022, Februari 2). 12 Framework untuk Pengembangan Web Terbaik. https://badoystudio.com/framework-untuk-pengembangan-web/
- I/O Software Inc. (2022, November 11). What's The Best Game Asset Management Software? https://iomovo.io/what-is-the-best-game-asset-management-software/.
- Kumparan. (2022, Juni 13). Mengenal Apa itu iOS dan Fungsinya. https://kumparan.com/berita-update/mengenal-apa-itu-ios-dan- fungsinya-1yGPvIfdKHx/full.
- Maulana, G. (2017). Pembelajaran Dasar Algoritma dan Pemrograman Menggunakan El-Goritma Berbasis Web. Jurnal Teknik Mesin (JTM), Vol 06, 69-73.
- Moreno, L. (2020, Mei 12). Fundamentals of layout in user interface design (UI): Composition, balance, and how to manage a good structure. https://uxdesign.cc/fundamentals-of-layout-in-interface-design-ui-3a9dba31f1.
- pp_pankaj. (2023, January 3). Computer Aided Software Engineering (CASE). https://www.geeksforgeeks.org/computer-aided-software-engineering- case/.
- Rehman, J. (2019, 09). Advantages and disadvantages of android operating system. https://www.itrelease.com/2019/09/advantages-and-disadvantages-of- android-operating-system/
- Volle, A. (2023, January 16). iOS Operating System. https://www.britannica.com/topic/iOS.

- Wahyudi, S. E. (2020, 02 12). Teori Warna (Multimedia #4). https://informatika.uc.ac.id/id/2020/02/teori-warna-multimedia-4/#.
- Western Governors University. (2021, April 1). 5 Most Popular Operating Systems. https://www.wgu.edu/blog/5-most-popular- operating-systems1910.html#close.

Refleksi

Berilah tanda (√) pada kotak yang dianggap sesuai! Setelah mempelajari bab ini, bagaimanakah penguasaankalian terhadap materi-materi berikut?

No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Basis Data			
2.	Tools Pengembangan Perangkat Lunak			
3.	Ragam Sistem Operasi			
4.	Pengelolaan Aset			
5.	Antarmuka Pengguna			
6.	Dasar Algoritma Pemrograman			

3. Dari materi-materi tersebut, bagian manakah yang paling Kalian sukai? Mengapa ?
4. Apa manfaat yang Kalian dapatkan setelah mempelajari materi bab ini untuk kehidupan sehari-hari?
5. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

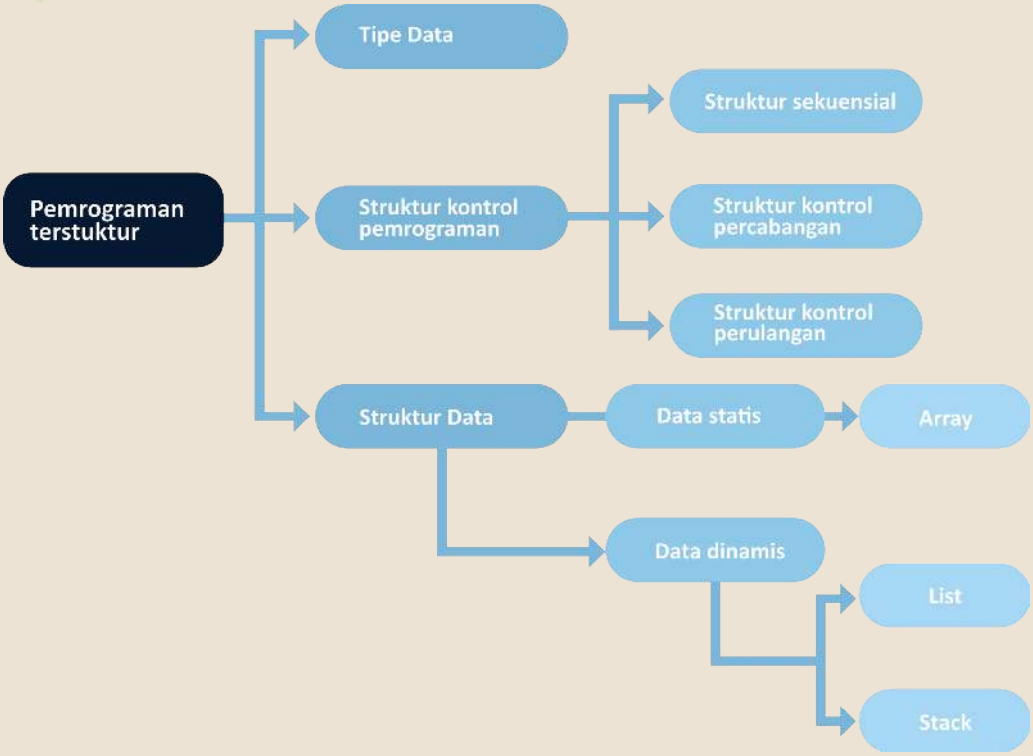
BAB 5

Pemrograman Terstruktur

Tujuan Pembelajaran

Setelah kalian mempelajari materi ini, diharapkan kalian dapat melakukan pemrograman terstruktur dengan penggunaan tipe data, struktur sekuensial, struktur kontrol percabangan, struktur kontrol perulangan, dan struktur data.

Peta Materi



Kata Kunci

♦ *Array* ♦ *Sort* ♦ *List* ♦ *Stack* ♦ *While* ♦ Variabel ♦ Konstanta ♦ Integer ♦ Sekuensial ♦ *Decision* ♦ *Queue* ♦ *Looping*

Gambar 5.1 Penyewaan mobil
Sumber: Marwondo, Rini Melati, dan Dana R. N. Adnan/2023

Apakah kalian tahu data apa saja yang dibutuhkan untuk menghitung total biaya sewa mobil dalam waktu tertentu?

Apersepsi

Apakah kalian pernah punya masalah dalam perhitungan matematika? Apa yang kalian pikirkan dan lakukan terhadap masalah tersebut? Diabaikan saja ataukah dicari solusinya?

Ya, tentu saja kalian akan berpikir untuk mencari solusi menyelesaikan masalah tersebut. Salah satu contoh masalah yang kalian hadapi dalam perhitungan matematika adalah menyelesaikan rumus matematika, seperti menghitung luas sebuah bangun datar. Misalnya menghitung luas segitiga.

Untuk menghitung segitiga, tentu kalian akan berpikir apa saja yang akan jadi input, proses, dan *output*-nya? Kalian harus menentukan dulu urutan pengerjaannya. Mulai dari memasukkan panjang kemudian lebarnya. Setelah itu dilakukan proses perhitungan rumusnya, baru kemudian akan muncul *output*-nya, yaitu luas segitiga.

Nah, langkah penyelesaian masalah yang kalian lakukan di atas adalah langkah pengerjaan yang bisa dilakukan dalam pemrograman. Pemrograman terstruktur merupakan suatu padadigma yang membagi kode program menjadi modul atau fungsi.

Pada bab sebelumnya kalian sudah mengenal dan mempelajari tentang agoritma pemrogrman itu dan cara pengerjaannya. Dan dalam bab ini kalian akan mempelajari lebih jauh tentang bagaimana mengimplementasikan algoritma tersebut, langkah-langkah dan intruksi-intruksi membentuk struktur program diantaranya struktur sekuensial, percabangan dan pengulangan serta akan dibahas juga tentang tipe data dan operator, operasi aritmatika juga struktur data nya. Kode program yang diimplementasikan dalam buku ini menggunakan bahasa pemrograman C++. Kalian dapat menggunakan integrated development environment yang populer seperti Netbeans, Visual Studio Code, dan lain-lain untuk mempraktekkannya.

A. Penggunaan Tipe Data dan *Identifier*

Aktivitas Belajar 5-1

Pada bagian ini kalian akan diminta melakukan analisis kode program berikut ini, kemudian kalian cari tahu tipe data, variabel, dan konstanta apa saja yang ada dalam kode program ini!

```
#include <iostream>
#include <conio.h>
using namespace std;
void main(){
    char huruf_1 = 'C', huruf_2 = '+';
    cout << "Tipe Data char pada " << huruf_1 << huruf_2 << huruf_2;
    getch();

}
```

```
#include <iostream>
using namespace std;


int main(){

    int a[12] = {1,3,5,4,7,2,99,16,45,67,89,45};
    int indeks, total = 0;
    for (indeks = 0; indeks <= (12-1); indeks++){
    total = a[indeks];
    cout << "Total setiap elemen array  adalah " << total << endl;
    }
    return 0;
}
```

Setelah kalian melakukan analisis kode program di atas, lengkapilah tabel berikut ini:

Tipe data	Variabel	Konstanta
.....	.....	.....
.....	.....	.....
.....	.....	.....

1. Tipe Data

Pada bab sebelumnya kalian sudah mempelajari tentang tipe data dan bagaimana pembuatan algoritmanya. Nah, saat ini kalian akan mempelajari bagaimana mengaplikasikan atau menerapkan algoritma tersebut ke dalam sebuah program.

a. Bilangan Bulat

Berikut ini adalah contoh kode program dengan tipe data bilangan bulat.

```
#include <iostream>
using namespace std;
int main(){
    int a, b, c, d;
    a = 10 + 4 * 2 /4;
    b = (10 + 4) * 2 /4;
    c = 20 % 3 * 2;
    d = 20 % (3 * 2);
    cout << "a = 10 + 4 * 2 /4" << endl;
    cout << "b = (10 + 4) * 2 /4" << endl;
    cout << "c = 20 % 3 * 2" << endl;
    cout << "d = 20 % (3 * 2)" << endl << endl;
    cout << "Hasil dari a = " << a << endl;
    cout << "Hasil dari b = " << b << endl;
    cout << "Hasil dari c = " << c << endl;
    cout << "Hasil dari d = " << d << endl;

    return 0;
}
```

Variabel yang menggunakan tipe data bilangan bulat yaitu **a**, **b**, **c**, dan **d**. tipe datanya adalah ***int***.

b. Bilangan Riil

Berikut ini adalah contoh kode program dengan tipe data bilangan Riil.

```
#include <iostream>
using namespace std;

int main(){
    float a, b, c, d, e, f, g, h;

    cout << "Masukkan Nilai a = "; cin >> a;
    cout << "Masukkan Nilai b = "; cin >> b;
    cout << "Masukkan Nilai c = "; cin >> c;

    d = a + 6 > 12;
    e = b > 4 + a;
    f = c - 3 <= 8;
    g = d || e || f;

    cout << "\n === Program Ekspresi OR == ";
    cout << "\n Hasil dari d = a + 6 > 12 adalah " << d;
    cout << "\n Hasil dari e = b > 4 + a adalah " << e;
    cout << "\n Hasil dari f = c - 3 <= 8 adalah " << f;
    cout << "\n Hasil dari g = d || e || f adalah " << g;

    return 0;
}
```

Variabel yang menggunakan tipe data bilangan bulat yaitu **a**, **b**, **c**, **d**, **e**, **f**, **g**, dan **h**. tipe datanya adalah ***float***

c. Karakter

Kode program berikut ini merupakan program yang menggunakan tipe data karakter:

```
#include <iostream>
#include <conio.h>
using namespace std;
void main(){
    char huruf_1 = 'C', huruf_2 = '+';
    cout << "Tipe Data char pada " << huruf_1 << huruf_2 << huruf_2;
    getch();

}
```

Variabel yang menggunakan tipe data karakter yaitu **huruf_1** tipe datanya adalah ***char***.

d. *Boolean*

Tipe data ini hanya memberikan pernyataan benar atau salah saja, true atau false, dan biasanya nilai yang diisi pun hanya true atau false saja. Tipe data ini sering digunakan sebagai penanda apakah suatu proses itu sudah selesai dilakukan atau belum. Contoh X bernilai true, Y bernilai false.

Kode program berikut ini merupakan program yang menggunakan tipe data karakter:

```
#include <iostream>

using namespace std;

int main()
{
  bool a = true;
  bool b = false;

  cout << "a = "  << a << endl;
  cout << "b = "  << b << endl;

  return 0;
}
```

Variabel yang menggunakan tipe data ***boolean*** yaitu **a** dan **b**.

2. Operator

Saat mendengar kata operator, apa yang ada di pikiran kalian? Apakah kalian berpikir operator itu adalah orang yang mengoperasikan suatu mesin atau alat? Dalam pemrograman, operator itu adalah sebuah simbol yang berada di antara dua buah operan dan bisa menghasilkan suatu hasil. Contohnya seperti ini. Dalam matematika jika tanda ("/") kalian letakkan di antara dua bilangan, maka akan menghasilkan angka lain sebagai hasil dari pembagian kedua angka tersebut. Nah, tanda bagi itulah yang dinamakan operator. Operator ini nantinya akan kalian gunakan untuk melakukan operasi pada nilai yang ada dalam sebuah variabel. Proses operasi yang akan dilakukan di sini di antaranya perhitungan, perbandingan, dan juga relasi. Lalu operator apa saja yang ada dalam pemrograman ini? Dalam bab ini, kalian akan diberikan penjelasan tentang operator berikut:

a. Operator Aritmatika, operator ini biasanya digunakan dalam matematika seperti pembagian, pengurangan, penjumlahan. Contoh: **8-7, 10+45, m = m+1, x/y**

   Tabel berikut ini menunjukkan apa saja yang termasuk dalam operator aritmatika:

OPERATOR	JENIS OPERASI	CONTOH
+	Penjumlahan	2+5=5
-	Pengurangan	5-3=2
*	Perkalian	2*3=6
/	Pembagian	10.0/3.0=3.3333
%	Sisa bagi	10%3=1

Tabel 5.1 Operator aritmatika

Berikut ini contoh kode program yang di dalamnya ada operasi aritmatikanya:

```
#include <iostream>
using namespace std;
int main(){
    int a, b, c, d;
    a = 10 + 4 * 2 /4;
    b = (10 + 4) * 2 /4;
    c = 20 % 3 * 2;
    d = 20 % (3 * 2);
    cout << "a = 10 + 4 * 2 /4" << endl;
    cout << "b = (10 + 4) * 2 /4" << endl;
    cout << "c = 20 % 3 * 2" << endl;
    cout << "d = 20 % (3 * 2)" << endl << endl;
    cout << "Hasil dari a = " << a << endl;
    cout << "Hasil dari b = " << b << endl;
    cout << "Hasil dari c = " << c << endl;
    cout << "Hasil dari d = " << d << endl;

    return 0;
}
```

Ini adalah output dari kode program di atas:

```
a = 10 + 4 * 2 /4
b = (10 + 4) * 2 /4
c = 20 % 3 * 2
d = 20 % (3 * 2)

Hasil dari a = 12
Hasil dari b = 7
Hasil dari c = 4
Hasil dari d = 2
```

b. Operator Relasi. Biasanya operator ini digunakan untuk melakukan perbandingan dari dua buah nilai. Contoh: **S < T, A > B, X == C, C>=D, L<>M**.

Tabel berikut ini menunjukkan apa saja yang termasuk dalam operator relasi:

OPERATOR	JENIS OPERASI	CONTOH
>	Lebih besar	(5>2)=1
<	Lebih kecil	(5<2)=0
>=	Lebih besar atau sama dengan	(5>=5)=1
<=	Lebih kecil atau sama dengan	(5<=2)=0
==	Sama dengan	(5==2)=2
!=	Tidak sama dengan	(5!=2)=1

Tabel 5.2 Operator relasi

Berikut adalah kode program untuk operator relasi:

```
#include <iostream>
using namespace std;

int main(){
    float a, b, c, d, e, f, g, h;

    cout << "Masukkan Nilai a = "; cin >> a;
    cout << "Masukkan Nilai b = "; cin >> b;
    cout << "Masukkan Nilai c = "; cin >> c;

    d = a + 6 > 12;
    e = b > 4 + a;
    f = c - 3 <= 8;
    g = d || e || f;

    cout << "\n === Program Ekspresi OR == ";
    cout << "\n Hasil dari d = a + 6 > 12 adalah " << d;
    cout << "\n Hasil dari e = b > 4 + a adalah " << e;
    cout << "\n Hasil dari f = c - 3 <= 8 adalah " << f;
    cout << "\n Hasil dari g = d || e || f adalah " << g;

    return 0;
}
```

Berikut ini output dari kode program di atas:

```
Masukkan Nilai a = 7
Masukkan Nilai b = 3
Masukkan Nilai c = 5

=== Program Ekspresi OR ===
Hasil dari d = a + 6 > 12 adalah 1
Hasil dari e = b > 4 + a adalah 0
Hasil dari f = c - 3 <= 8 adalah 1

Hasil dari g = d || e || f adalah 1Press any key to continue . . .
```

c. Operator Logika *Boolean*, biasanya operator ini digunakan untuk melogikakan dua atau lebih kondisi. Bentuk operator *Boolean* ini di antaranya AND, OR, NOT. Contoh menggunakan AND:

(L > M) && (X <> 10). Contoh menggunakan OR: **(L > M) || (X <> 10)**, sedangkan contoh penggunaan NOT: **! (P==R)**.

Tabel berikut ini menunjukkan apa saja yang termasuk dalam operator logika:

OPERATOR	JENIS OPERASI	CONTOH
&&	AND (dan)	1 && 1 = 1
||	OR (atau)	1 || 0 = 1
!	NOT (negasi)	!0 = 1

Tabel 5.3 Operator logika *boolean*

Berikut ini adalah contoh kode program untuk operator Logika *Boolean* AND:

```
#include <iostream>

using namespace std;

int main(){
      float a, b, c, d, e, f, g, h;

      cout<<" Masukkan Nilai a = ";cin>>a;
      cout<<" Masukkan Nilai b = ";cin>>b;
      cout<<" Masukkan Nilai c = ";cin>>c;

      d = a + 6 > 12;
      e = b > 4 + a ;
      f = c - 3 <= 8;
      g = d && e && f;

      cout<<"\n === Program Ekspresi AND ===";
      cout<<"\n Hasil dari d = a + 6 > 12 adalah "<<d;
      cout<<"\n Hasil dari e = b > 4 + a adalah " <<e;
      cout<<"\n Hasil dari f = c - 3 <= 8 adalah " <<f;
      cout<<"\n\n Hasil dari g = d && e && f adalah " <<g << endl;

      return 0;
}
```

Berikut *output* dari kode program di atas:

```
Masukkan Nilai a = 7
Masukkan Nilai b = 3
Masukkan Nilai c = 5

=== Program Ekspresi AND ===
Hasil dari d = a + 6 > 12 adalah 1
Hasil dari e = b > 4 + a adalah 0
Hasil dari f = c - 3 <= 8 adalah 1

Hasil dari g = d && e && f adalah 0
```

Berikut ini adalah contoh kode program dengan operator logika NOT:

```cpp
#include <iostream>

using namespace std;

int main(){
      int a, b, c;

      a = 2;

      b = (a + 8 < 12);
      c = !(b);

      cout<<" === Program Ekspresi NOT ===";

      cout<<"\n Nilai a = "<<a;
      cout<<"\n Nilai b = (a + 8 < 12) = "<<b;
      cout<<"\n Nilai c = !(b) = "<<c << endl;

      return 0;
}
```

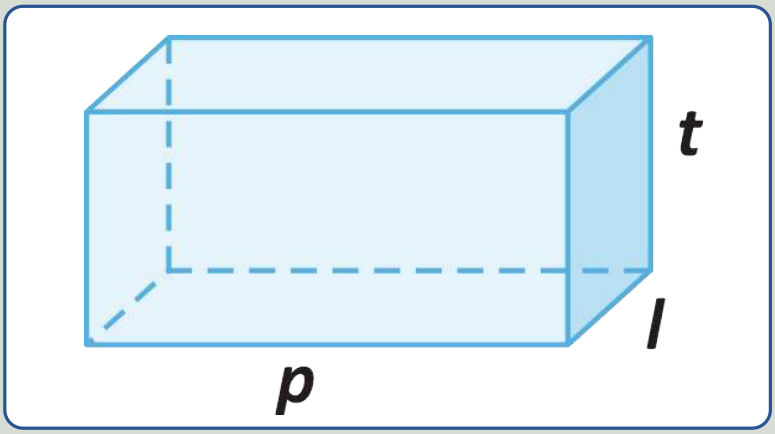
Coba kalian perhatikan kode program di atas. Variabel yang digunakan dalam kode di atas adalah **a, b, dan c** yang bertipe **int** atau *integer*. Kemudian kalian bisa melihat ada operator assignment atau operator penugasan dalam kode tersebut yaitu **a=2**. Pernyataan **a=2** ini akan menyalin nilai integer 2 pada variabel **a**. Tugas operasi ini selalu berlangsung dari kanan ke kiri dan tidak pernah sebaliknya. Kemudian pernyataan **b=(a+8<12)** akan mendapatkan nilai dari penjumlahan 8 ditambah a yang sudah memiliki nilai 2, maka b mendapatkan nilai 10 dari penjumlahan 2+8. Kemudian nilai 10 ini akan dibandingkan dengan 12 dan hasilnya adalah true atau 1. Kemudian pernyataan c =!(b) artinya c NOT b, maka hasilnya adalah 0.

Jika kalian menjalankan kode program di atas, maka akan tampak *output* seperti berikut:

```
=== Program Ekspresi NOT ===
Nilai a = 2
Nilai b = (a + 8 < 12) = 1
Nilai c = !(b) = 0
```

B. Struktur Sekuensial

Aktivitas Belajar 5-2



Gambar 5.2 Luas permukaan balok
Sumber: Marwondi, Rini Melati, & Dana R. N. Adnan/2023v

Misalkan kalian diminta untuk membuat algoritma dan program untuk menghitung luas dan volume balok. Adapun rumus luas permukaan balok adalah LuasPermukaan=(2*p*l)+(2*p*t)+(2*l*t), sedangkan volume balok adalah volume=p*l*t.

Nah, coba kalian rumuskan langkah-langkah penyelesaian program untuk kasus di atas kemudian diskusikanlah hasil rumusan kalian di depan kelas.

1. Struktur Sekuensial

Rangkaian instruksi dalam pembuatan program yang diproses secara berurutan, satu per satu, mulai dari instruksi pertama sampai instruksi terakhir dikenal dengan nama struktur program sekuensial.

Dalam strukur sekuensial ini instruksi program dikerjakan tepat sekali, dan tidak ada instruksi yang dikerjakan secara berulang. Instruksi terakhir yang dikerjakan merupakan akhir dari suatu program.

