

Bab 2

Algoritma dan Pemrograman Lanjut



Bagaimana sebuah aplikasi di ponsel dapat mengetahui apa yang kamu inginkan hanya dengan satu sentuhan? Bagaimana kamu menyusun langkah-langkah agar komputer dapat “mengerti” dan melakukan perintahmu dengan tepat?



Tujuan Pembelajaran

Setelah mempelajari dan akan mencermati uraian materi beserta aktivitas pembelajaran pada bab ini, kamu diharapkan mampu:

1. menerapkan pemrograman berbasis teks,
2. menerapkan *debugging*,
3. melakukan perbandingan algoritma.

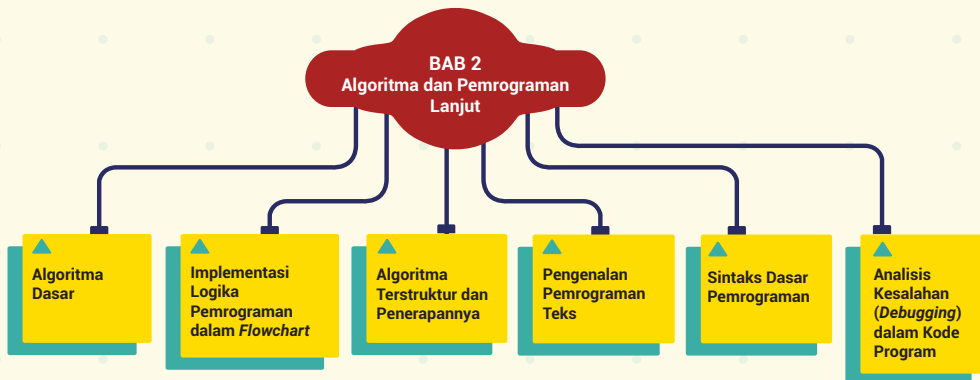


Kata Kunci

- Algoritma
- Pemrograman
- *Flowchart*
- *Debugging*



Peta Materi





Apersepsi

Setiap hari tanpa kamu sadari, kamu menjalankan aktivitas-aktivitas algoritma, seperti menentukan rute tercepat ke sekolah, menyalakan televisi, mengatur jadwal harian, dan mengerjakan tugas. Semua aktivitas tersebut melibatkan langkah-langkah berurutan agar tujuan tercapai dengan benar. Sistem kerja komputer bekerja dengan cara yang sama, mengikuti instruksi langkah demi langkah yang disebut algoritma dan instruksi tersebut ditulis dalam bentuk bahasa pemrograman tertentu. Dengan memahami algoritma dan pemrograman, kamu dapat menciptakan solusi cerdas, mulai dari aplikasi sederhana hingga teknologi yang kompleks. Jika kamu dapat memberi perintah ke komputer seperti kepada robot asisten pribadi, perintah apa yang ingin kamu berikan?

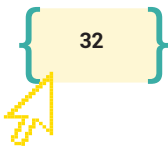




Sebelum memasuki pembelajaran, jawablah pertanyaan-pertanyaan berikut berdasarkan pengetahuan dan pengalamanmu sehari-hari.

A. Pilihlah salah satu jawaban yang paling sesuai dengan pendapat pribadimu.

1. Menurut pengetahuanmu, algoritma adalah ...
 - a. bahasa pemrograman untuk membuat aplikasi
 - b. kumpulan data yang tersimpan dalam komputer
 - c. rangkaian langkah-langkah logis dan sistematis untuk menyelesaikan masalah
 - d. perangkat keras yang menjalankan program
 - e. aplikasi pendukung kecerdasan artifisial
2. Di bawah ini yang tidak termasuk karakteristik algoritma adalah ...
 - a. *output* yang relevan
 - b. langkah yang jelas dan tidak ambigu
 - c. langkah yang tidak terbatas
 - d. dapat dieksekusi secara efektif
 - e. memahami perintah terstruktur
3. Perbedaan utama antara pemrograman berbasis blok dan tekstual adalah ...
 - a. Pemrograman blok menggunakan bahasa alami, sementara tekstual menggunakan simbol.
 - b. Pemrograman blok hanya digunakan untuk gim, sedangkan tekstual untuk aplikasi.
 - c. Pemrograman blok menggunakan komponen visual, sedangkan tekstual menggunakan penulisan kode.
 - d. Pemrograman blok dan tekstual sama menggunakan komponen visual dan tekstual kode.
 - e. Pemrograman blok dan visual tidak ada perbedaan.