2. Karakteristik dan Cara Kerja Struktur Sekuensial

Dalam paragraf sebelumnya sudah disampaikan seperti apa struktur sekuensial. Coba kalian lihat kode program berikut ini:

```cpp
#include <iostream>

using namespace std;

int main(){
    int alas, tinggi;
    float luas;
    cout << "==============================" << endl;
    cout << "Program Luas Segitiga" << endl;
    cout << "==============================" << endl;
    cout << "Masukkan Alas : " ; cin >> alas;
    cout << "Masukkan Tinggi : " ; cin >> tinggi;

    luas = alas * tinggi * 0.5;
    cout << "Luas Segitiga : " << luas << endl;
    cout << "==============================" << endl;

}
```

Kode program di atas adalah program untuk mencari luas segitiga. Dari program di atas kalian bisa melihat karakteristik dari struktur sekuensialnya, yaitu urutan program atau instruksinya hanya dikerjakan satu kali. Coba perhatikan instruksi ”Masukkan Alas:”, instruksi ini hanya dimasukkan satu kali dan tidak diulang. Kemudian coba kalian perhatikan instruksi ”Masukkan tinggi:”. Ini artinya bahwa dalam struktur sekuensial setiap instruksi dilakukan secara berurutan satu per satu.

Jika kode program di atas dijalankan maka akan tampak output seperti berikut ini:

```
==============================
Program Luas Segi Tiga
==============================
Masukkan Alas :6
Masukkan Tinggi :8
Luas Segitiga :24
==============================
```

Contoh program lain yang menggambarkan struktur sekuensial dapat kalian lihat pada kode program berikut:

```cpp
#include <iostream>
using namespace std;

int main(){
    int jam, menit, detik;

    cout << "==============================" << endl;
    cout << "Program Konversi Detik ke Jam" << endl;
    cout << "==============================" << endl;
    cout << "Detik yang akan dihitung : " ; cin >> detik;

    jam = detik / 3600;
    menit = (detik % (jam * 3600)) / 60;
    detik = detik - (3600 * jam) - (60 * menit);

    cout << "Waktu yang telah dikonversei : " << endl;
    cout << "Jam : " << jam << endl;
    cout << "Menit : " << menit << endl;
    cout << "Detik : " << detik << endl;
    cout << "==============================" << endl;


}
```

Sama halnya seperti kode program mencari luas segitiga sebelumnya yang kalian pelajari, kode program ini juga menunjukkan karakteristik dari struktur program sekuensial. Di antaranya adalah tidak ada instruksi program yang dilakukan berulang.

Jika kode programnya dijalankan maka akan tampak output seperti berikut:

```
==============================
Program Konversi Detik ke Jam
==============================
Detik yang akan dihitung : 3728
Waktu setelah dikonversi :
Jam : 1
Menit : 2
Detik : 8
==============================
```

Namun perlu kalian ketahui, dalam struktur sekuensial, ada kode program yang jika letaknya berbeda akan mempengaruhi output, tetapi ada juga kode program yang jika dipindah letaknya, maka output-nya akan sama. Nah, coba kalian eksplorasi lebih jauh tentang karakteristik ini, ya.

C. Struktur Kontrol Percabangan

Aktivitas Belajar 5-3

Pada bagian ini kalian akan diminta menganalisis kode program berikut:

```
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    int nilai;
    cout <<"Masukkan sebuah nilai : ";cin>> nilai;

    if (nilai>70){
        cout <<"Lulus \n";
    }
    else{
    cout<<"Tidak Lulus \n";
    }
}
```

Apa output dari kode program di atas jika nilai yang diinputkan adalah 70?

1. Struktur Kontrol Percabangan

Selain struktur kontrol sekuensial, kalian juga akan menemukan dan mempelajari tentang struktur kontrol percabangan. Mengapa? Karena program yang kalian buat tidak akan selalu berurutan seperti dalam struktur sekuensial. Ada kalanya kalian akan dihadapkan pada permasalahan program yang mengandung pemilihan.

Contoh nyata dalam kehidupan sehari-hari kalian, program yang berhubungan dengan pemilihan yaitu adanya pemberian diskon atau potongan harga jika pembelian di atas Rp100.000, misalnya. Nah, contoh permasalahan ini tidak bisa kalian selesaikan dengan struktur sekuensial, tetapi harus menggunakan struktur kontrol percabangan.

2. Karakteristik dan Cara Kerja Struktur Kontrol Percabangan

Dalam struktur kontrol percabangan ini, suatu pernyataan yang ada dalam blok percabangan akan dikerjakan atau diproses jika kondisi atau syarat yang ditentukan bernilai benar, sedangkan jika kondisi atau

syarat tidak sesuai atau bernilai tidak benar maka pernyataan dalam blok percabangan tersebut tidak akan dikerjakan atau diproses. Bentuk percabangan yang akan dibahas dalam buku ini yaitu IF... THEN dan SWITCH... CASE.

3. Percabangan Sederhana atau Satu Kondisi

Dalam struktur ini kalian bisa memahaminya dengan melihat bentuk IF berikut ini:

```
IF (kondisi)
        Pernyataan
```

Jika kondisi benar maka blok pernyataan akan dikerjakan, sedangkan jika kondisi salah maka langsung meloncat mengerjakan instruksi selanjutnya yang berada di luar blok IF.

Coba kalian perhatikan contoh berikut ini:

"Jika kalian melakukan pembelian dengan nominal lebih dari 10.000, maka kalian akan mendapatkan potongan atau discount sebesar 10%."

Contoh kasus di atas jika kalian buatkan kode program percabangannya, maka akan tampak seperti berikut:

```
discount=0;
if (nilai_beli>10000)
        discount=(10/100)*nilai_beli;
```

Melihat kode program di atas, kalimat "**discount=(10/100)*nilai_beli**" adalah pernyataan yang akan dikerjakan atau diproses jika kondisi "**nilai_beli>10000**" bernilai **benar**. Penulisan pernyataan seperti dalam blok IF di atas bisa kalian buat lebih dari 1, ya, pernyataan dalam blok IF itu bisa banyak atau majemuk. Coba kalian lihat bentuk IF berikut:

```
IF (kondisi)
        {
        Pernyataan1
        Pernyataan2
            ...
            ...
            ...
            pernyataann
    }
```

Jika dibuat kode programnya, maka kalian bisa melihat hasil berikut:

```
Discount=0;
If (nilai_beli>10000)
{
    Discount=(10/100)*nilai_beli;
    Bonus="Payung"
}
```

Di mana ada 2 pernyataan yang akan dikerjakan jika kondisi IF-nya bernilai benar, yaitu pernyataan "**Discount=(10/100)*nilai_beli**" dan pernyataan "**Bonus="Payung**".

Sudah paham, ya, penjelasan tentang percabangan sederhana ini. Silakan kalian bisa mencari tahu lebih banyak lagi contoh-contoh programnya dan kalian bisa latihan membuat program percabangan sederhana tersebut agar pemahaman kalian semakin mendalam.

4. Percabangan Dua Kondisi

Sekarang kita akan bahas tentang percabangan dua kondisi. Seperti apa ya percabangan dua kondisi ini? Dalam percabangan 2 kondisi, jika kondisi bernilai salah atau tidak terpenuhi maka akan ada pernyataan atau statemen lain yang dikerjakan atau diproses. Coba kalian perhatikan struktur berikut:

```
IF (kondisi)
        Pernyataan1
Else
        Pernyataan2
```

Artinya jika kondisi bernilai benar atau terpenuhi, maka pernyataan pertamalah yang akan dikerjakan, jika tidak maka pernyataan kedualah yang akan dikerjakan. Paham, ya? Berikut contoh dari percabangan dua kondisi ini: "**Kalian akan mendapatkan predikat lulus jika nilai pelajaran kalian lebih dari 70, jika tidak maka kalian dinyatakan tidak lulus**".

Jika contoh kasus di atas kalian buatkan kode programnya, maka akan tampak seperti berikut ini:

```
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    int nilai;
    cout <<"Masukkan sebuah nilai : ";cin>> nilai;

    if (nilai>70){
        cout <<"Lulus \n";
    }
    else{
    cout<<"Tidak Lulus \n";
    }
}
```

Dan jika kalian memasukkan angka 90 pada program di atas, maka akan tampak output seperti berikut:

```
masukkan nilai : 90
lulus
Press any key to continue . . .
```

Sama halnya seperti percabangan sederhana, dalam percabangan dua kondisi ini juga bisa dikerjakan pernyataan majemuk ya. seperti bentuk IF berikut:

```
IF (kondisi)
        {
        Pernyataan1
        Pernyataan2
            ...
            ...
            ...
            Pernyataan_n
        }
Else
        {
        Pernyataan4
        Pernyataan5
            ...
            ...
            ...
            Pernyataan_n
        }
```

Berikut ini adalah contoh kasus percabangan dua kondisi dengan pernyataan majemuk yang bisa kalian pahami:

"Saat kalian melakukan pembelian lebih atau sama dengan nominal 1.000.000, maka akan mendapatkan diskon 10% untuk total pembelian

kalian. Sedangkan jika pembelian yang kalian lakukan kurang dari 1.000.000 maka diskon yang di dapatkan untuk total pembelian kalian adalah 5%".

Jika kalian implementasikan kasus di atas, maka kode programnya akan tampak seperti berikut:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    float beli,disc,total;
        cout<<"masukkan pembelian : "; cin>>beli;

    if (beli>=100000)
        {disc=0.1*beli;
        total=beli-disc;}
    else
        {disc=0.05*beli;
        total=beli-disc;}

    cout<<"jadi pembelian adalah : "<<beli<<endl;
    cout<<"jadi discount adalah : "<<disc<<endl;
    cout<<"jadi total pembayaran adalah : "<<total<<endl;

return 0;

}
```

Bila program di atas dijalankan, maka hasilnya adalah sebagai berikut:

```
masukkan pembelian : 2000000
jadi pembelian adalah : 2000000
jadi discount adalah : 200000
jadi total pembayaran adalah : 1800000
Press any key to continue . . .
```

Nah, sekarang coba kalian cari tahu lebih banyak lagi tentang percabangan dua kondisi ini, kemudian buat kode programnya, ya, agar kalian lebih paham lagi.

5. Percabangan Banyak Kondisi

Dalam struktur IF ini, ada banyak kondisi yang harus diperiksa dalam satu kali pemeriksaan untuk bisa menjalankan suatu pernyataan yang ada dalam blok IF-nya. Bentuk dari percabangan ini adalah seperti berikut:

```
if (kondisi)
{
... perintah;
... perintah;
}
else if (kondisi)
{
... perintah;
... perintah;
}
else
{
... perintah;
... perintah;
}
```

Contoh penerapan percabangan banyak kondisi ini, bisa kalian lihat pada kasus berikut:

"Kalian diminta untuk menentukan apakah angka atau bilangan yang kalian masukkan itu termasuk bilangan positif, bilangan negatif, ataukah bilangan nol. Misalnya bilangan yang kalian masukkan adalah X".

Nah, seperti apa kode programnya? Silakan kalian lihat gambar berikut:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main(){

    int x;

    cout << "Masukkan Bilangan : " ;
    cin >> x;

    if (x != 0){
        if (x > 0){
        cout << "Bilangan Positif" << endl;
        }
        else if (x < 0){
        cout << "Bilangan Negatif" << endl;
        }

    }else
    {
        cout << "Bilangan Nol" << endl;
    }

}
```

Jika kode program di atas kalian jalankan, maka tampak output seperti berikut ini:

```
Masukkan Bilangan : 17
Bilangan Positif
Press any key to continue . .
```

Bisakah kalian menjelaskan cara kerja program di atas? Saat kalian memasukkan sebuah bilangan atau angka, maka angka atau bilangan tersebut akan tersimpan dalam variabel X. Angka yang kalian masukkan tadi akan diproses oleh program saat membaca kondisinya. Misalnya kalian masukkan angka 17, maka program akan mengeluarkan keterangan ”bilangan positif”. Jika angka yang kalian masukkan adalah 0 (nol), maka program akan mengeluarkan keterangan ”bilangan nol”.

6. Percabangan Bersarang (*Nested* IF)

Bentuk percabangan ini termasuk struktur percabangan yang cukup kompleks. Bentuk percabangan ini mengizinkan adanya struktur IF di dalam struktur IF lainnya. Perhatikan bentuk percabangannya berikut ini:

```
if(kondisi_1){
   if(kondisi_1a){
      Statemen_jika_kondisi_1_dan_1a_terpenuhi;
   }
   else if(kondisi_1b){
      Statemen_jika_kondisi_1_dan_1b_terpenuhi;
   }
   .....
   else{
      Statemen_jika_hanya_kondisi_1_yang_terpenuhi;
   }
}
else if(kondisi_2){
   if(kondisi_2a){
      Statemen_jika_kondisi_2_dan_2a_terpenuhi;
   }
   else if(kondisi_2b){
      Statemen_jika_kondisi_2_dan_2b_terpenuhi;
   }
   .....
   else{
      Statemen_jika_hanya_kondisi_2_yang_terpenuhi;
   }
}
else if(kondisi_3){
   Statemen_jika_kondisi_3_terpenuhi;
}
.....
else{
   Statemen_jika_semua_kondisi_tidak_terpenuhi;
}
```

Untuk lebih memahami tentang percabangan ini, coba kalian perhatikan kasus berikut:

"Sebuah perusahaan akan memberikan komisi pada para salesmannya. Nah, ada beberapa ketentuan yang harus dipenuhi, seperti berikut ini:

- *Angka yang dimasukkan tidak boleh negatif.*
- *Jika salesmannya berhasil menjual barangnya lebih dari Rp500.000 maka mereka akan mendapatkan uang jasa Rp30.000 dan tambahan komisi sebesar 20% dari hasil penjualannya hari itu.*
- *Jika salesmannya berhasil menjual barangnya lebih dari Rp200.000 sampai Rp500.000 maka mereka akan mendapatkan uang jasa Rp20.000 dan tambahan komisi sebesar 15% dari hasil penjualannya hari itu.*
- *Dan jika salesmannya berhasil menjual barangnya sampai Rp200.000 maka mereka akan mendapatkan uang jasa Rp10.000 dan tambahan komisi sebesar 10% dari hasil penjualannya hari itu."*

Jika kalian membuat kode program untuk kasus di atas maka akan tampak seperti berikut ini:

```
#include<iostream>
using namespace std;
int main()
{
    float pendptan, jasa=0, komisi=0, total=0;
    cout<<" Pendapatan Hari ini Rp. "; cin>>pendptan;
    if (pendptan >= 0 ){
        if (pendptan >= 500000 )
        {jasa=30000;
        komisi=0.2*pendptan;}

        else if(pendptan > 200000 && pendptan < 500000)
        {jasa=20000;
        komisi=0.15*pendptan;}

        else
        {jasa=10000;
        komisi=0.1*pendptan;}
        /* menghitung total */
        total = komisi+jasa;
        cout<<" Uang Jasa Rp. "<<jasa<<endl;
        cout<<" Uang Komisi Rp. "<<komisi<<endl;
        cout<<" ============================"<<endl;
        cout<<" Hasil Total Rp. "<<total<<endl;
    }
    else{cout << " Data yang di inputkan tidak boleh negatif" << endl;}
    return 0;
}
```

Jika kode program di atas dijalankan dengan pendapatan lebih dari Rp500.000 maka output-nya akan tampak seperti berikut:

```
Pendapatan Hari ini Rp. 600000
Uang Jasa Rp. 30000
Uang Komisi Rp. 120000
============================
Hasil Total Rp. 150000
```

Jika kode program di atas dijalankan dengan pendapatan antara Rp200.000 sampai Rp500.000 maka output-nya akan tampak seperti berikut:

```
Pendapatan Hari ini Rp. 250000
Uang Jasa Rp. 20000
Uang Komisi Rp. 37500
============================
Hasil Total Rp. 57500
```

Jika kode program di atas dijalankan dengan pendapatan kurang dari Rp200.000 maka output-nya akan tampak seperti berikut:

```
Pendapatan Hari ini Rp. 150000
Uang Jasa Rp. 10000
Uang Komisi Rp. 15000
============================
Hasil Total Rp. 25000
```

Jika kode program di atas dijalankan dengan angka negatif, maka output-nya akan tampak seperti berikut:

```
Pendapatan Hari ini Rp. -50000
Data yang di inputkan tidak boleh negatif
```

7. Perintah *Switch*

Setelah kalian mengetahui percabangan dengan IF...ELSE, sekarang kita akan bahas percabangan dengan *Switch*. Megnapa kita membahas *Switch*? Apakah tidak cukup program percabangan dibuat menggunakan struktur IF...ELSE saja?

Nah, ada kalanya kalian akan menemukan kasus pemrograman yang dihadapkan pada banyak pilihan serupa, sehingga kalian akan merasakan kesulitan dan kerepotan jika menggunakan struktur IF... ELSE saja. Pada kondisi inilah kalian lebih baik menggunakan *Switch*. Berikut adalah bentuk penulisan struktur *Switch*:

```
switch(ekspresi){
     case kondisi_1:perintah_1; break;
     case kondisi_2:perintah_2; break;
     default:
          perintah_3;
}
```

Apa yang harus kalian perhatikan saat menggunakan Switch? **Pertama**, kondisi harus berisi data yang bertipe *Integer*, *Char*, dan *Boolean*. **Kedua**, perintah **break** ini berfungsi untuk menjalankan perintah jika ada satu kondisi terpenuhi. Jika semua kondisi tidak terpenuhi, maka akan dikerjakan perintah yang ada pada bagian **default.**

Lalu seperti apa contoh penerapan Switch ini dalam kasus pemrograman sehari-hari? Perhatikan kode program berikut ini:

```
#include <iostream>
#include <conio.h>
using namespace std;

void main(){
    int bil;
    cout << "Masukkan bilangan : "; cin >> bil;

    switch(bil){
    case 1 : cout << "Anda memasukkan bil. satu \n" ;
    break;
    case 2 : cout << "Anda memasukkan bil. dua \n" ;
    break;
    case 3 : cout << "Anda memasukkan bil. tiga \n" ;
    break;
    default : cout << "Anda memasukkan bil selain 1, 2, dan 3";
    break;
    }
}
```

Berikut adalah output dari kode program di atas:

```
Masukkan bilangan : 3
Anda memasukkan bil. tiga
Press any key to continue . . .
```

Pada program di atas, jika kalian memasukkan angka **3** maka akan menghasilkan teks **Anda memasukkan bil.tiga**. dan jika kalian memasukkan angka **selain 1, 2, dan 3**, maka akan keluar teks pada bagian default "**Anda memasukkan bil selain 1, 2, dan 3**".

Selain penulisan struktur percabangan atau pembuatan programnya, kalian juga harus mengetahui apa saja yang harus diperhatikan saat kalian membuat program percabangan, di antaranya:

- Ketika menggunakan operator AND dalam percabangan, semua kondisi harus terpenuhi untuk memperoleh kondisi yang

bernilai benar atau TRUE, sedangkan menggunakan operator OR, hanya salah satu kondisi saja yang bernilai benar? TRUE maka kondisi akan bernilai benar/TRUE.

- Kalian harus memperhatikan penggunan tanda sama dengan. Jika maknanya sama dengan makan kalian gunakan tanda ini "==". Karena jika kalian menggunakan tanda sama dengannya hanya satu seperti "=", maka maknanya adalah sebagai operator penugasan (*assignment*).

D. Struktur Kontrol Perulangan

Aktivitas Belajar 5-4

Pada bagian ini kalian akan diminta menganalisis kode program berikut:

```
#include<iostream>
#include <conio.h>
using namespace std;
int main()
{
    int T=0,N=..... , X=.....;
    while(T<=100)
    {
            T=T+N;
            N=N+X;
            X=X+5;
    }
    cout<<"T="<<T<<endl;
    getch();
}
```

Untuk bisa menampilkan nilai T = 120, maka berapa nilai yang tepat untuk titik-titik di atas?

1. Struktur Kontrol Perulangan

Dalam materi sebelumnya kalian sudah mempelajari dan mengetahui tentang struktur sekuensial dan struktur kontrol percabangan. Selain kedua struktur tersebut, kalian juga harus mengetahui struktur kontrol perulangan. Apa itu struktur kontrol perulangan? Mengapa kalian harus mengetahui dan mempelajari struktur kontrol perulangan?

Pada saat kalian membuat program, kalian akan dihadapkan pada kasus pemrograman perulangan. Seperti membuat deret matematika, membuat data berulang, atau membuat program pemilihan menu yang bisa diulang-ulang. Dengan menggunakan struktur perulangan, kalian tidak perlu lagi melakukan penulisan kode program beberapa kali sebanyak data yang diulang.

2. Karakteristik dan Cara Kerja Struktur Kontrol Perulangan

Dalam perulangan ini, kalian harus mengetahui bagian-bagiannya:

a. Inisialisasi

Pada bagian ini, akan dilakukan pemberian nilai atau kondisi awal sebelum proses perulangan dilakukan. Pemberian nilai awal ini biasanya pada variabel.

b. Proses

Bagian ini berisi semua proses yang akan dikerjakan secara berulang- ulang.

c. Iterasi

Iterasi berfungsi untuk melakukan penambahan atau pengurangan pada kondisi perulangan.

d. Terminasi/kondisi perulangan

Pada bagian ini, menunjukkan terminasi atau berhentinya suatu proses yang dikerjakan berulang-ulang. Menentukan kondisi berhenti ini sangat penting kalian perhatikan agar perulangan bisa berhenti dan tidak melakukan perulangan terus-menerus.

3. Bentuk Perulangan

Dalam pemrograman ini, perulangan yang akan dibahas ada 3 bentuk, yaitu FOR, WHILE dan DO WHILE.

a. **Perintah FOR**

Saat kalian membuat program perulangan dengan FOR ini, maka kalian harus sudah tahu berapa kali perulangan akan dilakukan. Kalian harus tahu perulangan ini dimulai angka berapa dan di akhiiri angka berapa. Kalian akan membutuhkan sebuah variabel sebagai indeksnya.

Berikut bentuk struktur FOR:

1) **FOR menaik**

For (variabel=nilai_awal; kondisi perulangan; variabel++)
{
    Pernyataan;
}

2) **FOR menurun**

For (variabel=nilai_akhir; kondisi perulangan; variabel--)
{
    Pernyataan;
}

Untuk memahami bentuk FOR di atas, coba kalian perhatikan kode program berikut:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      for (n = 1; n <= 10; n++){
7          cout << n;
8      }
9      return 0;
10 }
```

Kode program di atas adalah program untuk menampilkan deret bilangan 1 sampai 10 dengan penambahan 1. Variabel yang digunakan dalam program di atas adalah n. Nilai awal yang diberikan pada n adalah 1. Batas perulangan yang akan

dilakukan adalah **11** (n<11). Program akan berhenti jika melebihi **10**. Kemudian penambahan n adalah 1 sampai n bernilai 10 (n++). Jika program di atas dijalankan, maka akan tampak output seperti berikut ini:

```
12345678910Press any key to continue . . .
```

b. **WHILE**

Berbeda dengan struktur FOR, dalam struktur ini pemeriksaan kondisi perulangan dilakukan di awal blok perulangan. Semua pernyataan yang ada dalam badan perulangan akan diproses apabila kondisi perulangan bernilai TRUE.

Bentuk WHILE tampak seperti berikut:

```
While (kondisi perulangan)
{
   Pernyataan;
}
```

Untuk lebih jelasnya tentang perulangan dengan bentuk WHILE ini, coba kalian perhatikan kode program berikut:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int i = 10;
7      while(i > 0){
8          cout << i << endl;
9          i--;
10     }
11     return 0;
12 }
```

Pada program di atas, kalian bisa mengetahui variabel yang digunakan adalah i. Nilai awal yang diberikan pada i adalah 10, artinya nilai awal yang akan ditampilkan adalah 10. Dalam program di atas terjadiproses pengurangan variabel i—artinya terjadi pengurangan 1. Pada badan perulangan akan dilakukan proses perulangan menampilkan nilai variabel i, sampai i bernilai 0 dan perulangan akan berhenti.

Hasil output dari kode program di atas adalah berikut:

c. **DO ... WHILE**

Berbeda dengan struktur WHILE, dalam struktur ini pemeriksaan kondisi perulangan akan dilakukan di akhir blok perulangan. Bentuk DO WHILE adalah seperti berikut:

```
do
{
   Pernyataan;
} While (kondisi perulangan);
```

Untuk lebih jelasnya, coba kalian perhatikan kode berikut ini:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int i = 0;
6      do {
7          cout << "C++\n";
8          i++;
9      }
10     while(i<15);
11
12     return 0;
13 }
```

Kode program di atas menunjukkan adanya pengulangan kata ”C++ ” sebanyak 15 kali. Variabel yang terdapat pada program di atas adalah i, dengan nilai awalnya adalah 0. Pengulangan akan dilakukan dengan penambahan variabelnya 1, dan akan berhenti saat i bernilai 15. Output dari program di atas adalah seperti berikut ini:

Penerapan struktur DO WHILE ini, bisa juga kalian lihat pada program struktur menu, serta pada program *Login* yang meng-input-kan nama *user* dan *password*.

a. Perulangan Bersarang

Seperti halnya percabangan, dalam perulangan pun kalian bisa menerapkan perulangan bersarang atau *nested looping*. Yang artinya ada blok perulangan di dalam blok perulangan. Perulangan bersarang ini bisa kalian praktikkan dengan struktur FOR (*Nested* FOR), struktur WHILE (*Nested While*), dan struktur *Do* WHILE (*Nested Do While*). Penjelasan terkait perulangan bersarang ini akan lebih cepat kalian pahami dengan memperhatikan, dan mempraktikkan serta menganalisis kode-kode program berikut:

1) **Nested FOR**

Bentuk umumnya:

```
for (inisialisasi1;kondisi1;pencacah1++){
    for (inisialisasi2;kondisi2;pencacah2++){
        for (inisialisasi3;kondisi3;pencacah3++){
            statemen_yang_akan_diulang;
        }
    }
}
```

Contoh programnya:

```
#include <iostream>
using namespace std;

int main(){
    int x, y;
    for(x = 1; x <= 3; x++){

        for (y = 1; y <= 4; y++){
            cout << y ;
        }
        cout << endl;

    }
}
```

Output-nya adalah:

```
1234
1234
1234
Press any key to continue . . .
```

2) **Nested WHILE**

Bentuk umum perulangan *while* bersarang:

```
while(kondisi perulangan){
        while(kondisi perulangan){
                pernyataan;
        }
        pernyataan; (bisa ditambah pernyataan)
}
```

Contoh program *nested while*:

```
#include <iostream>
using namespace std;

int main(){
    int x, y;
    x = 1;
    while(x <= 3){
        y = 1;
        while(y <= 4){
            cout << y ;
            y++;
        }
        cout << endl;
        x++;
    }
    return 0;
}
```

Output dari program ini adalah:

```
1234
1234
1234
Press any key to continue . . .
```

3) ***Nested* DO WHILE**

Bentuk umum perulangan *do*-while bersarang:

```
do{
        pernyataan;(bisa ditambah pernyataan)
        do{
                pernyataan;
        }
        while(kondisi perulangan);
        pernyataan;
}
while(kondisi perulangan);
```

Contoh program *nested do while*:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int x, y;
6      x = 1;
7
8      do {
9          y = 1;
10         do {
11             cout << y;
12             y++;
13
14         }while(y <= 4);
15         cout << endl;
16         x++;
17     }while(x <= 3);
18 }
```

Output dari program di atas adalah:

```
1234
1234
1234
Press any key to continue . . .
```

Aktivitas Belajar 5-5

Nah, sekarang coba kalian buat sebuah program perulangan untuk membuat segitiga dengan # seperti tampak berikut ini:

```
#
##
###
####
#####
```

E. Struktur Data

Aktivitas Belajar 5-6

Seperti pada bab-bab sebelumnya, di awal pembelajaran, kalian diminta mengisi tabel KWL (***Know***, ***Want to Know***, ***Learn***).

Berikut adalah panduan untuk mengisi tabel pada bab ini.

1. Apa yang kalian ketahui tentang struktur data dan struktur kontrol pemrograman? Apa yang kalian pahami tentang struktur data dan struktur kontrol permrograman serta bagaimana penerapannya dalam pemrograman?
2. Tuliskan apa yang ingin kalian ketahui tentang struktur data dan penerapannya dalam pemrograman?

K (Apa yang ingin diketahui) *Diisi di awal pembelajaran*	W (Apa yang ingin dipelajari) *Diisi di awal pembelajaran*	L (Apa yang sudah dipelajari) *Diisi di awal pembelajaran*

Setelah kalian mengisi tabel KWL di atas, silakan diskusikan jawaban kalian bersama teman-teman lain, untuk saling berbagi informasi.

Setelah kalian mempelajari materi mulai dari tipe data, struktur sekuensial, struktur percabangan, dan struktur perulangan, sekarang kalian akan mempelajari tentang struktur data. Ada 2 jenis struktur data yang akan dibahas dalam bab ini, yaitu struktur data statis dan struktur data dinamis.

Struktur data statis yang akan kalian pelajari adalah tentang array. Sedangkan struktur data dinamis yang akan kalian pelajari adalah list dan stack.

1. *Array*

Kalian pernah mendengar istilah array dalam pemrograman? Pastinya iya, karena array ini sangat banyak digunakan dalam pemrograman. *Array* ini merupakan sekumpulan dari beberapa nilai dengan tipe yang sama dalam sebuah urutan tertentu dan menggunakan nama yang sama. Letak atau posisi elemen array ini dikenal dengan nama indeks. Dalam pemrograman C++ *indeks* ini dimulai dari 0 (nol). Sedangkan cara penulisan array-nya menggunakan tanda kurung [ ] atau bracket.

Cara penulisan secara lengkap untuk array:

*Tipe_data nama_*array *[jumlah_elemen]*

Contoh pendeklarasiannya: **int X[9]** yang berarti mendefinisikan sebuah array dengan nama X sebanyak 9 elemen dari index 0 sampai 8, dengan cara akses:

X[0], X[1],.. dst

Sebagai gambaran kalian bagaimana array ini bekerja, kalian diminta menampilkan data yang bertipe sama sebanyak 5 angka yang sudah kalian inputkan sebelumnya dengan keyboard, maka kode program yang kalian buat akan seperti ini:

```
int x1, x2,x3,x4,x5;
    cin>>x1;
    cin>>x2;
    cin>>x3;
    cin>>x5;
    cin>>x5;
    cout<<x1;
    cout<<x2;
    cout<<x3;
    cout<<x4;
    cout<<x5;
```

Jika banyaknya data masih bisa dikerjakan dengan cara di atas, maka apakah kalian masih mengerjakan dengan cara yang sama jika data yang ditampung sebanyak 2.000 data? Berarti akan membutuhkan 2.000 variabel juga. Tentu saja tidak mungkin. Maka di sinilah kalian akan menggunakan array untuk menyelesaikannya.

Ada dua cara untuk mengisi dan menampilkan array, yaitu mengisi dan menampilkan array secara langsung, yang kedua mengisi dan menampilkan array dengan perulangan. Cara kedua ini bisa kalian gunakan dan lebih efektif jika data arraynya berjumlah banyak. Jika menggunakan cara yang pertama, akan sangat merepotkan kalian

nantinya, karena akan lebih banyak baris perintah yang dibutuhkan. Agar kalian bisa lebih paham penggunaan array, perhatikan kode program berikut ini:

```
#include <iostream>
using namespace std;

int main(){
    int a[12] = {1,3,5,4,7,2,99,16,45,67,89,45};
    int indeks, total = 0;
    for (indeks = 0; indeks <= (12-1); indeks++)
    {
        total += a[indeks];
    }
    cout << " Total setiap elemen array  adalah " << total << endl;
    return 0;
}
```

Program di atas adalah program yang digunakan untuk menghitung jumlah setiap elemen dalam array. Jika program di atas dijalankan, maka akan tampak output seperti berikut:

```
Total setiap elemen array  adalah 383
```

Jenis-jenis array yang harus kalian ketahui adalah array berdimensi satu, array berdimensi dua, dan array berdimensi tiga.

- *Array* berdimensi satu ini adalah array yang paling sering digunakan. Seperti namanya, dia hanya memiliki satu subscript. Bentuk penulisannya adalah *tipe_data nama*_array *[jumlah_elemen]*. Contohnya: int *bilangan[10],* char *nama[35], float nilai[100]*. Bentuk kode program:

```
#include <iostream>
using namespace std;

int main(){
    int square[100];
    int i, k;

    for(i = 0; i< 10; i++){
    k = i + 1;
    square[i] = k * k;
    cout << k << " pangkat 2 adalah  " << square[i] << endl;

    }
    return 0;
}

```

Bila program di atas dijalankan, akan tampak hasil seperti berikut:

```
1 pangkat 2 adalah  1
2 pangkat 2 adalah  4
3 pangkat 2 adalah  9
4 pangkat 2 adalah  16
5 pangkat 2 adalah  25
6 pangkat 2 adalah  36
7 pangkat 2 adalah  49
8 pangkat 2 adalah  64
9 pangkat 2 adalah  81
10 pangkat 2 adalah  100
```

- *Array* dua dimensi memiliki dua buah *subscript*. Kalian pasti pernah mendengar istilah matriks. Nah, array dua dimensi inilah yang digunakan untuk membentuk matriks. Dimensi pertama pada array ini sebagai dimensi baris dan dimensi kedua sebagai dimensi kolom. Bentuk penulisannya adalah *tipe_data nama_*array *[elemen_baris][elemen_kolom]*. Contohnya: int *matriks[3][4],* char *nama[4][4], float nilai[10][10]*.
- Array berdimensi tiga ini sangat jarang digunakan. Array ini terdiri dari tiga subscript. Bisa digunakan untuk membuat array dua dimensi yang tiap-tiap dimensinya menyimpan data string. Bentuk penulisannya adalah *tipe_data nama_*array *[elemen_baris] [elemen_kolom][jumlah_char]*. Contohnya: char *nama[2][2][10]*.

Untuk lebih memahami lagi tentang array, coba kalian buatkan kode program untuk array berdimensi dua dan tiga.

2. *List*

Sekarang kalian akan mempelajari tentang list. Apakah kalian tahu apa itu *List*? *List* ini biasanya digunakan untuk menyimpan data yang dilakukan dengan struktur, agar secara otomatis kalian bisa menciptakan suatu tempat penyimpanan yang baru kapan saja data itu diperlukan.

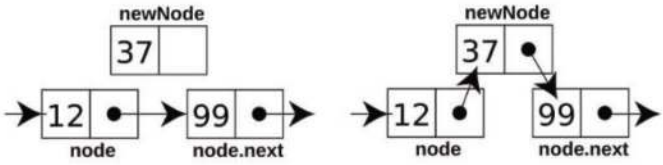
Ada beberapa teknik yang bisa kalian gunakan, di antaranya:

a. Pengulangan *linked list*.
b. Mengubah sebuah *pointer* dengan referensi *pointer*.
c. Membuat kepala senarai dengan perintah *push*().
d. Menambah ekor pada pada akhir senarai.
e. Membuat referensi lokal.

Struktur data merupakan suatu cara yang digunakan untuk menyimpan elemen data dalam memori komputer. Struktur data berguna untuk mengakses data secara efisien. Salah satu jenis struktur data adalah

struktur data linier yang menyimpan data secara berurutan atau satu per satu. *Stack* dan *Linked List* adalah dua struktur data linier tersebut.

Linked List termasuk struktur data yang terdiri dari sekumpulan *node* yang tersusun secara berurutan.



Gambar 5.3 *List*
Sumber: Marwondo & Rini Melati/2022

Berikut ini adalah beberapa jenis *linked list*, di antaranya:

a. *Single linked list*

   Node dalam list ini menyimpan data dan alamat *node* berikutnya. Bentuk strukturnya mirip seperti rantai. Memasukkan, menghapus, dan melintasi elemen merupakan beberapa operasi yang dapat dilakukan dalam *single linked list*.

b. *Double linked list*

   Node dalam list ini menyimpan data dengan dua alamat yaitu alamat *node* berikutnya dan alamat *node* sebelumnya. Mirip dengan daftar bertaut tunggal. Beberapa operasi yang bisa dilakukan dalam list ini di antaranya penyisipan, penghapusan, dan melintasi elemen.

Nah, dalam buku ini akan dibahas tentang *Single Linked List*, ya, karena ini termasuk list yang sederhana yang bisa kalian pahami, yang hanya memiliki satu pointer dengan satu arah data saja. Kalian harus tahu dulu bagaimana langkah membuat sebuah *linked list*:

a. Yang pertama yang harus kalian lakukan adalah mendeklarasikan dulu *struct node*-nya.

```
8  struct TNode
9  {
10     int nilai;
11     TNode *next;
12 };
```

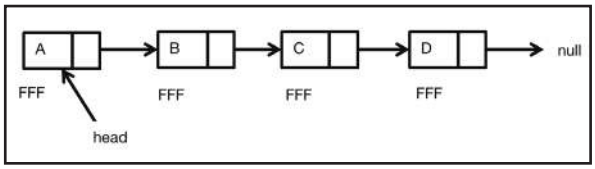
Keterangan:

- *Struct* yang dibuat diberi nama *TNode* yang memiliki dua *field* yaitu data dengan tipe data integer dan *next* bertipe pointer dari *TNode*.
- Setelah *struct* dibuat, kemudian membuat variabel *head* yang bertipe *pointer* dari *TNode* sebagai kepala dari *linked list*.

b. Lalu membuat *node head*-nya. Gunakan *keyword new* untuk membuat sebuah *node* baru yang akan diisi data lalu *pointer next*-nya adalah NULL

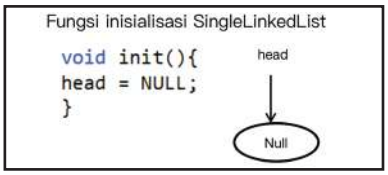
```
31      TNode *baru;
32      baru = new TNode;
33      baru -> nilai = TNodeBaru;
34      baru -> next = NULL;
```

c. Inisialisasikan *node head*-nya. Gunakan variabel *pointer head* yang akan selalu menunjuk *node* pertama.



Gambar 5.4 Inisialisasi *node head*
Sumber: Marwondo & Rini Melati/2022

Lakukan deklarasi *pointer* berikut:



Membuat fungsi yang akan digunakan untuk mengetahui apakah kosong atau tidaknya *single linked list*.

```
22  bool isEmpty()
23  {
24      if (head==NULL)
25          return true;
26      return false;
27  }
```

d. Tambahkan *node* baru di depan dan di belakang. Saat menambahkan *node* baru ini pasti akan dikaitan dengan *node* yang paling depan. Tetapi jika baru pertama kali dan data

masih kosong, maka menambahkan data ini dilakukan dengan menunjukkan *head* pada *node* baru tersebut. *Head* akan selalu jadi data data terdepan.

```
void tambahTNodeDepan (int TNodeBaru)
{
    TNode *baru;
    baru = new TNode;
    baru -> nilai = TNodeBaru;
    baru -> next = NULL;
    if (isEmpty())
    {
        head = baru;
        head->next=NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
    cout << "TNode Depan" << TNodeBaru << " Masuk" << endl;
}
```

e. Kemudian menghapus *node* tersebut. Fungsi berikut ini akan menghapus data paling atas dan yang terdepan yang ditunjuk oleh *head* dalam *linked list*.

```
void hapusDepan ()
{
    if (!isEmpty())
    {
        int tmp;
        if (head->next!=head)
        {
            Data *hapus;
            hapus=head;
            tmp=head->nilai;
            Data *bantu;
            bantu=head;
            while (bantu->next!=head)
            {
                bantu=bantu->next;
            }
            bantu->next=head->next;
            head=head->next;
            delete hapus;
        }
        else
        {
            tmp=head->nilai;
            awal();
        }
        cout << tmp << "Dihapus" << endl;
    }
    else
    {
        cout << "Data kosong";
    }
}
```

Berikut ini adalah hal-hal yang harus kalian perhatikan saat melakukan penghapusan *node*.

- Tidak boleh melakukan menghapus *node* saat sedang ditunjuk oleh pointer, kalian harus menggunakan pointer lain untuk menunjuk *node* yang akan dihapus.
- Sebelum kalian menghapus data terdepan, *head* harus dalam posisi menunjuk *node* sesudahnya agar list-nya tidak terputus.
- Jika *data head* yang masih *null*, itu artinya data masih kosong.

Single linked list ini ada dua jenis ya, yang pertama adalah single *linked list* non circular dan yang kedua adalah single *linked list* circular. Perbedaan keduanya ada pada next *node* terakhir yang ada pada circular *linked list* selalu akan merujuk pada *node* pertama.

Setelah kalian mengetahui tentang single *linked list*, kalian lakukan eksplorasi lebih jauh lagi tentang double *linked list*-nya ya.

Aktivitas Belajar 5-7

Berikut ini kalian akan diberi kode program untuk kedua jenis *single linked list* tersebut, hanya kode program utamanya saja, ya. Nah, tugas kalian adalah membuat program fungsinya, menjalankan kode program tersebut, kemudian analisis dan jelaskan bagaimana cara kerja kedua program tersebut. Tuliskan hasil analisis kalian pada tabel berikut ini:

Kode Program Utama	Program Fungsi	Jenis Linked List	Output Program	Cara kerja program

`int main()`
`{`
`awal();`
`tambahDataBelakang(3);`
`tambahDataDepan(5);`
`tambahDataBelakang(15);`
`tambahDataBelakang(1);`
`tambahDataBelakang(25);`
`tambahDataBelakang(8);`
`cout << "Data pada single linked list circular:" << endl;`
`Cetak();`
`cout << "Data paling depan dihapus:" << endl;`
`hapusDepan();`
`cout << "Data paling belakang`				

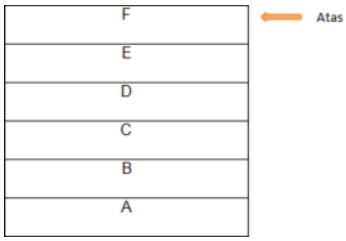
```
dihapus:" << endl;
hapusBelakang();
   cout << "Data
pada single linked
list circlar:" <<
endl;
   Cetak();
   cout <<
"Panjang linked
list:" << endl;
   cout <<
panjang();
   getch();
   return 0;
}
int main()
{
   awal();
tambahDataBelakang(3);
tambahDataDepan(5);
tambahDataBelakang(15);
tambahDataBelakang(1);
tambahDataBelakang(25);
tambahDataBelakang(8);
   cout << "Data
pada single linked
list non circular:"
<< endl;
   Cetak();
   cout << "Data
paling depan
dihapus:" << endl;
hapusDepan();
   cout << "Data
pada single linked list
non circular:"
<< endl;
   Cetak();
   cout << "Data
paling belakang
dihapus:" << endl;

hapusBelakang();
   cout << "Data
pada single linked list
non circlar:"
<< endl;
```

```
   Cetak();
   cout <<
"Panjang linked
list:" << endl;
   cout <<
panjang();
getch();
return 0;
}
```

3. *Stack*

Berikut ini kita akan bahas tentang stack. Sudah tahu tentang stack? Nah, secara sederhana stack atau tumpukan bisa diartikan sebagai kumpulan data yang seolah-olah diletakkan di atas data yang lain. Satu hal yang perlu kalian ingat bahwa kalian bisa menambah data atau menyisipkannya dan kalian juga bisa mengambil data atau menghapusnya melalui ujung yang sama yang disebut sebagai ujung atas stack.

Perhatikan gambar berikut:



Gambar 5.5 *Stack*
Sumber: Marwondi & Rini Melati/2022

Dari gambar di atas, bisa kalian lihat ya kotak B itu ada di atas kotak A, dan di bawah kotak C. sedangkan kotak D itu ada di atas kotak C dan di bawah kotak E. Posisi ujung stack itu adalah kotak F. Jika kalian akan menyisipkan kotak lain, maka posisi kotak tersebut ada di atas kotak F. Namun jika kalian ingin mengambil kotak lain, maka kalian harus mengambil kotak F terlebih dahulu.

Jadi ada dua operasi yang bisa dilakukan pada stack, yaitu menyisipkan data (*push*) dan menghapus data (*pop*).

- *Push*

  Ini artinya memasukkan data ke dalam stack. Berikut ini adalah contoh kode program untuk push:

```
void Push(NOD **T, char item)
{
    NOD *n;
    n = NodBaru(item);
    n -> next = *T;
    *T = n
}
```

- *Pop*

  Ini artinya kita akan menghapus data elemen yang ada di posisi paling atas sebuah stack. Berikut ini adalah contoh kode program untuk pop:

```
Char Pop (NOD **T)
{
        NOD *P; char item;

        if ( ! StackKosong (*T) ) {
            P = *T;
            *T = (*T) ->next;
            Item = P->data;
            Free (P);
        }
        Return item;
}
```

Uji Kompetensi

Jawablah pertanyaan di bawah ini untuk mengetahui apakah anda sudah menguasai teori tentang Bahasa Pemrograman. Anda bisa memilih* option *yang menurut anda benar!