B. Jawablah pertanyaan berikut dengan pengetahuan yang kamu miliki.

1. Mengapa kamu perlu menuliskan langkah-langkah algoritma sebelum membuat program komputer? Sebutkan satu contoh algoritma dalam kehidupan sehari-hari yang sering kamu lakukan!
2. Scratch dan Blockly adalah contoh pemrograman berbasis, sedangkan Python dan C menggunakan pemrograman berbasis.....

A. Algoritma Dasar

Algoritma merupakan langkah-langkah logis yang disusun secara sistematis untuk menyelesaikan masalah atau mencapai tujuan tertentu. Konsep algoritma ini telah kamu pelajari pada fase sebelumnya, seperti cara memahami dan menuliskan urutan instruksi sederhana dan menuliskan urutan langkah ke dalam bentuk *pseudocode*. Pada bab ini, kamu akan kembali meninjau konsep dasar tersebut sebagai fondasi, sebelum melangkah ke penerapan yang lebih kompleks dalam bentuk pemrograman berbasis teks, membandingkan algoritma, dan melakukan *debugging*. Mengingat kembali bahwa pemahaman akan konsep algoritma dasar ini, akan membantu membangun logika berpikir komputasional yang lebih kuat saat menulis kode program dan melakukan *debugging*.

Algoritma tidak terbatas pada bidang ilmu komputer saja, tetapi prinsip-prinsip algoritma dapat diterapkan pada berbagai disiplin ilmu, termasuk matematika, teknik, bahkan proses pengambilan keputusan dalam kehidupan sehari-hari. Dalam konteks ilmu komputer, algoritma digunakan untuk memproses data, membuat keputusan, dan menyusun program agar dapat menjalankan tugas secara efisien. Namun, konsep algoritma tidak hanya terbatas pada komputer, melainkan juga dapat ditemukan dalam berbagai aktivitas kehidupan sehari-hari, seperti resep masakan, prosedur perakitan,

atau instruksi pemecahan masalah. Desain algoritma perlu disusun dengan terstruktur dan sistematis dari permasalahan yang jelas sampai dengan solusi yang tepat, sehingga algoritma mudah dipahami.

1. Bentuk Algoritma Dasar

Untuk memudahkan pemahaman dan pengembangan program, bentuk algoritma dapat disajikan dalam berbagai format, seperti uraian naratif, *pseudocode*, tabel Input-Proses-Output (IPO), maupun *flowchart*. Penyajian algoritma dalam bentuk tersebut dapat mempermudah untuk memahami dan menerapkan solusi secara logis dan terstruktur, sehingga kamu terbiasa berpikir langkah demi langkah sebelum menerapkan solusi dalam bentuk kode bahasa pemrograman (koding).



Untuk mengetahui bentuk algoritma dasar, contoh algoritma untuk pemecahan masalah, dan contoh representasi algoritma dalam bentuk *flowchart* silakan pindai QR Code berikut.

<https://buku.kemdikbud.go.id/s/kka10ax>



Pindai Aku!

Setelah kamu memahami materinya, kerjakan aktivitas berikut secara mandiri ataupun berkelompok.



Ayo Berdiskusi!

Jenis Aktivitas : Kelompok

No Aktivitas : AP-K10-01

Menyusun Algoritma Sehari-hari: Langkah Menuju Sekolah

Aktivitas ini merupakan aktivitas berkelompok yang terdiri atas 2-3 orang. Setiap hari kamu melakukan aktivitas pergi dari rumah menuju sekolah hingga duduk di kelas. Susunlah sebuah algoritma langkah menuju sekolah.