1. Ada kumpulan data umur siswa yang menampung 5 buah data. Penulisan sintaks Array yang tepat untuk data tersebut adalah ....
   a. int umursiswa = {18,19,20,21,22};
   b. int umursiswa[] = {18,19,20,21,22};
   c. int umursiswa{} = {18,19,20,21,22};
   d. int umursiswa[] = [18,19,20,21,22];
   e. int umursiswa{} = [18,19,20,21,22];
2. Perhatikan pernyataan berikut ini:
   1. int daftarNilai [] = {10,9,8,10,9};
   2. int matriks [2][4] = {{1,2,3,4},{5,6,7,8}};
   3. int umur[] = {18,19,20,21,22};

4. int tabel[3][5] = {
   {1, 2, 3, 4, 5},
   {2, 4, 6, 8, 10},
   {3, 6, 9, 12, 15}
   };
5. char nama[100][100];

Dari pernyataan di atas, yang merupakan array 1 dimensi adalah ....

a. 1 dan 2
b. 1 dan 3
c. 2, 3, dan 4
d. 2, 3, dan 5
e. 4 dan 5

3. Di bawah ini yang merupakan tipe data dalam C++ ....
   a. *int, float, char, double,* const
   b. *float, int, double,* const
   c. *int, float, char, double*
   d. const, *main, void, double*
   e. const, *main, void, define*
4. Suatu tempat untuk menampung nilai atau data yang dapat berubah-ubah disebut?
   a. Operator
   b. Tipe data
   c. *Main*
   d. Variabel
   e. Konstanta
5. Perhatikan potongan program berikut:

```
cout<<"suhu=";cin>>suhu;
if (suhu>100)
       cout<<"gas";
else if(suhu>0)
              cout<<"cair";
       else
              cout<<"padat";
getch();
}
```

Jenis struktur percabangan yang digunakan pada potongan program di atas adalah ....

a. Menggunakan IF 1 kondisi
b. Menggunakan IF 2 kondisi
c. Menggunakan IF bersarang
d. Menggunakan IF banyak kondisi
e. Menggunakan IF ELSE

6. Perhatikan kode program berikut:

```
#include <iostream>
#include <conio.h>

using namespace std;

void main(){
    int x, y, z;
    cout << "x = "; cin >> x;
    cout << "y = "; cin >> y;
    cout << "z = "; cin >> z;

    if(x > y)
    {
        if (x > z)
        {
            cout << "yang paling besar = " << x;
        }
        else
        {
            cout << "yang paling besar = " << z;
        }
    }
    else if(y > z)
    {
        cout << "yang paling besar = " << y;
    }
    else
    {
        cout << "yang paling besar = " << z;
    }
}
```

Jika x, y, dan z masing-masing bernilai 7, 3, dan 5, maka program di atas akan menghasilkan ....

a. Yang paling besar 7
b. Yang paling besar 5
c. Yang paling besar 3
d. Yang paling besar x
e. Yang paling besar z

7. Diketahui data deret berikut:

   1, 2, 4, 8, 16, 32, 64, 128, 256, 512

   Jika menggunakan *Do While*, maka kondisi yang tepat untuk perulangan di atas adalah ....

   a. while(a<512);
   b. while(a<=512);
   c. while(a>512);
   d. while(a>=512);
   e. while(a=512);

8. Perhatikan program berikut ini:

```
#include<iostream>
#include <conio.h>
using namespace std;
int main()
{
    int T=0,N=..... , X=.....;
    while(T<=100)
    {
            T=T+N;
            N=N+X;
            X=X+5;
    }
    cout<<"T="<<T<<endl;
    getch();
}
```

   Untuk bisa menampilkan nilai T = 120, maka nilai yang tepat untuk titik-titik di atas adalah ....

   a. 10 dan 10
   b. 20 dan 20
   c. 30 dan 30
   d. 40 dan 40
   e. 50 dan 50

9. Perhatikan program berikut:

```
#include<iostream>
#include<conio.h>
using namespace std;
void main()
{
float N;
cout<<"Masukan NIlai N=";cin>>N;
if(N>50)
       {
          .....;
       }
else
      {
         N=N;
      }
.....;
cout<<"Tampilkan N="<<N;
getch();
}
```

Jika nilai N yang di-input-kan adalah 75 maka nilai N yang akan ditampilkan oleh program di atas adalah 60. Sintak yang tepat untuk titik-titik di atas adalah ....

a. N=N-25 dan N=N+10
b. N=N-30 dan N=N+20
c. N=N-45 dan N=N+35
d. N=N-50 dan N=N+40
e. N=N-55 dan N=N+50

10. Proses *looping* yang menggunakan penghitung (*counter*) dapat dibuat menggunakan pernyataan ini. Pernyataan ini digunakan bila anda sudah tahu berapa kali anda akan mengulang satu atau beberapa pernyataan. Yang dimaksud adalah *looping* ....

a. FOR
b. WHILE
c. DO WHILE
d. IF THEN
e. REPEAT

Pengayaan

Jika kalian tertarik dengan materi ini dan ingin mendalaminya lebih jauh, berikut buku yang bisa kalian jadikan referensi:

1. BUKU PEMROGRAMAN DASAR untuk siswa SMK/MAK Kelas X Program Keahlian Teknik Komputer dan Informatika, Penulis Rini Melati, S.Kom, Penerbit PT Sarana Pancakarya Nusa, Tahun 2019, Nomor ISBN 978-602-485-172-9.

Selain buku di atas, kalian juga bisa mencoba membuat sebuah projek atau aplikasi utuh yang didalamnya ada struktur percabangan dan perulangan. Salah satu program yang kalian bisa eksplorasi lebih jauh adalah seperti berikut ini:

”Buatlah program untuk menginputkan 100 nilai ujian siswa. Program ini harus bisa menampilkan nilai tertinggi dan terendah ya. selain itu program ini juga harus bisa menampilkan ada berapa siswa yang mendapat nilai terbesar dan terkecil. ” Semangat eksplorasi ya.

Refleksi

Kalian bisa memberikan tanda (√) pada kotak yang kalian anggap sesuai! Setelah kalian mempelajari bab ini, bagaimanakah penguasaan kalian terhadap materi-materi berikut?

No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Penggunaan Tipe Data			
2.	Struktur Kontrol Sekuensial			
3.	Struktur Kontrol Percabangan			
4.	Struktur Kontrol Perulangan			
5.	Penerapan Struktur Data			
6.	Dasar Algoritma Pemrograman			

1. Berdasarkan materi-materi yang sudah kalian pelajari di atas, manakah bagian yang kalian sangat sukai? Jelaskan alasannya!
2. Manfaat apa saja yang kalian peroleh untuk kehidupan sehari-hari, setelah kalian mempelajari semua materi-materi di atas?
3. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2023

Dasar-Dasar Pengembangan Perangkat Lunak dan Gim untuk SMK/MAK Kelas X

Penulis: **Marwondo dan Rini Melati**
ISBN: 978-623-194-476-4 (no.jil.lengkap PDF)
978-623-194-377-1 (jil.1 PDF)

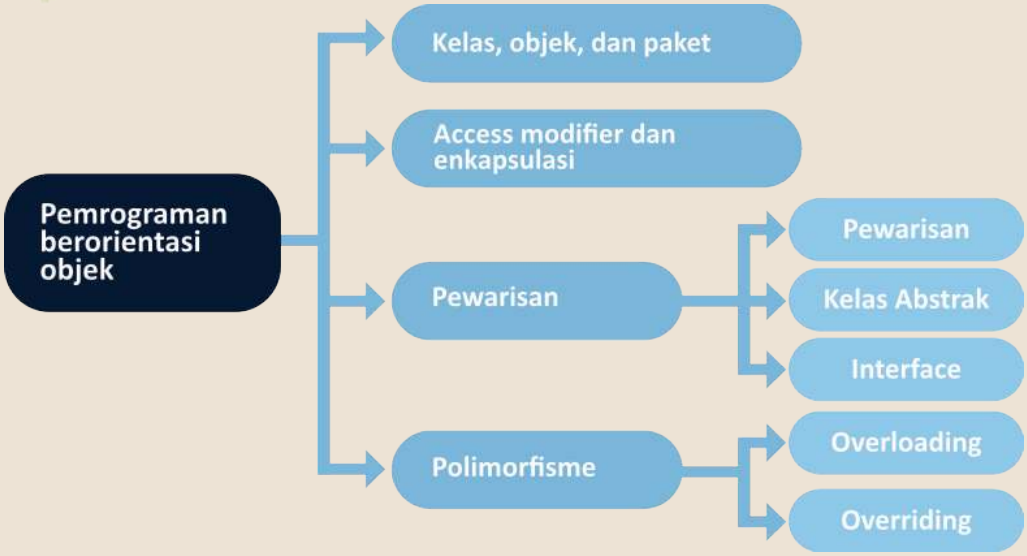
BAB 6

Pemrograman Berorientasi Objek

Tujuan Pembelajaran

Setelah mempelajari ini diharapkan kalian mampu membuat program dengan paradigma berorientasi objek dengan menerapkan berbagai karakteristik yang menyertainya.

Peta Materi



Kata Kunci

♦ **Pemrograman** ♦ **Objek** ♦ **Kelas** ♦ **Paket** ♦ ***Modifier*** ♦ **Enkapsulasi** ♦ **Pewarisan** ♦ **Abstrak** ♦ ***Interface*** ♦ **Polimorfisme** ♦ ***Overload*** ♦ ***Override***.

Gambar 6.1 Candi Borobudur
Sumber: Marwondo, Rini Melati, dan Dana R. N. Adnan/2023

Tahukah kalian, objek apa saja yang menyusun sebuah candi? Bagaimana cara objek-objek tersebut menyusun sebuah candi? Atau, Pernahkah kalian membuat nasi goreng? Apa yang kalian pikirkan ketika akan membuat nasi goreng?

Apersepsi

Ya, tentu dari kalian ada yang berpikir bagaimana urutan kegiatan membuat nasi goreng. Pertama harus bagaimana, setelah itu bagaimana, dan seterusnya. Artinya kalian melihat suatu masalah dari urutan aktivitas bagaimana sebuah masalah tersebut bisa diselesaikan. Tentu ada juga dari kalian yang berpikir bahan dan peralatan apa yang diperlukan. Betul? Ya, itu adalah bagaimana juga kalian melihat masalah dalam perspektif yang berbeda.

Dalam pemrograman juga berlaku hal yang sama. Ada beberapa cara pandang yang bisa digunakan dalam memecahkan suatu masalah. Cara pandang yang berbeda itu yang kita kenal dengan istilah paradigma pemrograman.

Jika sebelumnya kalian sudah memahami membuat program dengan paradigma terstruktur, kali ini kalian akan mempelajari bagaimana melakukan

pemrograman berorientasi objek, sebuah paradigma pemrograman yang tidak hanya melihat pemrograman berdasarkan urutan aktivitasnya tetapi berdasarkan objek-objek yang membangunnya.

A. Kelas, Objek, dan Paket

Pemrograman berorientasi objek merupakan salah satu paradigma dalam pemrograman. Tahukah kalian apa itu paradigma? Paradigma merupakan cara pandang terhadap sesuatu dalam memecahkan masalah. Paradigma mengandung cara, pola, gaya, maupun kerangka berpikir yang berbeda ketika memecahkan suatu masalah.

Bagaimana dengan paradigma pemrograman? Paradigma juga digunakan dalam pemrograman komputer. Paradigma pemrograman adalah kerangka berpikir atau cara pandang pembuatan program (kerangka berpikir atau cara pandang seperti apa yang digunakan pada saat membuat atau menulis program). Paradigma dalam pemrograman dapat digunakan untuk mengklasifikasikan bahasa pemrograman berdasarkan fitur mereka.

Ada banyak paradigma yang berkembang dalam pemrograman. Beberapa paradigma pemrograman yang populer antara lain:

1. Prosedural

   Suatu permasalahan pemrograman dipandang berdasarkan model komputer Von Neumann yang memisahkan antara instruksi (program) dan data yang tersimpan pada memori. Data diakses dan dimanipulasi menggunakan serangkaian perintah yang dijalankan secara runut.

2. Fungsional

   Permasalahan pada pemrograman dilihat atau dipandang dari sisi fungsi-fungsi apa saja yang membangunnya. Fungsi-fungsi didekomposisi sampai pada fungsi yang primitif (tidak dapat diturunkan lagi).

3. Deklaratif, predikatif, atau logis

   Cara memandang paradigma ini didasarkan pada hubungan sebab-akibat antar individu. Program ditulis dalam bentuk basis pengetahuan yang berisi fakta-fakta dan aturan yang digunakan untuk menjawab permasalahan. Hasilnya berupa sebuah kesimpulan berdasarkan basis pengetahuan.

4. Konkuren (*parallel programming*)

   Paradigma ini muncul karena *task* (tugas) tidak lagi bisa dilakukan secara beruntun. Ada beberapa tugas yang harus dijalankan secara bersamaan. Paradigma ini mensinkronisasikan antartugas yang berjalan secara bersama-sama.

5. Objek

   Paradigma ini didasari oleh konsep objek, yaitu suatu entitas yang mempunyai atribut (data) dan metode (fungsi) yang dapat berinteraksi dengan objek lainnya. Dalam sebuah permasalahan pasti ada objek-objek yang menyusunnya.

Dalam buku ini kalian akan mempelajari dasar-dasar Pemrograman Berorientasi Objek secara umum dan diimplementasikan dengan Bahasa Pemrograman Java. Kalian dapat menggunakan Integrated Development Environment seperti Netbeans, Eclipse, BlueJ, IntelliJ IDEA, jGRASP, JDeveloper, dan lain-lain untuk mempraktikkannya.

Aktivitas Belajar 6-1

Lakukan secara berkelompok! Deskripsikan benda-benda berikut

1. Koran dan Majalah
2. Televisi dan Monitor
3. Kursi dan Sofa
4. Mobil dan Motor
5. Gelas dan Cangkir

Deskripsi yang dibuat berisi ciri-ciri serta perilaku yang dimiliki oleh benda-benda tersebut. Perhatikan baik-baik dari deskripsi yang kalian buat, dan diskusikan dengan teman sekelompok. Lebih banyak persamaan atau perbedaankah yang muncul?

Dalam pemrograman, segala sesuatu yang bersifat abstrak yang harus didefinisikan menjadi ciri-ciri dan sifat tertentu agar dikenali. Bisa saja kalian menemukan benda yang secara fisik berbeda tetapi secara definisi ciri memiliki kesamaan. Koran dan Majalah secara fisik berbeda, tetapi secara lojik adalah dua hal yang sama yaitu media cetak. Begitu juga benda-benda lainnya. Berdasarkan fakta itulah kemudian muncul istilah Kelas dan Objek. Media cetak adalah kelasnya, Koran dan Majalah adalah wujud fisik yang kita sebut dengan Objek. Namun Koran dan Majalah juga dapat menjadi Kelasnya sedangkan Objeknya adalah wujud fisik dari Koran dan majalah tersebut.

1. Objek dan Kelas

Objek merupakan satuan terkecil yang merupakan komponen pada perangkat lunak seperti halnya objek pada dunia nyata. Sebuah objek memiliki data-data yang menjadi ciri dan mendeskripsikan objek itu sendiri. Data-data tersebut diistilahkan dengan atribut. Objek juga memiliki perilaku yang dijabarkan dalam metode. Metode tersebut dalam pemrograman diwujudkan dalam fungsi.

Kelas merupakan sekumpulan objek yang memiliki karakteristik yang sama. Kelas adalah templat definisi untuk menyusun dan membuat objek dengan atribut dan metode yang sama (Poo, Kiong, & Ashok, 2008). Dalam pemrograman, kelas merupakan struktur dasar dari pemrograman berorientasi objek. Dalam kelas kalian dapat mendefinisikan **atribut** (data) dan **metode** (fungsi).

Objek merupakan instansiasi dari sebuah kelas. Dapat dikatakan objek mirip dengan sebuah variabel yang menggunakan tipe data berbentuk kelas. Objek merupakan elemen yang dapat diciptakan dan dimanipulasi, pada saat eksekusi serta dihancurkan pada saat tidak lagi digunakan. Jadi dengan kata lain, memrogram objek tidak lain adalah menciptakan, memanipulasi, dan menghancurkan objek. Karena kelas merupakan sekumpulan objek, maka sebuah kelas dapat diinstansiasi menjadi beberapa objek.

Atribut merupakan elemen data yang dimiliki oleh objek. Atribut berisi informasi tentang objek atau dapat juga disebut sebagai variabel, data, properti, *data field*. Atribut digunakan sebagai identitas yang membedakan satu objek dengan objek lainnya. Maka dari itu, atribut harus menggunakan tipe data tertentu dan diberi nama yang unik untuk membedakan satu dengan yang lainnya. Misalkan yang menjadi objek adalah mobil, maka dia mungkin memiliki atribut *merek, jenis, warna* yang bisa diisi, seperti:

- Merek: Honda, Toyota, Mitsubishi, Daihatsu
- Jenis: Sedan, SUV, MPV,
- Warna: Hitam, Merah, Putih, Biru

Atribut dari objek dapat juga mencakup informasi keadaan dari objek itu sendiri, misalkan pada mobil dapat didefinisikan atribut untuk kondisi mesin (hidup atau mati) maupun posisi gigi. Atribut dapat dianalogikan sebagai variabel global yang dimiliki oleh objek dan dapat diisi dengan nilai yang berbeda-beda ketika digunakan.

Metode merupakan perilaku yang dimiliki oleh sebuah objek. Perilaku merupakan cara objek melakukan sesuatu terhadap dirinya sendiri. Misalkan pada contoh objek Mobil, ada beberapa perilaku yang bisa dimiliki seperti:

- Menghidupkan mesin
- Maju
- Mundur
- Berbelok
- Berpindah gigi
- Berhenti

Metode dapat dikategorikan dalam beberapa bentuk:

a. *Assesor*, metode yang digunakan untuk memanggil nilai atribut. Umumnya diawali dengan kata kunci set dan diikuti dengan nama atributnya. Contoh: sebuah kelas Kotak memiliki atribut Panjang dan lebar, maka dapat dibuat metodenya, yaitu getPanjang(), getLebar().

b. *Mutator*, metode yang digunakan untuk mengubah atau memberi nilai pada atribut. Umumnya diawali dengan keyword get dan diikuti dengan nama atributnya. Contoh: pada kelas Kotak dapat dibuat metode setPanjang(int), setLebar(int).

c. Konstruktor, metode yang dipanggil saat suatu objek diciptakan (diinstansiasi). Saat objek diciptakan, konstruktor akan mengalokasikan memori, memberi nilai awal pada data yang dipunyai objek, dan secara otomatis menjalankan (mengeksekusi) semua pernyataan yang menjadi isinya. Konstruktor merupakan fungsi anggota yang memiliki kesamaan nama dengan kelasnya dan dideklarasikan sebelum fungsi-fungsi lain. Dalam satu kelas dapat dimungkinkan untuk membuat lebih dari satu konstruktor. Contoh: pada kelas Kotak dapat dibuat konstruktor Kotak(), Kotak(int, int), Kotak(float, float).

d. Program Utama, metode atau fungsi yang akan dipanggil pertama kali ketika program dijalankan. Bentuk program utama pada Java ditulis dengan:

```
public static void main(String[] args){
}
```

e. Metode lainnya, metode selain yang sudah dijelaskan di atas. Contoh: pada kelas Kotak dapat dibuat metode HitungLuas(), HItungKeliling(), HitungDiagonal(), AssignKotak(int, int), DisplayKotak().

Berikut contoh pendefinisian kelas:

Tabel 6.1 Contoh pendefinisian kelas

No	Nama Kelas	Atribut	Metode
1	Kotak	Panjang Lebar	getPanjang() getLebar() setPanjang(int) setLebar(int) HitungLuas() HitungKeliling() HitungDiagonal() AssignKotak(int, int) DisplayKotak()
2	Waktu	Jam Menit Detik	getJam() getMenit() getDetik() setJam(int) setMenit(int) setDetik(int) assignWaktu(int, int, int)
3	Mobil	No Polisi Merek Type Tahun Pembuatan No Rangka No Mesin Bahan Bakar	Melaju() Belok() Berhenti() Mundur()
4	Pegawai	ID Pegawai Nama Tanggal Lahir Tanggal Masuk Divisi	Masuk() Keluar() Mutasi()

Untuk memahami lebih lanjut perbedaan kelas dan objek, perhatikan contoh berikut ini.

Tabel 6.2 Kelas mobil dan objek-objeknya

Kelas: Mobil		Mobil 1	Mobil 2	Mobil 3
Atribut	No. Polisi	B1234YZ	DK1234CK	KT1234BC
	Merek	Mitsubishi	Daihatsu	Toyota
	Type	XPander	Xenia	Avanza
	Tahun Pembuatan	2015	2019	2021
	No. Rangka	MA3GMH31SA1162468	MA3GMF31SA0212345	MA3GMF31KB1232468
	No. Mesin	KX2BN4052285	K10BC4012385	K10BN1234285
	Bahan Bakar	Bensin	Bensin	Bensin
Metode	*Melaju*			
	Belok			
	Berhenti			
	Mundur			

Pada tabel 6.2 dapat dilihat sebuah Mobil 1, Mobil 2, Mobil 3 adalah objek yang tercipta dari kelas Mobil. Ketiganya memiliki atribut yang sama yaitu No. Polisi, Merek, Type, Tahun Pembuatan, No. Rangka, No. Mesin, Bahan Bakar, serta memiliki metode yang sama, yaitu Melaju, Belok, Berhenti, Mundur yang didapat dari kelasnya.

Aktivitas Belajar 6-2

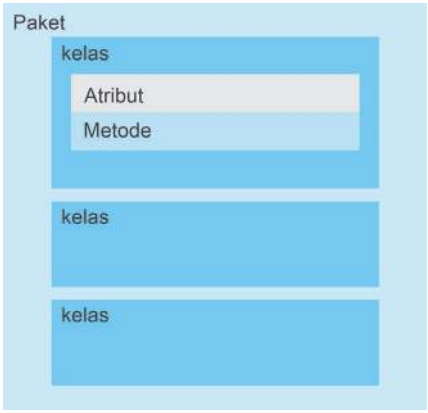
Perhatikan di sekitar ruang kelas kalian, ada objek apa saja yang dapat kalian temukan? Tuliskan dalam format berikut:

No.	Nama Objek	Atribut	Metode
1.	Papan Tulis	LebarTinggiWarna...	DitulisDihapus...
2.	...		
3.	...		

Kelompokkan objek-objek yang kalian temukan, kemudian tentukan termasuk ke dalam kelas apakah objek tersebut?

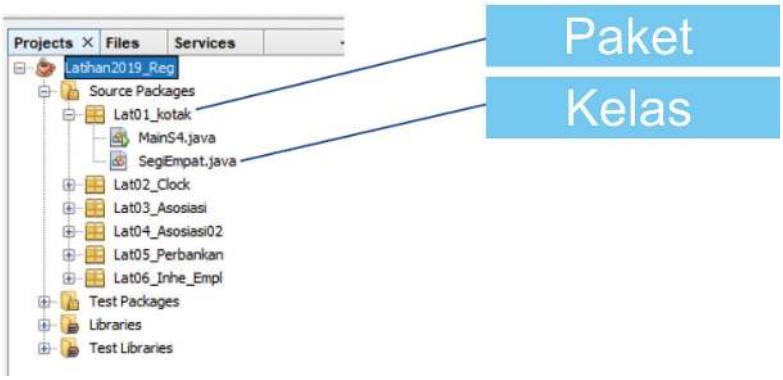
No.	Nama Objek	Kelas
1.	Papan Tulis	...
2.	...	...
3.	...	...

2. Paket



Gambar 6.2 Komposisi pemrograman objek dalam bahasa pemrograman
Sumber: Marwondo & Rini Melati/2022

Perhatikan gambar 6.2, kelas-kelas pada pemrograman berhimpun dalam satuan yang lebih besar yang disebut dengan **Paket** (*package*). Hal ini menandakan bahwa kelas merupakan anggota sebuah paket. Dengan paket dimungkinkan untuk kelas-kelas yang digunakan pada permasalahan yang sama dikelompokkan. Pada dasarnya, paket merupakan direktori-direktori yang disusun sedemikian rupa sehingga memudahkan pengelompokan jenis-jenis kelas.



Gambar 6.3 Contoh Paket
Sumber: Marwondo & Rini Melati/2022

3. Mendefinisikan Kelas

Kelas adalah definisi statik dari objek. Suatu kelas mirip dengan tipe data baru (*struct*), tetapi mempunyai kemampuan untuk menyembunyikan data dan mempunyai layanan. Secara umum, kelas didefinisikan menggunakan *keyword* ***class***. Bentuk umum dari kelas adalah:

```
<<modifier>>class <<NamaKelas>> {
   <<modifier>>atribut;
   <<modifier>>konstruktor(){ }
   <<modifier>>metode(){ }
}
```

Contoh: Sebuah kelas dengan nama Kotak memiliki atribut: *panjang, lebar*, dan memiliki metode *AssignKotak(int, int), Luas(), Keliling()*. Berikut adalah implementasi kelas dalam Bahasa C++ dan Java.

Kode Program 6.1 Contoh Deklarasi Kelas dalam Bahasa C++

```
01 //deklarasi kelas Kotak
02 class Kotak {
03 //deklarasi atribut dengan enkapsulasi private
04 private:
05    int panjang;
06    int lebar;
07
08 //deklarasi method anggota kelas
09 public:
10    //Konstruktor 1
11    Kotak(){
12       cout << "Objek diciptakan..." <<endl;
13       panjang = lebar = 0;
14    };
15    //Konstruktor 2
16    Kotak(int p,int l){
17       cout << "Objek diciptakan..." <<endl;
18       panjang = p;
19       lebar = l;
20    };
21
22    void AssignKotak(int p, int l){
23       panjang=p;
24       lebar=l;
25    }
26
27    long Luas(){
28       return panjang*lebar;
29    }
30
31    long Keliling(){
32       return 2*(panjang+lebar);
33    }
34 }
```

Kode Program 6.2 Contoh Deklarasi Kelas dalam Bahasa Java

```
01 //deklarasi kelas Kotak
02 public class Kotak {
03    //deklarasi atribut dengan enkapsulasi private
04    private int panjang;
05    private int lebar;
```

```
06
07    //konstruktor 1
08    Kotak(){
09    }
10
11    //konstruktor 2
12    Kotak(int p, int l){
13       panjang=p;
14       lebar =l;
15    }
16    //deklarasi method anggota kelas
17    public void AssignKotak(int p, int l){
18       panjang=p;
19       lebar=l;
20    }
21
22    public long Luas(){
23       return panjang*lebar;
24    }
25
26    public long Keliling(){
27       return 2*(panjang+lebar);
28    }
29 }
```

4. Instansiasi Objek

Kelas yang telah didefinisikan belum dapat digunakan jika belum dilakukan instansiasi objeknya. Instansiasi objek mirip dengan deklarasi variabel yang ada pada Java, tetapi ditambah dengan *keyword* ***new***. Instansiasi objek dituliskan sebagai berikut.

```
NamaKelas objek;
```

Contoh instansiasi pada Bahasa Pemrograman

C++	Java
Kotak K1;	Kotak K1=new Kotak();
Kotak K2;	Kotak K2=new Kotak(3,4);

Setelah diinstansiasi menjadi objek, maka semua atribut dan metode yang dimiliki oleh kelas melekat pada objek tersebut. Perhatikan contoh di atas! Ketika K1 dan K2 dideklarasikan sebagai objek dari kelas Kotak, maka K1 memiliki atribut panjang dan lebar yang dimiliki oleh kelas Kotak serta memiliki metode *AssignKotak(int, int), Luas(), Keliling()*. Instansiasi objek bisa dilakukan dalam kelas itu sendiri maupun kelas yang berbeda. Berikut adalah bagaimana kita menginstansiasi objek pada kelas yang berbeda.

Kode Program 6.3 Contoh penggunaan objek pada kelas lain dalam Bahasa C++

```
01  class Utama {
02     void main(){
03        Kotak K1;
04        K1.AssignKotak(3,4);
05        cout<<""\nKotak K1:""<<endl;
06        cout<<""Luas = ""<< K1.Luas()<<endl;
07        cout<<""Keliling = ""<< K1.Keliling()<<endl;
08        getch();
09     }
10  }
```

Kode Program 6.4 Contoh penggunaan objek pada kelas lain dalam Bahasa Java

```
01  public class Main {
02     public static void main(String[] args) {
03        Kotak K1=new Kotak();
04        K1.AssignKotak(3,4);
05        System.out.println ("Kotak K1");
06        System.out.println ("Luas Kotak = " + K1.Luas());
07        System.out.println ("Keliling Kotak = " + K1.Keliling());
08     }
09  }
```

Setelah diinstansiasi menjadi objek, tentu atribut atau metode yang ada perlu digunakan atau dipanggil agar dapat bekerja. Untuk memanggil anggota kelas maka ditulis:

```
Objek.anggotakelas
```

Contoh untuk memanggil method Luas() dan Keliling() pada kelas Kotak

```
K1.Luas();
K1.Keliling(;
```

Bagaimana penerapannya pada program? Perhatikan contoh kode program 6.3 dan 6.4 pada baris 04 sampai 07!

Aktivitas Belajar 6-3

Lakukan implementasi contoh ke dalam bahasa pemrograman Java. Gunakan IDE yang kalian kenal

- Buat kelas baru dengan nama kelas Kotak, salin kode program 6-2
- Tambahkan kelas baru yang digunakan untuk kelas utama, salin kode program 6-4
- Jalankan program utama, amati hasil keluarannya.

B. *Access Modifier* dan Enkapsulasi

1. *Access Modifier*

Access Modifier merupakan salah satu fitur pada pemrograman berorientasi objek yang digunakan untuk melakukan penyembunyian data (*data hiding*). Dengan fitur ini, kalian dapat mengatur mana saja atribut atau metode yang boleh diakses dan tidak. Secara *default* semua atribut dan metode dapat dilihat di semua kelas dalam satu package yang sama. Untuk memodifikasi visibilitas atribut dan metode, Java menyediakan empat pengubah akses (hal ini bisa berbeda pada bahasa pemrograman yang lain).

- ***Private***: atribut dan metode yang dinyatakan *private* hanya dapat diakses oleh metode di dalam kelasnya sendiri dan tidak dapat diakses kelas lain meskipun dalam package yang sama.
- ***Protected***: atribut dan metode dapat diakses oleh semua kelas dari package saat ini serta subkelas (turunan) dalam package lain. Namun, selain sub kelas yang ada dalam package lain tidak dapat mengakses.
- ***Public***: atribut dan metode yang dideklarasikan sebagai *public* dapat diakses oleh semua kelas di semua program.
- ***Default***: jika atribut dan metode tidak ditentukan tingkat akses apa pun itu, akan menjadi default. *Default* berarti dapat diakses oleh semua kelas dalam satu package.

Umumnya, atribut akan dinyatakan *private* sedangkan metode dinyatakan *public*.

Aktivitas Belajar 6-4

Buka kembali kode program yang telah dibuat pada aktivitas 6-3. Ubah modifier metode Luas() pada kelas Kotak menjadi **private**. Jalankan kembali program utama. Amati dengan baik apa yang terjadi! Uraikan penyebab mengapa hal itu bisa terjadi!

2. Enkapsulasi

Enkapsulasi merupakan sebuah konsep dalam pemrograman berorientasi objek yang menekankan pada pembungkusan data dan fungsi. Enkapsulasi memungkinkan data atau fungsi diatur sedemikian rupa

sehingga objek yang lain tidak perlu mengetahui cara kerja pada objek tersebut tetapi dapat memanfaatkan apa yang dimiliki oleh objek melalui metodenya. Enkapsulasi menyebabkan terjadinya abstraksi dan "*data hiding*" dalam pemrograman objek.

Dalam praktik dunia nyata, enkapsulasi ini dapat dianalogikan seperti sebuah Remot TV. Pada remot, yang terlihat hanya tombol-tombol yang memiliki berbagai fungsi. Pengguna tidak perlu tahu bagaimana cara kerja tombol-tombol tersebut, tetapi cukup mengetahui apa yang terjadi ketika sebuah tombol ditekan.

Pada bagian sebelumnya, kalian telah mengetahui tentang ***Access Modifier***, kan? Ya, enkapsulasi tidak lepas juga dari penggunaan ***Modifier***. Kalian juga sudah mengetahui jenis-jenis metode pada sebuah kelas, kan? Metode apa yang digunakan untuk mengakses atau mengubah atribut? Ya, ada **assesor** dan **mutator** yang bisa digunakan. Enkapsulasi dilakukan dengan menggunakan kedua jenis metode tersebut. Atur atribut menjadi *private* sedangkan assesor dan mutator menjadi *public*. Dengan demikian, objek lain tidak dapat langsung mengakses atribut tetapi harus melalui assesor dan mutator.

Aktivitas Belajar 6-5

Buka kembali kode program yang telah dibuat pada aktivitas 6-4. Ubah sedikit kelas utama menjadi:

```
01 public class Main {
02    public static void main(String[] args) {
03       Kotak K1=new Kotak();
04       K1.panjang=3;
05       K1.panjang=4;
06       System.out.println ("Kotak K1");
07       System.out.println ("Panjang = " + K1.panjang);
08       System.out.println ("Lebar = " + K1.lebar);
09
10       System.out.println ("Luas Kotak = " + K1.Luas());
11       System.out.println ("Keliling Kotak = " + K1.Keliling());
12    }
13 }
```

Perhatikan baris 4, 5, 7, dan 8. Jalankan program utama dan amati apa yang terjadi.

C. Pewarisan

Aktivitas Belajar 6-6

1. Deskripsikan bangun berikut:
   a. Prisma segitiga, balok, tabung
   b. Limas segiempat, kerucut

   Perhatikan hasil deskripsi kalian, apakah kesamaan ciri yang dimiliki oleh masing-masing kelompok?
2. Tuliskan sifat-sifat yang dimiliki oleh beberapa hal berikut:
   (a) Makhluk
   (b) Hewan
   (c) Kuda
   (d) Burung

   Apakah sifat yang dimiliki oleh (a) dimiliki juga oleh (b)? Apakah sifat yang dimiliki oleh (b) dimiliki juga oleh (c) dan (d)?
3. Perhatikan silsilah dalam keluarga kalian, apakah ada ciri-ciri fisik ayah atau ibumu yang kamu miliki? Sebutkan ciri-ciri fisik yang manakah itu?

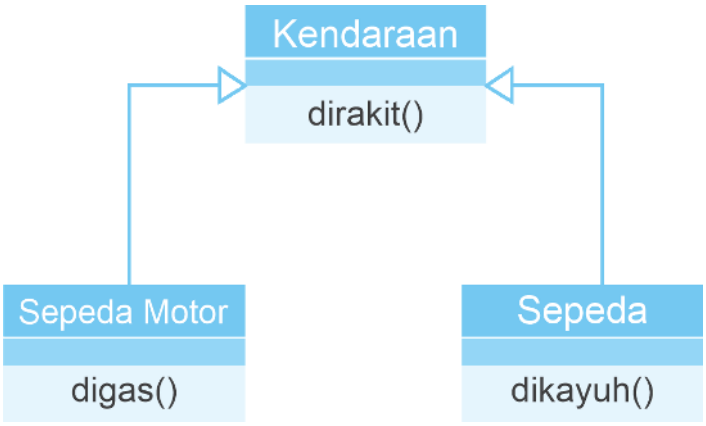
1. Pewarisan

Pewarisan (*inheritance*) merupakan kemampuan sebuah kelas menurunkan atau mewariskan sifat-sifat yang dimilikinya kepada kelas yang lain. Sifat-sifat yang dimaksud adalah atribut dan metode. Kelas yang mewariskan sifat dinamakan kelas induk atau kelas dasar (*base* class) sedangkan yang menerima sifat-sifat yang diwariskan disebut kelas turunan (*sub* class). Kelas turunan tidak perlu mendefinisikan kembali atribut dan metode yang ada pada kelas induk. Cukup atribut dan metode yang tidak ada pada kelas induk yang didefinisikan pada kelas turunan. Atribut dan metode yang bisa diwariskan adalah selain yang dinyatakan *private*.

Secara umum mekanisme pewarisan ini dikelompokkan menjadi:

a. Pewarisan **Tunggal** (***Single Inheritance***)

   Pewarisan berasal dari satu kelas induk. Meskipun kelas induknya hanya satu, tetapi dapat membentuk beberapa kelas turunan. Perhatikan gambar 6.4 berikut.

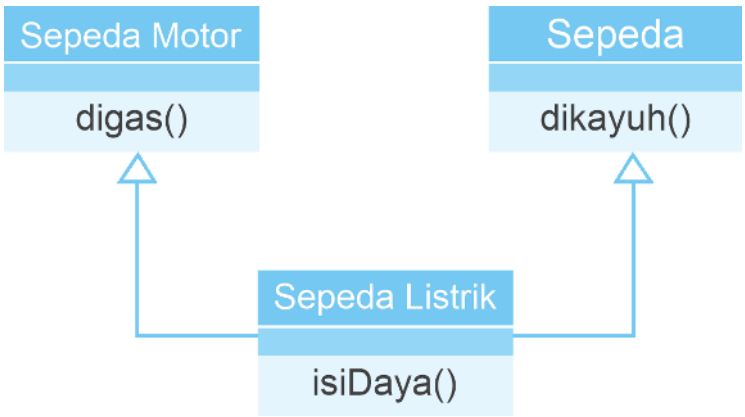


Gambar 6.4 Contoh pewarisan tunggal
Sumber: Marwondo & Rini Melati/2022

Kelas `Kendaraan` merupakan kelas induk yang memiliki kelas turunan `SepedaMotor` dan kelas `Sepeda`. Pada kelas `SepedaMotor`, selain memiliki sifat `digas()` juga memiliki sifat `dirakit()` yang merupakan sifat yang dimiliki `Kendaraan`. Begitu juga dengan kelas `Sepeda`, selain memiliki sifat `dikayuh()` juga memiliki sifat `dirakit()` yang berasal dari `Kendaraan`.

b. Pewarisan **Jamak** (***Multiple Inheritance***)

Merupakan mekanisme pewarisan yang berasal dari dua atau lebih kelas induk. Beberapa kelas induk dapat membentuk satu kelas baru yang mewarisi sifat dari beberapa kelas induk. Perhatikan gambar 6.5 untuk memperjelas.

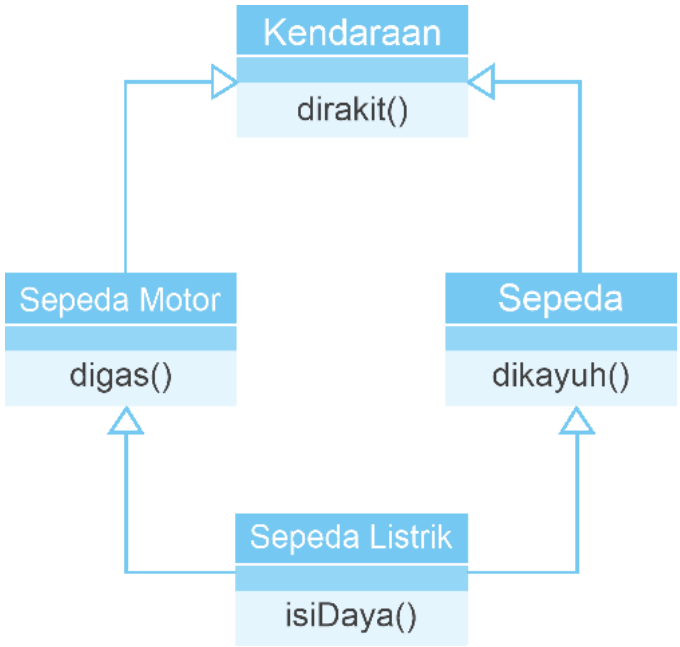


Gambar 6.5 Contoh pewarisan jamak
Sumber: Marwondo & Rini Melati/2022

`SepedaListrik` merupakan kelas turunan dari kelas `SepedaMotor` dan kelas `Sepeda`. Oleh karena itu, `SepedaListrik` selain memiliki sifat `isiDaya()` juga memiliki sifat `digas()` yang berasal dari kelas `SepedaMotor` dan sifat `dikayuh()` yang berasal dari kelas `Sepeda`.

c. Pewarisan **Jamak Maya** (***Virtual Multiple Inheritance***)

Mekanisme pewarisan ini ada karena kelas induk merupakan kelas turunan yang berasal dari induk yang sama (pewarisan jamak) sehingga ada sifat yang sama menjadi berulang pada satu kelas yang sama. Perhatikan gambar 6.6!



Gambar 6.6 Contoh pewarisan jamak maya
Sumber: Marwondo & Rini Melati/2022

`Sepeda` dan `SepedaMotor` merupakan kelas turunan dari `Kendaraan`, akibatnya keduanya memiliki sifat yang sama yang diwarisi dari `Kendaraan`, yaitu `dirakit()`. `SepedaListrik` merupakan turunan dari `Sepeda` dan `SepedaMotor` dan sama-sama mewariskan sifat `dirakit()`. Akibatnya, sifat `dirakit()` "bentrok" pada kelas `SepedaListrik`.

Bahasa Java hanya mengenal pewarisan tunggal, tidak mengenal pewarisan jamak, sehingga dengan sendirinya tidak pernah, terjadi pewarisan jamak maya.

Pada pewarisan, kelas yang dibuat pertama adalah kelas induk diikuti dengan kelas-kelas turunan. Kata kunci ***extends*** sesudah nama kelas digunakan untuk menunjukkan bahwa kelas tersebut adalah turunan dari kelas yang disebutkan sesudah *extends*.

```
class KelasTurunan extends KelasInduk{
}
```

Contoh penggunaan

```
public class Makhluk{
}
public class Hewan extends Makhluk{
}
public class Kuda extends Hewan{
}
```

Aktivitas Belajar 6-7

Lakukan praktik dalam Bahasa Java. Ikuti langkah-langkah berikut.

1. Buat kelas **Sepeda** dengan rincian berikut.

Kode Program 6.5 Kelas Induk Sepeda

```
01 public class Sepeda{
02    public int gear;
03    public int kecepatan;
04
05    //Konstruktor
06    public Sepeda(int gear, int kecepatan)
07    {
08       this.gear = gear;
09       this.kecepatan = kecepatan;
10    }
11
12    public void rem(int kurang)
13    {
14       kecepatan -= kurang;
15    }
16
17    public voidtambahCepat(int tambah)
18    {
19       kecepatan += tambah;
20    }
21
22    public String info()
23    {
24       return ("Sepeda ada di gear nomor" + gear + "\n"
25             + "dengan kecepatan" + kecepatan);
26    }
27 }
```

2. Buat Kelas **SepedaGunung** yang merupakan turunan dari **Sepeda**.

Kode Program 6.6 Kelas SepedaGunung turunan Sepeda

```
01 public class SepedaGunung extends Sepeda {
02    public int tinggiDudukan;
03
04    //konstruktor
05    public SepedaGunung(int gear, int kecepatan,
06                        int tinggiAwal)
07    {
08        super(gear, kecepatan);
09        tinggiDudukan = tinggiAwal;
10    }
11
12    public void setTinggiDudukan (int y)
13    {
14        tinggiDudukan = y;
15    }
16    public int getTinggiDudukan ()
17    {
18        return tinggiDudukan;
19    }
20
21    public String info()
22    {
23        return (super.info() + "\tinggi dudukan"
24               + getTinggiDudukan());
25    }
26 }
```

3. Buat kelas utama yang digunakan untuk mengoperasikan kelas **SepedaGunung**.

Kode Program 6.7 Kelas utama menjalankan Pewarisan

```
01 public class CobaWaris{
02    public statisc void main(String[] args){
03        SepedaGunung sg=new SepedaGunung(3,55,25);
04        System.out.println(sg.info());
05    }
06 }
```

4. Jalankan kelas utama dan amati bagaimana keluarannya?
5. Lakukan perubahan, tambahkan kode program pada kelas utama untuk:
   a. Menambah kecepatan (gunakan metode tambahCepat()
   b. Mengurangi kecepatan (gunakan metode rem()
   c. Mengisi nilai gear, kecepatan, dan tinggiDudukan secara dinamis

2. Kelas Abstrak

Kelas Abstrak merupakan salah satu karakteristik yang dimiliki oleh pemrograman berorientasi objek. Kelas ini tidak dapat diisntansiasi seperti kelas pada umumnya. Kelas abstrak digunakan untuk mengekspresikan atribut dan metode yang bersifat umum yang dapat digunakan oleh kelas-kelas turunannya (Eck, 2021). Jadi kelas abstrak sebenarnya adalah kelas induk yang berisi sifat-sifat dasar secara umum. Instansiasi dilakukan oleh kelas-kelas turunannya.

Kelas abstrak dideklarasikan menggunakan *keyword* ***abstract***. Bentuk umum kelas abstrak dapat dituliskan sebagai berikut:

```
public abstract class NamaKelasAbstrak {
   //deklarasi atribut dan method abstract
   //definisi method tidak abstrak
}
```

Karena kelas abstrak tidak dapat diinstansiasi dan hanya bisa dilakukan pada kelas turunan, maka kalian harus membuat kelas turunan sama halnya dengan pewarisan biasa.

```
public abstract class NamaKelasAbstrak {
   //deklarasi atribut dan method abstract
   //definisi method tidak abstrak
}
```

Kelas abstrak dapat memiliki metode abstrak (hanya deklarasi) dan non-abstrak (metode dengan definisinya). Perhatikan contoh berikut.

```
01 public abstract class MakhlukHidup {
02    public void bernafas(){
03       System.out.println("Makhluk hidup bernafas.");
04    }
05
06    public void makan(){
07       System.out.println("Makhluk hidup perlu makan.");
08    }
09    public abstract void bergerak(); //method abstrak
10 }
```

Ketika sebuah kelas mewarisi sebuah kelas abstrak, kelas tersebut harus mendefinisikan implementasi dari metode abstrak yang ada dalam kelas induknya. Jika tidak didefinisikan implementasinya, maka kelas tersebut akan menjadi kelas abstrak juga, contoh: Sebuah kelas turunan yang mewarisi kelas abstrak harus mendefinisikan metode yang dibuat abstrak jika akan menggunakannya. Hal ini dilakukan agar

kelas turunan tidak berubah menjadi kelas abstrak. Perhatikan contoh penggunaan berikut.

```
01 public class Manusia extends MakhlukHidup {
02 //implementasi dari method abstrak walk()
03 public void bergerak(){
04 System.out.println("Manusia bergerak");
05 }
06 }
```

Aktivitas Belajar 6-8

Lakukan praktik dalam Bahasa Java. Ikuti langkah-langkah berikut.