1. Tuliskan urutan langkah “Langkah Menuju Sekolah” dalam bentuk narasi. Dari bangun tidur hingga duduk di bangku kelas.
2. Pilihlah satu orang perwakilan untuk membacakan algoritma yang telah kalian tulis.
3. Kelompok lain memperhatikan dan mengevaluasi algoritma yang dibacakan:
 - a. Apakah langkahnya sudah logis dan tidak melompat-lompat?
 - b. Adakah ada perbedaan langkah antara kelompok yang satu dengan yang lain? Jika ada sebutkan!
 - c. Apakah ada langkah yang dapat disederhanakan?
4. Kelompok lain dipersilakan untuk memberikan tanggapan dan mendiskusikannya.



Ayo Menganalisis!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-02

Membuat Algoritma: Langkah Menuju Sekolah dengan Pilihan Jalur

Setiap pagi kamu pergi ke sekolah. Biasanya kamu bangun pukul 05.00. Jika kamu bangun tepat waktu, maka kamu dapat menikmati sarapan dengan tenang dan berangkat jalan kaki lewat jalan biasa. Tapi jika kamu bangun terlambat, maka kamu harus memilih antara dua pilihan berikut.

1. **Naik sepeda sendiri** jika ada sepeda.
2. **Berlari** jika tidak ada sepeda.

Tuliskan algoritma dalam bentuk *pseudocode* berdasarkan cerita di atas.

1. Baca dan memahami soal cerita dengan baik.
2. Tuliskan urutan langkah menggunakan bentuk narasi.
3. Mintalah penjelasan ulang konsep penggunaan kondisi IF-THEN pada guru jika kamu belum benar-benar memahaminya.
4. Ubah urutan langkah tersebut ke dalam bentuk *pseudocode*.
5. Presentasikan hasil algoritma dalam bentuk *pseudocode* untuk mengetahui apakah sudah benar atau masih ada kekurangan.
6. Minta teman yang lain untuk memperhatikan dan memberikan komentar mengenai hal berikut.
 - a. Apakah *pseudocode* sudah lengkap?
 - b. Apakah struktur kondisi yang digunakan sudah benar?
7. Mintalah kepada gurumu untuk memberikan umpan balik dan koreksi.



Ayo Berdiskusi!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-03

Membuat *Flowchart* Menghitung Nilai Rata-Rata

Buatlah *flowchart* menghitung nilai rata-rata dari tiga mata pelajaran (matematika, bahasa Indonesia, dan bahasa Inggris).

1. Bagilah tiga tahapan input-proses-output:
 - a. **Input** : Nilai matematika, bahasa Indonesia, dan bahasa Inggris.

- b. **Proses** : Menjumlahkan ketiga nilai, kemudian membaginya dengan tiga.
 - c. **Output** : Menampilkan nilai rata-rata.
2. Gambarlah *flowchart* menggunakan simbol standar sesuai proses yang termasuk input, proses, dan *output*.
 3. Presentasikan hasil *flowchart* dan jelaskan alur logika yang telah kamu buat.



Ayo Mencoba!



Jenis Aktivitas : Kelompok

No Aktivitas : AP-K10-04

Bermain *Flowchart*

Dalam aktivitas ini, kamu akan bekerja secara berkelompok untuk menyusun potongan-potongan langkah menjadi sebuah *flowchart* yang utuh dan logis. Kalian akan bergiliran meletakkan potongan/ menuliskan langkah atau simbol *flowchart* pada papan atau kertas kerja. Setiap langkah yang dipilih harus saling berkesinambungan dan sesuai dengan alur algoritma yang benar. Kerja sama dan komunikasi yang baik sangat membantu dalam menyelesaikan tantangan ini. Siap bermain dan berpikir logis? Mari kita mulai!

Petunjuk:

1. Buatlah kelompok yang beranggotakan 3–5 orang.
2. Pahami dan patuhi aturan permainan berikut.
 - a. Setiap kelompok mendapat satu kartu instruksi topik secara acak yang akan dibagikan oleh guru.

- b. Anggota kelompok menyusun langkah-langkah proses dalam bentuk *flowchart* secara bergantian.
- c. Tidak diperbolehkan menggambar dua simbol dalam satu kesempatan.
3. Kelompok mendiskusikan urutan langkah-langkah algoritma berdasarkan topik yang didapat.
4. Secara bergantian, setiap anggota kelompok maju ke depan untuk menggambar satu simbol *flowchart* hingga membentuk *flowchart* yang lengkap.
5. Setelah *flowchart* selesai disusun, kelompok diminta untuk menjelaskan alur proses dari awal hingga akhir.