1. Buat kelas **Hewan** dengan rincian berikut.

Kode Program 6.8 Kelas Abstrak Hewan

```
01 abstract class Hewan{
02    String nama;
03    public abstract void habitatHewan();
04    public void namaHewan(){
05       System.out.println("\n Method di dalam abstract
06          class Hewan");
07       System.out.println("Nama hewan: "+nama);
08    }
09 }
```

2. Buat kelas **Karnivora** dengan seperti kode berikut.

Kode Program 6.9 Kelas Karnivora turunan Hewan

```
01 class Karnivora extends Hewan{
02    String habitat;
03    public void habitatHewan(){
04       System.out.println("\n Method di dalam class
05          Karnivora");
06       System.out.println("Habitat hewan: " +habitat);
07    }
08 }
```

3. Buat Kelas **Utama** untuk menjalankannya.

Kode Program 6.10 Kelas Utama menjalankan kelas Abstrak

```
01 public class CobaAbstrak1{
02    public static void main(String[] args){
03       Karnivora lion=new Karnivora();
04       lion.nama=" Asgard"
05       lion.habitat="Hutan"
06       System.out.println("Penggunaan Kelas Abstrak);
07       lion.namaHewan();
08       lion.habitatHewan();
09    }
10 }
```

4. Jalankan kelas utama dan amati bagaimana keluarannya?
5. Lakukan perubahan:
   a. Ubah metode namaHewan() menjadi metode *abstract*, bagaimana hasil eksekusinya? Beri penjelasan singkatnya!
   b. Jika method habitatHewan() tidak didefinisikan pada kelas Karnivora, bagaimana hasil eksekusinya? Jelaskan secara singkat!
   c. Apa yang terjadi jika metode *abstract* habitatHewan() dideklarasikan dalam *class* Karnivora? Beri penjelasan singkat!

3. *Interface*

Interface muncul sebagai jawaban atau solusi dari bentuk pewarisan jamak. Tidak semua bahasa pemrograman mendukung mekanisme pewarisan jamak. Java termasuk bahasa pemrograman yang tidak memungkinkan melakukan pewarisan majemuk. Interface di java merupakan salah satu mekanisme yang memungkinkan kalian untuk berbagi konstanta atau metode yang dapat digunakan oleh sejumlah kelas yang lainnya. Dalam *interface* berisi definisi konstanta-konstanta serta deklarasi metode-metode (hanya nama metode beserta parameternya) yang nantinya didefinisikan dalam kelas yang menggunakannya. Atribut dinyatakan sebagai *public*, *static*, dan *final* sehingga bertindak sebagai konstanta. Semua metode dinyatakan *abstract* dan *public* dan tidak boleh ada konstruktor yang menyertai.

Setiap bahasa pemrograman biasanya sudah menyediakan *interface* untuk diimplementasikan oleh pemrogram. Namun pada saat melakukan pemrograman kadang ada juga yang harus kalian buat sendiri. Untuk membuat *interface* sendiri, secara umum bentuk penulisannya sebagai berikut.

```
public interface NamaInterface {
   //definisi konstanta
   //deklarasi abstract method
}
```

Kata kunci yang digunakan bukan ***class*** tetapi ***interface*** meskipun strukturnya mirip dengan kelas. Untuk menggunakan mirip dengan mendeskripsikan kelas turunan hanya saja kata kunci yang digunakan bukan ***extends*** melainkan ***implement***. Bentuk penulisannya sebagai berikut.

```
class NamaKelasPengguna implements NamaInterface {
   //penggunaan konstanta
   //pendefinisian abstract method
}
```

Setelah sebuah kelas mengimplementasikan *interface*, kelas tersebut juga dapat melakukan pewarisan. Cara melakukan pewarisan sama saja dengan mewariskan kelas seperti yang sudah dijelaskan sebelumnya.

```
public interface Interface_Y extends Interface_X {
   //definisi konstanta
   //deklarasi abstract method
}
```

Aktivitas Belajar 6-9

Lakukan praktik dalam Bahasa Java. Ikuti langkah-langkah berikut.

1. Buat *interface* operasi seperti pada kode program berikut.

Kode Program 6.11 Interface Operasi

```
01 interface Operasi {
02    double kons_pi=3.14;
03    String kons_panjang = " cm";
04    void kelilingLingkaran(double radius);
05    void kelilingPersegi();
06 }
```

2. Buat kelas yang mengimplementasian operasi.

Kode Program 6.12 Kelas Hitung implement Interface Operasi

```
01 class Hitung implements Operasi{
02    double lingkaran, persegi;
03    double sisi=5;
04
05    public void kelilingLingkaran(double radius){
06       System.out.println("Menghitung Keliling Lingkaran");
07       System.out.println("Nilai radius = "
08          +radius+kons_panjang);
09       lingkaran=kons_pi*2*radius;
10       System.out.println("Keliling lingkaran = "
11          + lingkaran + kons_panjang);
12    }
13    public void kelilingPersegi(){
14       System.out.println("Menghitung Keliling Persegi");
15       System.out.println("Nilai sisi = "
16          +sisi+kons_panjang);
17       persegi=4*sisi;
18       System.out.println("Keliling persegi = " + persegi
19          + kons_panjang);
20    }
21 }
```

3. Buat kelas **Utama** untuk menjalankannya.

Kode Program 6.13 Kelas Utama untuk menjalankan Interface Operasi

```
01 public void CobaInterface1{
02    public static void main(String[] args){
03
04       Hitung h=new Hitung();
05       h.kelilingLingkaran(6);
06       h.kelilingPersegi();
07    }
08 }
```

Aktivitas Belajar 6-10

1. Buat kelas **Berjalan** seperti pada kode program berikut.

Kode Program 6.14 Interface Berjalan

```
01 public interface Berjalan {
02    void jalan();
03 }
```

2. Buat kelas **Person** yang mengimplementasikan kelas **Berjalan**.

Kode Program 6.15 Kelas Orang implementasi Interface Berjalan

```
01 public class Orang implements Berjalan {
02    private String nama;
03    public Orang(String nama) {
04       this.nama = nama;
05    }
06    public void jalan() {
07       System.out.println(nama +
08             "(seseorang) sedang berjalan.");
09    }
10 }
```

3. Buat kelas **Bebek** yang mengimplementasikan kelas **Berjalan**.

Kode Program 6.16 Kelas Duck implementasi Interface Walkable

```
01 public class Bebek implements Berjalan {
02    private String nama;
03    public Bebek(String nama) {
04       this.nama = nama;
05    }
06    public void jalan() {
07       System.out.println(nama +
08       "(seekor bebek) sedang berjalan.");
09    }
10 }
```

4. Buat kelas **Utama** untuk menjalankannya,

Kode Program 6.17 Kelas Utama untuk Menjalankan Orang dan Bebek

```
01 public void CobaInterface2{
02    public void static main(String[] args){
03       Orang o = new Orang("Arman");
04       Bebek b = new Bebek("Donal");
05
06       o.jalan();
07       b.jalan();
08    }
09 }
```

D. Polimorfisme

Polimorfisme (*polymorphism*) erat kaitannya dengan Pewarisan. Polimorfisme merupakan salah satu kemampuan pemrograman berorientasi objek yang memungkinkan suatu kelas memiliki beberapa bentuk yang berbeda. Bentuk yang dimaksud adalah kesamaan nama metode dan parameter tetapi memiliki definisi yang berbeda-beda di dalamnya. Secara umum, polimorfisme memiliki dua bentuk dasar yaitu polimorfisme *static* dan polimorfisme *dynamic*. Polimorfisme *static* dilakukan dengan menggunakan teknik ***overloading*** sedangkan polimorfisme *dynamic* menggunakan teknik ***overriding***.

Overloading terjadi manakala sebuah kelas memiliki bebebrapa metode dengan nama yang sama tetapi dengan parameter yang berbeda. Pemanfaatan *overloading* yang paling gampang kalian jumpai adalah pada konstruktor, perhatikan contoh kode program berikut.

Kode Program 6.18 Kelas Kotak *Overloading*

```
01 //deklarasi kelas Kotak
02 public class Kotak {
03    //deklarasi atribut dengan enkapsulasi private
04    private int panjang;
05    private int lebar;
06
07    //konstruktor 1
08    Kotak(){
09    }
10
11    //konstruktor 2
12    Kotak(int p, int l){
13       panjang=p;
14       lebar =l;
15    }
16
17    //deklarasi method anggota kelas
18    public  void assignKotak(int p, int l){
```

```
19          panjang=p;
20          lebar=l;
21       }
22
23       public long Luas(){
24          return panjang*lebar;
25       }
26
27       public long Keliling(){
28          return 2*(panjang+lebar);
29       }
30    }
```

Pada contoh di atas, keyword Kotak digunakan pada tiga definisi yang berbeda yaitu Kotak sebagai kelas, Kotak sebagai konstruktor 1 dan Kotak sebagai konstruktor 2.

Overloading juga dapat digunakan pada metode lainnya, kalian dapat membuat metode dengan nama yang sama tetapi memiliki deskripsi atau definisi yang berbeda dengan memberikan parameter yang berbeda seperti di bawah ini:

- Metode Gambar dengan parameter menggambar titik

```
Gambar(int x,int y)
```

- Metode Gambar dengan parameter untuk menggambar garis

```
Gambar(int x1,int y1, int x2,int y2)
```

- Metode Gambar dengan parameter untuk menggambar segitiga

```
Gambar(int x1,int y1, int x2,int y2)
```

- Metode Gambar dengan parameter untuk menggambar segiempat

```
Gambar(int x1,int y1, int x2,int y2, int x3,int y3, int x4, y4)
```

Overloading ini dapat terjadi pada kelas yang sama atau pada suatu kelas induk dan kelas turunannya. Ciri-ciri overloading adalah:

1. nama metode harus sama
2. parameter harus berbeda
3. tipe return bisa sama, bisa juga berbeda

Aktivitas Belajar 6-11

1. Buatlah kelas **Segi Empat** seperti rincian berikut.

Kode Program 6.19 Kelas Segi Empat *Overloading*

```
01 public class SegiEmpat{
02    int x1=0;
03    int x2=0;
```

```
04    int y1=0;
05    int y2=0;
06
07    public void segiEmpat(int x1, int y1, int x2, int y2){
08       this.x1=x1;
09       this.y1=y1;
10       this.x2=x2;
11       this.y2=y2;
12    }
13
14    public void segiEmpat(Point topLeft,Point
15       bottomRight){
16       x1=topLeft.x;
17       y1=topLeft.y;
18       x2=bottomRight.x;
19       y2=bottomRight.y;
20    }
21
22    public void segiEmpat(Point topLeft,int w,int h){
23       x1=topLeft.x;
24       y1=topLeft.y;
25       x2=x1+w;
26       y2=y1+h;
27    }
28
29    public void cetakSegiEmpat(){
30       System.out.println("Segi Empat <"+x1+","
31       +y1 +","+x2+","+y2+">");
32    }
33 }
```

2. Buat kelas **Utama** untuk menjalankannya.

Kode Program 6.20 Kelas Utama Segiempat *Overloading*

```
01 public class Utama{
02    public static void main(String[] args){
03       SegiEmpat rect=new SegiEmpat();
04       System.out.println("Buat segi empat dengan
05          koordinat"+ (25,25) dan (50,50)");
06       rect.segiEmpat(25,25,50,50);
07       rect.cetakSegiEmpat();
08       System.out.println();
09       SegiEmpat rect=new SegiEmpat();
10       System.out.println("Buat segi empat dengan point"
11          + (10,10) dan (20,20)");
12       rect.segiEmpat(new Point(10,10),new Point(20,20));
13       rect.cetakSegiEmpat();
14       System.out.println();
15       SegiEmpat rect=new SegiEmpat();
16       System.out.println("Buat segi empat dengan poin"
17          + (10,10) dan ukuran(50,50)");
```

```
18          rect.segiEmpat(new Point(10,10),50,50);
19          rect.cetakSegiEmpat();
20
21      }
22  }
```

Overriding merupakan suatu mekanisme pada pemrograman objek yang bisa terjadi pada pewarisan. Intinya, overriding berarti menulis ulang deskripsi dari sebuah metode yang ada pada kelas induk dalam kelas turunannya. Karena menulis ulang metode, maka nama, parameter, maupun tipe keluaran harus sama, hanya deskripsinya yang berbeda. Perhatikan contoh terjadinya *Overriding* berikut.

Kode Program 6.21 Kode Program 6.21 Kelas *Overriding*

```
01  public class Parent{
02     public void info(){
03        System.out.println(”ini kelas Parent”);
04     }
05  }
06
07  public class Child extends Parent{
08     public void info(){
09        System.out.println(”ini kelas Child”);
10     }
11  }
```

Pada contoh di atas, metode info() pada kelas “*Child*” melakukan *override* terhadap metode info() di kelas “*Parent*”.

Aktivitas Belajar 6-12

1. Buatlah kelas **Binatang** dengan rincian berikut.

Kode Program 6.22 Kelas Binatang *Overriding*

```
01  class Binatang{
02     public void info() {
03        System.out.println(" Info tentang Hewan: ");
04     }
05  }
```

2. Buatlah kelas **Herbivora** yang merupakan turunan dari kelas **Binatang.**

Kode Program 6.23 Kelas Herbivora *Overriding*

```
01  class Herbivora extends Binatang {
02     public void info() {
03        System.out.println ("Info pada herbivora: Memakan
04           makanan berupa tumbuh - tumbuhan");
05     }
06  }
```

3. Buatlah kelas **Kelinci** yang merupakan turunan dari kelas **Herbivora.**

Kode Program 6.24 Kelas Kelinci *Overriding*

```
01 class Kelinci extends Herbivora{
02    public void info(){
03       System.out.println("Info pada Kelinci: Memakan
04          makanan berupa wortel");
05    }
06 }
```

4. Buat kelas **Utama** untuk menjalankannya.

Kode Program 6.25 Kelas Utama Polimorfisme

```
01 public class UtamaPoli1 {
02    public static void main(String[] args)
03    {
04       Herbivora herb;
05       Kelinci kelinciku;
06       Binatang hewan;
07
08       herb=new Herbivora();
09       kelinciku=new Kelinci();
10       hewan=herb;
11       hewan.info();
12       hewan=kelinciku;
13       hewan.info();
14    }
15 }
```

Aktivitas Belajar 6-13

1. Buat kelas **Kendaraan** seperti rincian berikut.

Kode Program 6.26 Kelas Kendaraan *Overriding*

```
01 public class Kendaraan {
02    private String model;
03    public kendaraan (String model){
04       this.model = model;
05    }
06    public void informasi(){}
07 }
```

2. Buat kelas **Pesawat** yang merupakan turunan dari **Kendaraan**.

Kode Program 6.27 Kelas Pesawat *Overriding*

```
01 public class Pesawat extends Kendaraan{
02    private String nama;
03    private String jenis;
04    public Pesawat(String nama){
05       super("pesawat");
06       this.nama = nama;
```

```
07        jenis = "belum teridentifikasi";
08     }
09     public Pesawat (String nama, String jenis){
10        super("pesawat");
11        this.nama=nama;
12        this.jenis=jenis;
13     }
14     public void informasi(){
15        System.out.println("nama pesawat adalah"+nama);
16        System.out.println("jenis pesawat adalah"+jenis);
17     }
18  }
```

3. Buat kelas **Mobil** yang merupakan turunan dari **Kendaraan**.

Kode Program 6.28 Kelas Mobil *Overriding*

```
01  public class Mobil extends Kendaraan{
02     private String nama;
03     private String jenis;
04     public Mobil (String nama){
05        super("mobil");
06        this.nama=nama;
07        jenis="belum teridentifikasi";
08     }
09     public Mobil (String nama, String jenis){
10        super("mobil");
11        this.nama=nama;
12        this.jenis=jenis;
13     }
14     public void informasi(){
15        System.out.println("nama mobil adalah"+nama);
16        System.out.println("jenis mobil adalah"+jenis);
17     }
18  }
```

4. Buat kelas **Kapal** yang merupakan turunan dari **Kendaraan**.

Kode Program 6.29 Kelas Kapal *Overriding*

```
01  public class Kapal extends Kendaraan{
02     private String nama;
03     private String jenis;
04     public Kapal (String nama){
05        super("kapal");
06        this.nama=nama;
07        jenis="belum teridentifikasi";
08     }
09     public Kapal (String nama, String jenis){
10        super("kapal");
11        this.nama=nama;
12        this.Jenis=jenis;
```

```
13      }
14      public void informasi(){
15          System.out.println("Nama kapal adalah " +nama);
16          System.out.println("Jenis kapal adalah " +jenis);
17      }
18  }
```

5. Buat kelas **Utama** untuk menjalankannya.

Kode Program 6.30 Kelas Utama Kendaraan

```
01  public class Main{
02      public static void main(String[] args){
03          Kendaraan p;
04          Pesawat psw = new Pesawat("Boeing 737", "Pesawat
05              Komersil");
06          Mobil mbl1 = new Mobil("Mitsubhisi Pajero", "SUV");
07          Mobil mbl2 = new Mobil("Honda City", "Sedan");
08          Mobil mbl3 = new Mobil("VW Combi");
09          Kapal kpl = new Kapal("Queen Mary 2", "Kapal Pesiar");
10
11          p=psw;
12          p.informasi();
13          p=mbl1;
14          p.informasi();
15          p=mbl2;
16          p.informasi();
17          p=mbl3;
18          p.informasi();
19          p=kpl;
20          p.informasi();
21      }
22  }
```

Uji Kompetensi

1. Ujian Tulis

Baca baik-baik pertanyaan yang ada, jawab dengan singkat dan jelas pada lembar jawaban yang disediakan.

Amati penggalan kode program berikut:

Kode Program 6.31 Kelas Segitiga

```
01  public class Segitiga{
02      int alas;
03      int tinggi;
04
05      Segitiga(){
06      }
```

```
07
08      Segitiga(int a, int t){
09          alas=a;
10          tinggi=t;
11      }
12
13      void hitLuas(){
14          double hitLuas = 0.5*alas*tinggi;
15      }
16  }
```

Kode Program 6.32 Kelas Utama Segitiga

```
01  public class UtamaS3{
02      public static void main(String[] args) {
03          Segitiga s=new Segitiga();
04          s.alas=3;
05          s.tinggi=4;
06          System.out.println("Luas Segitiga = " +
07          s.hitLuas());
08      }
09  }
```

1. Kelas UtamaS3 merupakan kelas utama yang akan dieksekusi pertama kali saat progam dijalankan. Amati dengan seksama, apakah kelas UtamaS3 dapat dijalankan? Jika ya, bagaimana keluarannya? Jika tidak, mengapa?

   .....

   .....

2. Perhatikan baris 13 kelas Segitiga. Baris tersebut menunjukkan metode yang dimiliki oleh kelas Segitiga dengan nilai "*void*". Nilai "*void*" inilah yang menyebabkan metode tersebut tidak dapat dipanggil pada kelas utama. Bagaimana seharusnya kode program pada metode tersebut agar dapat dieksekusi oleh kelas UtamaS3?

   .....

   .....

3. Atribut dan metode pada kelas Segitiga belum diatur *acces modifier*- nya sehingga belum bisa dilakukan enkapsulasi dengan baik. Seharusnya setiap atribut dan metode ditentukan *acces modifier* sesuai dengan tingkat visibilitas yang dikehendaki. Berdasarkan aturannya, bagaimana *access modifier* yang paling tepat untuk atribut dan metode pada kelas Segitiga?

   .....

   .....

4. Jika access modifier pada atribut kelas Segitiga diubah menjadi ***private***, maka kelas lain tidak akan dapat memanggil maupun memanipulasi atribut tersebut. Ada beberapa bentuk metode yang bisa digunakan untuk memanggil maupun memanipulasinya. Jelaskan bagaimana cara memberi nilai untuk atribut-atribut tersebut jika digunakan oleh kelas lainnya?

   .....

   .....

5. Pada kelas UtamaS3 akan ditambahkan sebuah objek baru dengan nama s2 dari kelas Segitiga. Objek tersebut langsung dilakukan inisialisasi untuk nilai atribut ”alas” dan ”tinggi” pada saat objek diciptakan. Misalkan alas=5 dan tinggi =8. Bagaimana kode program pada kelas UtamaS3 untuk mengakomodir kedua objek tersebut serta menampilkan luas keduanya?

   .....

   .....

2. Ujian Praktik

Diketahui sebuah kelas Pegawai memiliki deskripsi berikut.

Kelas
Pegawai
Atribut
String name
long salary
Date birthday
Metode
getName()
setName(String name)
getSalary()
setSalary(long salary)
getBirthday()
setBirthday(Date birthday)
getUsia()

Ket:
getUsia() digunakan untuk menghitung usia pegawai yaitu tanggal sekarang - birthday

Kelas kedua merupakan kelas turunan dari kelas Pegawai dan diberi nama Manajer dengan deskripsi berikut.

Kelas
Manajer
Atribut
String departemen
Metode
Manager(String n, long s, Date d,String dept)
public String getDepartment()
setDepartment(String department)
getTunjangan()
getTotalGaji()

Ket:
getTunjangan() digunakan untuk menghitung tunjangan Manajer yaitu 20% dari *salary*

getTotalGaji() digunakan untuk menghitung total gaji manajer yaitu *salary* + tunjangan

1. Buat kode program untuk mengimplementasikan kedua kelas tersebut!
2. Buat kelas utama yang digunakan untuk mengisi data-data *manager* dan menampilkan Usia, Tunjangan, dan Total gaji.

Pengayaan

Untuk mendalami materi ini, kalian bisa mempelajari lebih dalam bahan-bahan bacaan berikut.

1. Modul
   - Avestro, J. (2007). Modul Pelatihan Java Education Network Indonesia (JENI). Yogyakarta: JEDI.
2. Buku-Buku:
   - Buyya, R., Somasundaram, T. S., & Chu, X. (2018). *Object Oriented Programming with Java: Essentials and Applications*. New Delhi, India: McGrawHill Education (India) Pvt Ltd.
   - Eck, D. J. (2021). *Introduction to Programming Using Java Version 8.1.3*. Geneva, NY: Hobart and William Smith Colleges.
   - Ladwa, H. (2021). *Object Oriented Programming with JAVA*.:.
   - Sekhar, G., & Reddy, E. (2020). *Lecture Notes on Object Oriented Programming Through Java*. Pune, India: INSTITUTE OF AERONAUTICAL ENGINEERING.
   - Wu, C. Thomas. *A Comprehensive Introduction to Objectoriented Programming With Java 1st Ed*. New York: McGrawHill Higher Education.

3. Artikel Daring

- BelajarCPP Team. Tutorial C++. https://www.belajarcpp.com/tutorial/cpp/. Akses terakhir 21 Oktober 2022.
- W3School. C++ Tutorial. https://www.w3schools.com/cpp/default.asp. Akses terakhir 21 Oktober 2022.
- W3School. Java Tutorial. https://www.w3schools.com/java/default.asp. Akses terakhir 21 Oktober 2022.
- GeeksforGeeks. Object Oriented Programming in C++. https://www.geeksforgeeks.org/object-oriented-programming-in- cpp/?ref=lbp. Akses terakhir 21 Oktober 2022.
- GeeksforGeeks. Java Programming Language. https://www.geeksforgeeks.org/java/?ref=ghm. Akses terakhir 21 Oktober 2022.
- JavaPoint Team. Java Tutorial. https://www.javatpoint.com/java-tutorial. Akses terakhir 21 Oktober 2022.
- Petani Kode. Tutorial Pemrograman Java untuk Pemula. https://www.petanikode.com/tutorial/java/. Akses terakhir 21 Oktober 2022.

Refleksi

Berilah tanda (√) pada kotak yang dianggap sesuai! Setelah mempelajari bab ini, bagaimanakah penguasaan kalian terhadap materi-materi berikut?

No.	Materi	Tidak Menguasai	Menguasai	Sangat Menguasai
1.	Kelas, Objek, dan Package			
2.	Access Modifier dan Enkapsulasi			
3.	Pewarisan			
4.	Polimorfisme			

1. Dari materi-materi tersebut, bagian manakah yang paling kalian sukai? Mengapa?
2. Apa manfaat yang kalian dapatkan setelah mempelajari materi bab ini untuk kehidupan sehari-hari?
3. Keterampilan apa saja yang dapat kalian kembangkan setelah mengikuti pembelajaran ini?

DAFTAR PUSTAKA

A.S., R. (2018). Logika Algoritma dan Pemrograman Dasar. Bandung: Modula.

Arief, U. M., Wibawanto, H., & Nastiti, A. L. (2019). Membuat Game Augmented Reality(AR) dengan Unity 3D. Yogyakarta: Andi.

Bachtiar, A. M. (2018). Pemrograman C dan C++. Bandung: Penerbit Informatika.

Buyya, R., Somasundaram, T. S., & Chu, X. (2018). Object Oriented Programming with Java: Essentials and Applications. New Delhi, India: McGraw-Hill Education (India) Pvt Ltd.

Eck, D. J. (2021). Introduction to Programming Using Java Version 8.1.3.Geneva, NY : Hobart and William Smith Colleges.

Fathansyah. (2015). Basis Data. Revisi Kedua. Bandung: Informatika Bandung.

Fitriyani, M., & Prahastuti, N. F. (2020). Personal Branding. Yogyakarta: Laksana.

Furman, B. J. (2010). Lecture Note on Algorithms, Pseudocode, and Flowcharts. San Jose: SAN JOSÉ STATE UNIVERSITY.

Galin, D. (2004). Software Quality Assurance From Theory to Implementation. London: Pearson Education Limited.

Gupta, S. B., & Mittal, A. (2017). Introduction To Database Managenent System, Second Edition. New Delhi: Laxmi Publications (P) Ltd.

Hery, A. (2021). BUKU KEWIRAUSAHAAN BUKU AJAR UNTUK MAHASISWA. Bandung: Yrama Widya.

Ismara, K. I., Pramono, H. S., Nugroho BU, Dwijonagoro, S., & Kuncoro, I. H. (2020). Strategi Penerapan Budaya Kerja Industri di Pendidikan Vokasi dengan Selamat dan Sehat. Yogyakarta: UNY Press.

Kadir, A. (2019). Logika Pemrograman menggunakan C++. Jakarta: Elex Media Komputindo.

Knuth, D. E. (2013). The Art of Computer Programmning. Volume 1. Fundamental Algorithms. Third Edition. Digital Release. Boston: Addison - Wesley.

Kotler, P., & Keller, K. L. (2016). Marketing Management. 15E. Uttar Pradesh, India: Pearson India Education Services Pvt. Ltd.

Ladwa, H. (2021). Object Oriented Programming with JAVA. -: -.

Maulana, G. (2017). Pembelajaran Dasar Algoritma dan Pemrograman Menggunakan El-Goritma Berbasis Web. Jurnal Teknik Mesin (JTM), Vol 06, 69-73.

Mayhew, D. J. (2008). Principles and Guidelines in Software User Interface Design. Pearson.

Mayhew, D. J. (2008). User Eficiency : Evaluation And Design. West Tisbury, MA, USA: Deborah J. Mayhew & Associate.

Melati, R. (2019). Pemrograman Dasar untuk siswa SMK/MAK Kelas X. Bandung: Sarana Pancakarsa Nusa.

Mohapatra, P. K. (2010). Software Engineering (A Lifecycle Approach). New Delhi: New Age International (P) Limited, Publishers.

Munir, R., & Lidya, L. (2016). Algoritma dan Pemrograman Dalam Bahasa Pascal, C, C++. Edisi Keenam. Bandung: Informatika Bandung.

Poo, D., Kiong, D., & Ashok, S. (2008). Object-Oriented Programming and Java. Second edition. London: Springer-Verlag London Limited.

Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition . New York: McGraw Hill Education.

Project Management Institute. (2013). A Guide To Project Management of Knowledge (PMBOK Guide)-Fifth Edition. Pennsylvania: Project Management Institute, Inc.

Ramadan, R., & Widyani, Y. (2013). Game Development Life Cycle Guidelines. 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 95-100). Sanur Bali, Indonesia: IEEE.

Sekhar, G., & Reddy, E. (2020). Lecture Notes on Object Oriented Programming Through Java. Pune, India: INSTITUTE OF AERONAUTICAL ENGINEERING.

Setyaningsih, E. (2018). Struktur Data. Yogyakarta: Akprind Press.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database System Concept, Seventh Edition. New York: McGraw-Hill.

Sommerville, I. (2016). Software Engineering. Tenth Edition. Harlow: Pearson Education Limited.

Suyanto. (2021). Buku Artificial Intelligence Edisi 3.Bandung: Penerbit Informatika.

Tamimy, M. F., & S., L. (2017). Sharing-mu, Personal Branding-mu. Jakarta: Visimedia.

Tanenbaum, A. S., & Bos, H. (2015). Modern Operating System. Fourth Edition. New Jersey: Pearson Education, Inc.

Wahyudi, S. E. (2020, 02 12). Teori Warna (Multimedia #4). Retrieved from informatika.uc.ac.id: https://informatika.uc.ac.id/id/2020/02/teori-warna-multimedia-4/#

Western Governors University. (2021, April 1). 5 Most Popular Operating Systems. Retrieved from wgu.edu: https://www.wgu.edu/blog/5-most-popular-operating-systems1910.html#close

White, A. W. (2011). The Elements of Graphic Design, Second Edition. Allworth.

Wirth, N. (2004). Algorithms and Data Structures. Oberon version. Updated 2012. New Jersey: Prentice Hall, Inc.

Young, R. R. (2004). The Requirements Engineering Handbook. Norwood: Artech House Inc.

DAFTAR LAMAN YANG DIAKSES

Administrator. (2011, Maret 27). Game Edukasi. Retrieved from EduChannel Indonesia: https://educhannel.id/blog/artikel/game-edukasi.html. Akses terakhir 12 Desember 2022.

Administrator. (2020, Januari 13). Pengenalan Teknologi Informasi. Retrieved from DISKOMINFO Kabupaten Kediri: https://diskominfo.kedirikab.go.id/baca/pengenalan-teknologi-informasi#:~:text=IT%20(Information%20AND%20Technology)%20%2C,mengomunikasikan%20dan%2Fatau%20menyebarkan%20informasi. Akses terakhir 20 Desember 2022.

Administrator. (2021, Agustus 28). Tantangan Big data dalam implementasi di perusahaan. Retrieved from BigBox Blog: https://bigbox.co.id/blog/permasalahan-tantangan-dan-solusi-big-data/. Akses terakhir 21 Desember 2022.

Administrator. (2022, Maret 18). Kelebihan dan Kekurangan Teknologi Cloud Computing. Retrieved from PT Cloud Hosting Indonesia: https://idcloudhost.com/cloud-computing-adalah/. Akses terakhir 20 Desember 2022.

Administrator. (2022, Desember 19). Keuntungan dan Kelemahan Cloud Computing Untuk Infrastruktur IT Perusahaan. Retrieved from Indonesian Cloud: https://indonesiancloud.com/keuntungan-dan-kelemahan-cloud-computing/. Akses terakhir 19 Desember 2022.

Administrator. (2022, Desember 5). Pentingnya Fraud Analytics bagi Bisnis, Jenis, dan Tekniknya. Retrieved from PT Verihubs Inteligensia Nusantara: https://verihubs.com/blog/fraud-analytics/ Akses terakhir 3 Januari 2023.

Administrator. (2022, Desember 28). Vision Vs Passion, Kenali Perbedaannya. Retrieved from Visecoach By Coaching Indonesia: https://visecoach.com/articles/read/vision-vs-passion-kenali-perbedaannya#:~:text=Visi%20dimaksudkan%20untuk%20menjadi%20panduan,atau%20kesenangan%20saat%20mengerjakan%20sesuatu. Akses terakhir 28 Desember 2022.

Alex Melnichuk. (6 Januari 2022). Top 8 Software Development Models. Retrieved from ncube: https://ncube.com/blog/top-software-development-models. Akses terakhir 21 Oktober 2022.

Anggakara, M. (2022, Februari 4). Technopreneur: Pengertian, Tujuan, Contoh, dan Perbedaan dengan Entrepreneur. Retrieved from PT Linov Roket Prestasi: https://www.linovhr.com/technopreneurship/ Akses terakhir 25 Desember 2022.

Annisya, F. (2021, September 23). Profesi/job profile bidang Perangkat Lunak dan GIM | Dasar-dasar Pengembangan Perangkat Lunak dan Gim. Retrieved from https://www.youtube.com/watch?v=xxpwZUFIeeU Akses terakhir 25 Desember 2022.

Aprilia, P. (2022, September 6). Mengenal User Interface: Pengertian, Kegunaan, dan Contohnya. Retrieved from Niagahoster.co.id: https://www.niagahoster.co.id/blog/user-interface/. Akses terakhir 20 Desember 2022.

Ariffudin, M. (2022, April 21). Apa itu DBMS? Pengertian, Fungsi, Kelebihan, Macam-macam DBMS. Retrieved from Niagahoster.co.id: https://www.niagahoster.co.id/blog/dbms-adalah/. Akses terakhir 20 Desember 2022.

Arnetta. (25 Oktober 2021). Telah Rilis 14 Versi, Ini Sejarah Windows dari Awal Sampai Sekarang. Retrieved from Dailysocial.id: https://dailysocial.id/post/sejarah-windows-dari-awal-sampai-sekarang. Akses terakhir 20 Desember 2022.

BelajarCPP Team. (12 Juni 2020).Tutorial C++. Retrieved from Belajar C++: https://www.belajarcpp.com/tutorial/cpp/. Akses terakhir 21 Oktober 2022

Devin Pickell. (15 Oktober 2019). The 7 Stages of Game Development. Retrieved from G2.com: https://www.g2.com/articles/stages-of-game-development. Akses terakhir 21 Oktober 2022.

Direktorat Bina K3 Kementerian Ketenagakerjaan RI. (16 Januari 2018). https://www.youtube.com/watch?v=7epjbQAc-eQ. Diakses pada 11 November 2022.

GeeksforGeeks. (11 Juli 2022). Object Oriented Programming in C++. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp. Akses terakhir 21 Oktober 2022.

GeeksforGeeks. (16 Januari 2023). Java Programming Language. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/java/?ref=ghm. Akses terakhir 17 Januari 2023.

Guntoro. (2022, Februari 2). 12 Framework untuk Pengembangan Web Terbaik. Retrieved from https://badoystudio.com/: https://badoystudio.com/framework-untuk-pengembangan-web/. Akses terakhir 20 Desember 2022.

Humas Kesmas. (15 Maret 2021). https://www.youtube.com/watch?v=6ZhBPy_5el0. Diakses pada tanggal 13 November 2022.

I/O Software Inc. (2022, November 11). What's The Best Game Asset Management Software? Retrieved from iomovo.io: https://iomovo.io/what-is-the-best-game-asset-management-software/. Akses terakhir 20 Desember 2022.

JavaPoint Team. (). Java Tutorial. Retrieved from Javapoint.com: https://www.javatpoint.com/java-tutorial. Akses terakhir 21 Oktober 2022

Java Point. (21 Oktober 2022). Requirement Engineering. Retrieved from Javapoint.com: https://www.javatpoint.com/software-engineering-requirement-engineering. Akses terakhir 21 Oktober 2022

Juego Studios. (13 April 2017). What Is the Game Development Life Cycle?. Retrieved from Gamedeveloper.com: https://www.gamedeveloper.com/business/what-is-the-game-development-life-cycle-. Akses terakhir 21 Oktober 2022.

Karsandi, R. (2022, Desember 26). Tips Berwirausaha Dengan Mengetahui Passion Anda. Retrieved from Grevia Networks: http://www.grevia.com/article/2/tips_berwirausaha_dengan_mengetahui_passion_anda#:~:text=Perasaan%20kepuasan%20hati%20itulah%20yang,sering%20disebut%20orang%20dari%20hobi. Akses terakhir 26 Desember 2022.

Kumparan. (2022, April 7). 7 Aplikasi Pembuat Game di PC, Tak Perlu Coding! Retrieved from Kumparan.com: https://kumparan.com/how-to-tekno/7-aplikasi-pembuat-game-di-pc-tak-perlu-coding-1xpSfKyB8JS/full Akses terakhir 12 Desember 2022.

Kumparan. (2022, Juni 13). Mengenal Apa itu iOS dan Fungsinya. Retrieved from Kumparan.com: https://kumparan.com/berita-update/mengenal-apa-itu-ios-dan-fungsinya-1yGPvIfdKHx/full. Akses terakhir 20 Desember 2022.

Martin, D. (2019, Desember 27). Software Requirements Analysis - The Key to Developing Great Software. Retrieved from medium.com: https://medium.com/illumination/software-requirements-analysis-the-key-to-developing-great-software-20162c5b0580. Akses terakhir 20 Desember 2022.

Moreno, L. (2020, Mei 12). Fundamentals of layout in user interface design (UI) : Composition, balance, and how to manage a good structure. Retrieved from uxdesign.cc: https://uxdesign.cc/fundamentals-of-layout-in-interface-design-ui-3a9dba31f1. Akses terakhir 20 Desember 2022.

Petani Kode. (). Tutorial Pemrograman Java untuk Pemula. Retrieved from Petanikode.com: https://www.petanikode.com/tutorial/java/. Akses terakhir 21 Oktober 2022

pp_pankaj. (2023, January 3). Computer Aided Software Engineering (CASE). Retrieved from www.geeksforgeeks.org: https://www.geeksforgeeks.org/computer-aided-software-engineering-case/. Akses terakhir 20 Desember 2022.

Redaksi. (2019, Juni 24). 8 Genre Game Online yang Perlu Anda Ketahui. Retrieved from okezone.com: https://techno.okezone.com/read/2019/06/24/326/2070092/8-genre-game-online-yang-perlu-anda-ketahui. Akses terakhir 12 Desember 2022.

Rehman, J. (2019, 09). Advantages and disadvantages of android operating system. Retrieved from itrelease.com: https://www.itrelease.com/2019/09/advantages-and-disadvantages-of-android-operating-system/. Akses terakhir 20 Desember 2022.

Rizkinaswara, L. (2020, Januari 28). Revolusi Industri 4.0. Retrieved from Dirjen Aptika Kominfo: https://aptika.kominfo.go.id/2020/01/revolusi-industri-4-0/. Akses terakhir 26 Nopember 2022.

Safety Sign Indonesia. (2021, Juni 14). Ergonomi Komputer: Bekerja di Depan Komputer Juga Ada Aturannya, Bagaimana Menurut Regulasi? Retrieved from safetysignindonesia.id: https://safetysignindonesia.id/ergonomi-komputer-bekerja-di-depan-komputer-juga-ada-aturannya-bagaimana-menurut-regulasi/. Akses terakhir 15 Nopember 2022.

Setiawan, R. (2021, November 24). Apa itu Tipografi dan Apa Kegunaannya? Retrieved from Dicoding: https://www.dicoding.com/blog/apa-itu-tipografi/ Akses terakhir 10 Januari 2023.

Subianto, W. (2021, Januari 26). Dampak Perkembangan Internet of Things terhadap Kehidupan Manusia. Retrieved from Kompasiana.com: https://www.kompasiana.com/weslysubianto8822/600ffb908ede48162f49f1d7/dampak-perkembangan-internet-of-things-terhadap-kehidupan-manusia?page=4&page_images=1 Akses terakhir 17 Desember 2022.

Technolgy Profession. (11 September 2015). https://www.youtube.com/watch?v=pquPUX1EihM. Diakses pada 6 Januari 2023.

Tim Tutorialspoint.com. (2022, Oktober 22). Software Case Tools Overview. Retrieved from tutorialspoint.com : https://www.tutorialspoint.com/software_engineering/case_tools_overview.htm# Akses terakhir 20 Desember 2022.

Volle, A. (2023, January 16). iOS Operating System. Retrieved from Britanica.com: https://www.britannica.com/topic/iOS Akses terakhir 20 Desember 2022.

Wahyudi, S. E. (12 Februari 2020). Teori Warna. Retrieved from Informatika Universitas Ciputra: https://informatika.uc.ac.id/id/2020/02/teori-warna-multimedia-4/ Akses terakhir 25 Desember 2022.

Widyananda, R. F. (2021, Maret 12). 7 Teknologi Baru yang Jadi Tren di 2020, Semakin Canggih. Retrieved from Merdeka.com: https://www.merdeka.com/jatim/7-teknologi-baru-yang-jadi-tren-di-2020-semakin-canggih-kln.html Akses terakhir 15 Desember 2022.

SUMBER GAMBAR

Gambar 3.2 diunduh dari https://medium.com/illumination/software-requirements-analysis-the-key-to-developing-great-software-20162c5b0580. Akses terakhir 20 Desember 2022.

Gambar 3.3 dikutip dari Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition . New York: McGraw Hill Education.

Gambar 3.4 dikutip dari Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition . New York: McGraw Hill Education.

Gambar 3.5 dikutip dari Sommerville, I. (2016). Software Engineering. Tenth Edition. Harlow: Pearson Education Limited.

Gambar 3.6 dikutip dari Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition . New York: McGraw Hill Education.

Gambar 3.7 dikutip dari Sommerville, I. (2016). Software Engineering. Tenth Edition. Harlow: Pearson Education Limited.

Gambar 3.8 dikutip dari Sommerville, I. (2016). Software Engineering. Tenth Edition. Harlow: Pearson Education Limited.

Gambar 3.9 dikutip dari Presman, R. S., & Maxim, B. R. (2020). Software Engineering A Practitioner's Approach Ninth Edition . New York: McGraw Hill Education.

Gambar 3.10 dikutip dari Ramadan, R., & Widyani, Y. (2013). Game Development Life Cycle Guidelines. 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 95-100). Sanur Bali, Indonesia: IEEE.

Gambar 4.3 dikutip dari Gupta, S. B., & Mittal, A. (2017). Introduction To Database Managenent System, Second Edition. New Delhi: Laxmi Publications (P) Ltd.

Gambar 4.4 dikutip dari Gupta, S. B., & Mittal, A. (2017). Introduction To Database Managenent System, Second Edition. New Delhi: Laxmi Publications (P) Ltd.

Gambar 4.5 diunduh dari : https://www.niagahoster.co.id/blog/dbms-adalah/. Akses terakhir 20 Desember 2022.

Gambar 4.6 diunduh dari : https://www.niagahoster.co.id/blog/dbms-adalah/. Akses terakhir 20 Desember 2022.

Gambar 4.7 diunduh dari : https://www.niagahoster.co.id/blog/dbms-adalah/. Akses terakhir 20 Desember 2022.

Gambar 4.8 diunduh dari https://www.tutorialspoint.com/software_engineering/case_tools_overview.htm#. Akses terakhir 20 Desember 2022.
Gambar 4.9 dikutip dari Tanenbaum, A. S., & Bos, H. (2015). Modern Operating System. Fourth Edition. New Jersey: Pearson Education, Inc.
Gambar 4.10 diunduh dari https://dailysocial.id/post/sejarah-windows-dari-awal-sampai-sekarang. Akses terakhir 20 Desember 2022.
Gambar 4.11 diunduh dari https://www.jurnalponsel.com/pengertian-unix-beserta-sejarah-ciri-ciri-dan-contoh-unix/. Akses terakhir 20 Desember 2022.
Gambar 4.12 dunduh dari https://www.freepnglogos.com/images/linux-22617.html. Akses terakhir 25 Desember 2022.
Gambar 4.13 diunduh dari https://www.pngegg.com/id/png-dqudb. Akses terakhir 25 Desember 2022.
Gambar 4.14 diunduh dari https://www.pngwing.com/en/free-png-iwmmv. Akses terakhir 25 Desember 2022.
Gambar 4.15 diunduh dari https://1000logos.net/ios-logo/. Akses terakhir 25 Desember 2022.
Gambar 4.16 diunduh dari https://www.freepnglogos.com/images/android-logo-12379.html. Akses terakhir 25 Desember 2022.
Gambar 4.17 diunduh dari https://informatika.uc.ac.id/id/2020/02/teori-warna-multimedia-4/ Akses terakhir 25 Desember 2022.
Gambar 4.19 dikutip dari Furman, B. J. (2010). Lecture Note on Algorithms, Pseudocode, and Flowcharts. San Jose: SAN JOSÉ STATE UNIVERSITY

GLOSARIUM

5R	Budaya kerja yang banyak diterapkan pada dunia industri terdiri dari Ringkas, Rapi, Resik, Rawat dan Rajin yang diadopsi dari budaya kerja 5S di Jepang yaitu *Seiri*, *Seiton*, *Seiso*, *Seiketsu*, dan *Shitsuke*.
Abstrak	Salah satu bentuk kelas yang digunakan untuk mengekspresikan atribut dan metode yang bersifat umum yang dapat digunakan oleh kelas-kelas turunannya dan tidak dapat diinstansiasi.
Additive	Jenis warna yang didasarkan pada penglihatan manusia, berasal dari spektrum cahaya.
Agile Development	Metodologi pengembangan perangkat lunak dan gim jangka pendek yang didasarkan pada proses pengerjaan suatu tahapan secara berulang, terdiri dari aturan dan solusi yang sudah disepakati, dan dilakukan secara kolaborasi antartim.
Algoritma	Urutan logis pengembilan keputusan pada pemecahan masalah dengan bantuan komputer.
Artificial Intelligence	Teknologi yang memungkinkan mesin (benda mati) mempunyai kecerdasan layaknya manusia dan bisa diatur sesuai keinginan manusia.
Basis Data	Sekumpulan data dan informasi yang tersimpan bersama dengan redudansi terkontrol yang digunakan untuk melayani satu atau lebih aplikasi secara optimal.
Big Data	Teknologi pengolahan data yang memiliki volume atau ukuran yang sangat besar yang terdiri dari data yang terstruktur (*structured*), semi-terstruktur (*semi structured*), dan tidak terstruktur (*unstructured*).
Blackbox Testing	Teknik pengujian perangkat lunak yang didasarkan pada fungsionalitas program.
Brainstorming	Metode yang digunakan untuk pemecahan masalah dengan mengumpulkan ide-ide dari anggota tim.

Buggy	Masih ditemukan kesalahan (*error*) pada perangkat lunak dan gim yang sedang dibangun.
CASE	Perangkat lunak yang digunakan untuk membantu setiap fase dalam pengembangan perangkat lunak dan gim.
Cloud Computing	Proses pengolahan sumber daya komputasi yang memanfaatkan kombinasi teknologi komputer dan penggunaan berbasis internet.
Enkapsulasi	Konsep dalam pemrograman berorientasi objek yang menekankan pada pembungkusan data dan fungsi.
Flowchart	Suatu bagan yang menggambarkan arus logika dari mulai masuk, diproses, sampai dengan menjadi keluaran.
Gameplay	Pola, aturan, atau mekanisme yang mengatur bagaimana proses interaksi pemain dengan gim yang diciptakan.
Gim	Berasal dari kata Game, permainan elektronik yang melibatkan interaksi antarmuka pengguna atau perangkat masukan.
HAKI	Hak yang diperoleh dari hasil olah pikir berupa produk atau proses yang bisa digunakan dan dimanfaatkan oleh manusia yang kemudian menimbulkan hak atas hasil olah pikir tersebut
IEEE	Sebuah organisasi yang mengurusi masalah pengembangan teknologi yang berhubungan dengan keteknikan elektro dan elektronika yang terdiri dari berbagai ahli di bidang teknik yang menawarkan berbagai pengembangan standar-standar dan bertindak sebagai pihak yang mempercepat teknologi-teknologi baru dalam semua aspek dalam industri dan rekayasa (*engineering*), yang mencakup telekomunikasi, komputer, kelistrikan, antariksa, dan elektronika.
Information Security	Perlindungan kepada data dan informasi dari akses, penggunaan, gangguan, modifikasi, dan perusakan yang tidak diizinkan.

Inheritance	Kemampuan sebuah kelas pada pemrograman berorientasi objek untuk menurunkan atau mewariskan sifat-sifat yang dimilikinya kepada kelas yang lain.
Interface	Mekanisme pada pemrograman berorientasi objek yang memungkinkan untuk berbagi konstanta atau metode yang dapat digunakan oleh sejumlah kelas yang lainnya.
IoT	Sebuah konsep teknologi menghubungkan perangkat lain dengan media internet dan dapat dikendalikan dari jarak jauh.
Kelas	Definisi statis dari sekumpulan objek yang memiliki karakteristik yang sama, terdiri dari atribut (data) dan fungsi (metode).
Konstruktor	Metode pada kelas yang dipanggil saat suatu objek diciptakan (diinstansiasi).
Manajemen Aset	Pengelolaan asset-aset dalam pengembangan perangkat lunak dan gim mencakup penyiapan, penciptaan, penelusuran, penggunaan kembali, dan pembaruan.
Modifier	Salah satu fitur pada pemrograman berorientasi objek yang digunakan untuk melakukan penyembunyian data dan fungsi (atribut dan metode).
Multitasking	Kemampuan sistem operasi untuk melakukan pekerjaan (proses) yang berbeda menggunakan CPU yang sama secara bersamaan.
Multiuser	Kemampuan sitem operasi untuk digunakan oleh banyak pengguna secara bersamaan.
Objek	Satuan terkecil dalam pemrograman objek yang merupakan instansiasi dari kelas.
Open Source	Suatu kode program yang dipublikasikan secara umum sehingga bisa digunakan, diperbaiki, dimodifikasi, atau disebarkan untuk keperluan pribadi atau hobi, perusahaan atau untuk komersil, tanpa harus membayar biaya sedikit pun.

Overload	Kemampuan pada pemrograman berorientasi objek untuk memiliki nama metode yang sama tetapi memiliki kemampuan yang berbeda.
Override	Kemampuan pada pemrograman berorientasi objek untuk menulis ulang deskripsi dari sebuah metode.
Paradigma Pemrograman	Kerangka berpikir atau cara pandang seperti apa yang digunakan pada saat membuat atau menulis program.
Passion	Semangat yang mendorong seseorang untuk melakukan sesuatu untuk mencapai suatu visinya.
Perangkat Lunak	Serangkaian baris perintah (program komputer) yang ketika dieksekusi akan menjalankan unjuk performa seperti yang diharapkan, didalamnya terdapat struktur data yang dimungkinkan untuk dimanipulasi untuk menghasilkan informasi, serta informasi deskripstif yang menjelaskan pengoperasian dan penggunaan program
Personal Branding	Citra diri seseorang, yang menggambarkan identitas pribadi dan menjadi ciri khasnya.
Polimorfisme	Kemampuan pemrograman berorientasi objek yang memungkinkan suatu kelas memiliki beberapa bentuk yang berbeda.
Pseudocode	Suatu cara penulisan logika program menyerupai Bahasa pemrograman tertentu tanpa memperdulikan *syntax* dari bahasa pemrograman tersebut.
Rekayasa Perangkat Lunak	Bagaimana menerapkan pendekatan yang sistematis, terdisiplin, terukur untuk pengembangan, pengoperasian, dan pemeliharaan perangkat lunak.
Requirement	Atribut yang diperlukan dalam suatu sistem, pernyataan yang mengidentifikasi kemampuan, karakteristik, atau faktor kualitas suatu sistem agar memiliki nilai dan kegunaan bagi pelanggan atau pengguna.
Revolusi Industri	Perubahan besar-besaran mengenai cara manusia dalam mengolah sumber daya untuk memproduksi barang dan jasa dalam berbagai sektor sehingga berdampak pada kehidupan ekonomi, politik, bahkan sosial-budaya.

Sandbox	lapisan perlindungan tambahan yang bertujuan untuk mencegah kode dan perangkat lunak berbahaya yang menyerang dan membahayakan sistem.
Server	Sistem komputer yang didesain untuk memproses permintaan (*request*) dan mengirim data ke komputer lain melalui internet atau jaringan lokal.
Software Architect	Orang yang ahli dalam menciptakan desain *high-level*, mengembangkan suatu perangkat lunak.
SQL	Bahasa standar yang digunakan untuk mengelola basis data relasional.
Substractive	Jenis warna yang didasarkan pada pigmen bahan
Technopreuneur-ship	Wirausahawan yang memanfaatkan perkembangan teknologi untuk menciptakan, mengembangkan, dan memasarkan suatu produk atau jasa.
Tipografi	Ilmu dalam memilih dan menata huruf dengan pengaturan penyebarannya pada ruang-ruang yang tersedia, untuk menciptakan kesan tertentu, sehingga dapat menolong pengguna untuk mendapatkan kenyamanan membaca semaksimal mungkin.
User Experience	Salah satu aspek yang harus diperhatikan dalam merancang antarmuka pengguna yang didasarkan pada pengalaman pengguna saat menggunakan suatu perangkat lunak.
User Interface	Tampilan visual yang pertama kali bersentuhan langsung dengan pengguna.
Vision	Rangkaian kalimat yang menyatakan cita-cita atau impian yang ingin dicapai seseorang atau organisasi.
Whitebox Testing	Teknik pengujian perangkat lunak yang didasarkan pada logika program.
Workstation	Perangkat komputer berspesifikasi tinggi yang dimanfaatkan untuk keperluan pekerjaan berat, seperti perhitungan ilmiah atau bidang Teknik.

INDEX

R

S

T

U

W

PROFIL PENULIS

Nama Lengkap : Marwondo, S.T., M.Kom.
Surel : m1214w@gmail.com
Instansi : Universitas Informatika dan Bisnis Indonesia (UNIBI)
Alamat Instansi : Jl. Soekarno-Hatta No 643 Bandung, Jawa Barat
Bidang Keahlian : Rekayasa Perangkat Lunak, Cybersecurity, Kecerdasan Buatan, Multimedia

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir):

1. Dosen Tetap Program Studi Informatika, Universitas Informatika dan Bisnis Indonesia (UNIBI) Bandung, 2003-sekarang

Riwayat Pendidikan dan Tahun Belajar:

1. S2 Sistem Informasi, Bidang Rekayasa Sistem. STMIK LIKMI, Bandung-Jawa Barat. Lulus tahun 2017.
2. S1 Teknik Informatika, Bidang RPL Terapan. STMIK PMBI (Dharma Negara), Bandung-Jawa Barat. Lulus tahun 2003.
3. SMU Al-Azhar Boarding School, Bekasi-Jawa Barat. Lulus tahun 1998.
4. SMP Negeri 1 Lamongan, Jawa Timur. Lulus tahun 1995.
5. SDN Sidomulyo II, Lamongan-Jawa Timur. Lulus tahun 1992.
6. MI Al-Falahiyah, Lamongan-Jawa Timur. Lulus tahun 1992.

Judul Buku dan Tahun Terbit (10 Tahun Terakhir):

1. Manajemen Retail Di Era Pemasaran Online. 2021. Bandung: UNIBI Press. Penulis ke-3

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir):

1. 2022 - Augmented Reality untuk Pembelajaran Alat Musik Tradisional Calung Renteng
2. 2022 - Penerapan Data Mining Menggunakan Regresi Linier Sederhana Untuk Menganalisis Pengaruh Nilai Tukar Terhadap Volume Ekspor Integrated Circuit Korea Selatan
3. 2022 - Sistem Informasi Kunjungan Pasien Pada Klinik Mitra Medika
4. 2021 - Penerapan Augmented Reality Pada Multimedia Pembelajaran Tata Surya Untuk Tingkat Sekolah Menengah Pertama
5. 2021 - Sistem Informasi Akuntansi Pengelolaan Pinjaman Studi Kasus Pada Koperasi Gapoktan Mekar Kagugat Desa Bojongloa
6. 2020 - Pengembangan Perangkat Lunak Pengelolaan Layanan Perbaikan Komputer
7. 2019 - Pengembangan Media Pembelajaran Limit Fungsi Berbasis Multimedia Untuk Sekolah Menengah Atas Kelas XI
8. 2018 - Sistem Informasi Pada Pengelolaan Pengarsipan Studi Kasus Pada Kantor Pusat Bala Keselamatan Bandung
9. 2018 - Sistem Informasi Pengadaan Material Studi Kasus Pada PT. Duta Bangun Kreasindo

Informasi Lain:

1. Google Scholar ID: gJ9HcDYAAAAJ https://scholar.google.com/citations?hl=id&user=gJ9HcDYAAAAJ
2. HKI : Modul Pembelajaran Pemrograman Berorientasi Objek dengan Java, Tahun 2022, Nomor P/ID 000340368

PROFIL PENULIS

Nama Lengkap : Rini Melati, S.Kom
Surel : ummiaghif@gmail.com
Instansi : SMK Negeri 11 Bandung
Alamat Instansi : Jl. Budi Cilember Kota Bandung
Bidang Keahlian : Rekayasa Perangkat Lunak

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir):

1. Guru SMK Negeri 11 Bandung 2008 sampai sekarang.

Riwayat Pendidikan dan Tahun Belajar:

1. SD lulus tahun 1993 sdn Pasirkaliki Iv Bandung.
2. SMP lulus tahun 1996 SMPS Mutiara 4 Bandung.
3. SMA/SMK lulus tahun 1999 SMKN 1 Bandung .
4. Sarjana lulus tahun 2005 STMIK “Amik Bandung” Bandung Jurusan Sistem Informasi.

Judul Buku dan Tahun Terbit (10 Tahun Terakhir):

1. Pemrograman Dasar untuk Smk Kelas X, Penerbit Sarana Pancakarsa Nusa, Tahun 2019

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir):

Tidak ada

PROFIL PENELAAH

Nama Lengkap : Irya Wisnubhadra
Surel : irya.wisnubhadra@uajy.ac.id
Instansi : Universitas Atma Jaya Yogyakarta
Alamat Instansi : Jl. Babarsari 44, Yogyakarta

Bidang Keahlian : Pemrograman, Database System, Business Intelligence
Data Penelitian dan Karya detail dapat dilihat di Google Scholar

Riwayat Pekerjaan/Profesi (3 Tahun Terakhir):

1. Dosen Pengajar Tetap, Teknik Informatika Universitas Atma Jaya Yogyakarta, (1994-sekarang).
2. Pengajar di Lembaga Pelatihan Teknologi Informasi, Pilar Teknotama, (2019 – sekarang).
3. Scientific Committee Gerakan PANDAI, Bebras Indonesia, supported by Google.org (2019 - sekarang).

Riwayat Pendidikan dan Tahun Belajar:

1. S1: Department Teknik Elektro dan Teknologi Informasi, Universitas Gadjah Mada (1988-1994).
2. S2: Teknik Informatika, Rekayasa Perangkat Lunak, Institut Teknologi Bandung (1998-2001).
3. S3: Faculty of Information and Communication Technology, Universiti Teknikal Malaysia, Melaka (2018-sekarang).

Judul Buku/Karya dan Tahun Terbit (3 Tahun Terakhir):

1. Copyright Perangkat Lunak, Aplikasi Business Intelligence Transportasi Sawit, 2022.
2. Copyright Perangkat Lunak, Aplikasi monitoring transportasi buah sawit, logtransawit. online, 2019.
3. Buku Panduan Guru Informatika, Kelas VII, Penerbit: Puskurbuk.
4. Buku Siswa Informatika, Kelas VII, Penerbit: Puskurbuk.
5. Buku Panduan Guru Informatika, Kelas VIII, Penerbit: Puskurbuk.
6. Buku Siswa Informatika, Kelas VIII, Penerbit: Puskurbuk.
7. Buku Panduan Guru Informatika, Kelas X, Penerbit: Puskurbuk.
8. Buku Siswa Informatika, Kelas X, Penerbit: Puskurbuk.
9. Buku Panduan Guru Informatika, Kelas XI, Penerbit: Puskurbuk.
10. Buku Siswa Informatika, Kelas XI, Penerbit: Puskurbuk.
11. Mobility Data Warehouse for Transportation of Oil Palm Fresh Fruit Bunches, ICIC Express Letters, Part B: Applications, 2022, 13(1), pp. 11–19.
12. Qb4MobOLAP: A vocabulary extension for Mobility OLAP on the Semantic Web, Algorithm, MDPI, 2021.
13. Open Spatiotemporal Data Warehouse for Agriculture Production Analytics, International Journal of Intelligent Engineering and Systems 13 (6), 419-431, 2020.
14. Classification of pertussis vulnerable area with location analytics using multiple attribute decision making, Int. J. Innov. Comput. Inf. Control 16 (6), 1943-1957, 2020.
15. Sistem Informasi Berbasis Web Sebagai Sarana Penyebaran Informasi dan Pengelolaan Pemerintahan Desa Barepan, Proceeding of The URECOL, 2020.

Judul Penelitian dan Tahun Terbit (3 Tahun Terakhir):

1. Pengembangan Mobility Business Intelligence untuk peningkatan produktivitas sistem transportasi TBS kelapa sawit secara berkelanjutan, Penelitian Terapan, Tahun 2020 – 2022, DIKTI.
2. Sistem Informasi Desa untuk Efektivitas dan Efisiensi Pelayanan Masyarakat Desa Barepan, Program Kemitraan Masyarakat, Tahun 2019 – 2020, DIKTI.
3. Pemodelan dan Pengembangan Query Mobility Business Intelligence pada Semantic Web, Tahun 2019 – 2020, DIKTI.
4. Rancang Bangun Kendali Tinggi Muka Air Lahan Gambut Otomatis dan Real Time Untuk Menjamin Produktivitas Kelapa Sawit, Tahun 2019 – 2019, DIKTI.

PROFIL PENELAAH

Nama Lengkap : Dr. Asep Wahyudin, S.Kom., M.T.
Surel : away@upi.edu
Instansi : Universitas Pendidikan Indonesia
Alamat Instansi : Jl. Dr. Setiabudi No.229, Isola, Kec. Sukasari, Kota Bandung, Jawa Barat 40154
Bidang Keahlian : Information System; Software Engineering; IS/IT Strategic Plan; Information Management; IS/IT Audit; IS/IT Governance; Business Process Management

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir):

1. Kepala KBK Software Engineering and Information Management Laboratory,
2. Program Studi Ilmu Komputer UPI, 2021-Sekarang
3. Dosen Tetap Program S1 Studi Ilmu Komputer, 2006-sekarang
4. Dosen Tetap Program Studi S2 Pendidikan Ilmu Komputer, 2006-sekarang
5. Tim Pengembang Kurikulum Universitas Pendidikan Indonesia, 2021-sekarang
6. Asesor Lembaga Akreditasi Mandiri (LAM) INFOKOM, 2022-sekarang
7. Kepala Divisi Pengembangan Sistem Informasi pada Direktorat TIK UPI, 2007-2019

Riwayat Pendidikan dan Tahun Belajar:

1. Universitas Indonesia, Program Doktoral S3, tahun lulus 2019
2. Institut Teknologi Bandung, Program Magister S2, tahun lulus 2005
3. STMIK Bandung, Program Sarjana S1, tahun lulus 2000

Judul Buku dan Tahun Terbit (10 Tahun Terakhir):

1. Otomasi Perkantoran, Program Studi Manajemen Perkantoran S2 UPI

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir):

1. The Development and Effectiveness of Business Strategy Model with A Partnership Approach for Growing Entrepreneurship, The 5th Global Conference on Business, Management and Entrepreneurship 2021
2. Virtual collaboration strategic planning process using balanced scorecard and critical success factors, Annual Applied Science and Engineering Conference (AASEC),
3. Strategic Alignment Maturity Level Model Using Drivers of Change in a Business Environment, The 6th International Conference on Science in Information Technology (ICSITech), 2020
4. The use of SMART and WebGIS visualization methods in recommending regions that require clean water supply, Journal of Engineering Science and Technology, 2020
5. Collaborative Information System Monitoring and Evaluation Tools Model, The 6th International Conference on Science in Information Technology (ICSITech), 2020
6. A preliminary phase on anatomizing multiple sensitive attribute by determining main sensitive attribute, Annual Applied Science and Engineering Conference (AASEC), 2020
7. Business-information systems strategic alignment readiness maturity level: Corporate and business-technology driver perspective, Journal of Engineering Science and Technology, 2019
8. A Collaborative Process Scheme in Strategic Information Systems Planning, Third International Conference on Informatics and Computing (ICIC), 2018
9. The use of scale invariant feature transform (SIFT) algorithms to identification garbage images based on product label, 3rd international conference on science in information technology (ICSITech), 2017
10. Research Classification in SISP Development A Critical Review, International Conference on Science in Information Technology, 2015

PROFIL EDITOR

Nama Lengkap : Annis Diniati Raksanagara

Email : annisdr@indo.net.id

Instansi : Editor lepas

Alamat Instansi : Bogor

Bidang Keahlian : Editor Buku

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir):

1. Editor buku Matematika Militer Dasar, Unhan, 2022.
2. Editor buku siswa Matematika Kelas III SD, Pusbuk, 2021.
3. Editor buku guru Matematika Kelas III SD, Pusbuk, 2021.

Riwayat Pendidikan dan Tahun Belajar:

1. S2, lulus dari Jurusan Matematika ITB 1999.
2. S1, lulus dari Jurusan Matematika ITB 1992.

Sertifikat Keahlian

1. Editor Buku, sertifikasi LSP PEP a.n. BNSP 2021

Judul Buku dan Tahun Terbit (10 Tahun Terakhir):

Tidak ada buku non-fiksi

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir):

Tidak ada.

PROFIL ILUSTRATOR

Nama Lengkap : Dana Rizki Nur Adnan
Email : dananuradnan@gmail
Instagram : @danaaddnan
Instansi : Giattt Studio
Alamat Instansi : Puri Randusari E13, RT 03/RW 09 Prambanan Klaten 57454
Bidang Keahlian : Visual Art, Animator, Illustrator

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir)

1. Owner Studio Giattt (2018 -sekarang)
2. Illustrator Freelance (2012 - 2018)
3. Tentor Matematika dan Fisika (2010 - 2012) Galileo, Gongsin

Riwayat Pendidikan dan Tahun Belajar

1. Pendidikan Teknologi dan Kejuruan PascaSarjana UNY (2012 - 2018)
2. Pendidikan Teknik Pemesinan UNY (2006 - 2011)
3. SMA Negeri 1 Klaten (2003 - 2006)

Judul Buku dan Tahun Terbit (10 Tahun Terakhir)

1. The Strategy Journey by Julie Choo (sebagai Illustrator) 2020
2. Jago Taekwondo by Agus Herdadi SP.MMA (sebagai Illustrator) 2020
3. Ape Mind, Old Mind, New Mind by John W (sebagai Illustrator) 2018
4. American Sign Language by Vicky Allen (sebagai Illustrator) 2017

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir)

1. Pengembangan Model Pembelajaran Personal Leadership pada Siswa Sekolah Menengah Kejuruan. 2019

Kemampuan dan Pelatihan

1. Software yang dikuasai: Corel Draw, Photoshop, Adobe Illustrator, Autodesk Inventor, Office
2. Bahasa yang dikuasai: Indonesia, Jawa, English
3. Pelatihan yang diikuti: Training of Mentor Inkubator Bisnis Umby by LUNAS (2021); Workshop Seniman Pasca Terampil by PSBK Jogja (2020)

PROFIL DESAINER

Nama Lengkap : Eko Fitriono
Telp Kantor/HP : - / 085155314772
Surel : ekofitriono365@gmail.com
Bidang Keahlian : Desainer Grafis
Instansi : Freelancer

Riwayat Pekerjaan/Profesi (10 Tahun Terakhir):

1. Desainer Freelance sejak 2015 hingga sekarang.
2. Tim Desainer Buku SMA 2019, Pusat Pengembangan dan Pembinaan Bahasa Kemendikbudristek, 2019.
3. Tim Desainer Buku SMK 2019, Pusat Pengembangan dan Pembinaan Bahasa Kemendikbudristek, 2019.
4. Tim Desainer Buku Terjemahan SD 2020, Pusat Pengembangan dan Pembinaan Bahasa Kemendikbudristek, 2020.
5. Tim Desainer Buku SMK 2022, Pusat Pengembangan dan Pembinaan Bahasa Kemendikbudristek, 2022.
6. Desainer di Letterhend Studio sejak 2020 hingga sekarang.

Riwayat Pendidikan dan Tahun Belajar:

1. SDN 2 Serpong, Tangerang Selatan.
2. SMPN 1 Serpong, Tangerang Selatan.
3. SMAN 1 Cisauk, Tangerang Selatan.
4. S-1 di Universitas Pendidikan Indonesia tahun 2007-2014.

Judul Buku dan Tahun Terbit (10 Tahun Terakhir):

Tidak ada

Judul Penelitian dan Tahun Terbit (10 Tahun Terakhir):

Tidak ada