Algoritma memberikan gambaran langkah-langkah yang jelas dalam menyelesaikan suatu masalah. Algoritma tidak hanya berhenti pada perencanaan langkah-langkah, tetapi juga perlu diwujudkan dalam bentuk instruksi yang dapat dipahami komputer. Proses penerjemahan dari algoritma ke dalam instruksi inilah yang dikenal sebagai logika pemrograman. Pada subbab berikutnya, kamu akan mempelajari implementasi logika pemrograman melalui *flowchart*.

B. Implementasi Logika Pemrograman dalam *Flowchart*

Logika berasal dari bahasa Yunani yaitu *logos* yang berarti ilmu. Logika dapat diartikan ilmu yang mengajarkan cara berpikir untuk melakukan kegiatan dengan tujuan tertentu. Sedangkan program adalah kumpulan instruksi berupa pernyataan yang ditulis dengan menggunakan bahasa pemrograman yang melibatkan pemilihan struktur data. Logika pemrograman adalah landasan utama untuk menentukan cara kerja suatu program komputer yang memungkinkan untuk menyelesaikan persoalan dan melaksanakan berbagai tugas melalui penulisan kode bahasa pemrograman. Logika pemrograman memiliki peran yang sangat penting, karena membantu seorang pengembang menulis kode program yang sesuai dengan kebutuhan, mudah dibaca, dikelola, dan dikembangkan. Prinsip logika pemrograman adalah memanfaatkan

pola pikir yang terstruktur dan sistematis untuk mengatur program dalam mengolah data, mengambil keputusan, serta menjalankan perintah. Logika ini tidak bergantung pada aturan sintaks dari bahasa pemrograman tertentu, melainkan berfokus pada pendekatan dalam menyelesaikan masalah dan penerapan solusi secara logis dalam kode program yang merupakan dasar dari semua bahasa pemrograman, baik itu C++, Pascal, Java, atau Python. Logika ini membantu pengembang untuk merancang alur kerja yang efisien sehingga program dapat berjalan dengan benar.

Logika pemrograman memiliki kaitan erat dengan algoritma pemrograman. Jika logika pemrograman adalah cara penyelesaian suatu masalah dengan prinsip dan aturan tertentu, maka algoritma pemrograman adalah urutan atau langkah-langkah sistematis yang disusun untuk mewujudkan penyelesaian masalah tersebut. Penguasaan logika pemrograman yang baik memungkinkan program berjalan lebih efektif, meminimalkan kesalahan, dan meningkatkan efisiensi. Selain itu, pemahaman yang kuat terhadap logika ini mempermudah proses belajar bahasa pemrograman baru dan mendorong munculnya solusi optimasi program yang inovatif pada pemrograman yang kompleks.

Berikut jenis-jenis logika pemrograman yang dapat kamu pelajari. Simak dan pelajari dengan saksama.

1. Logika Urutan (*Sequence Logic*)

Logika urutan merupakan instruksi secara berurutan dari atas ke bawah sebagaimana tertulis dalam kode program. Setiap pernyataan dieksekusi satu per satu sesuai dengan alur logika yang telah dirancang tanpa adanya percabangan atau pengulangan.

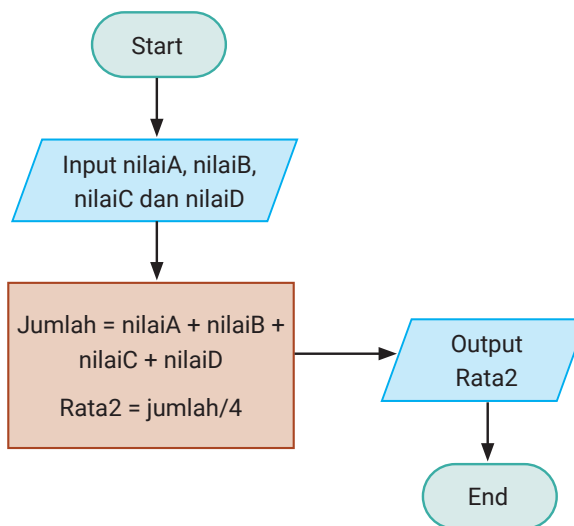
Contoh:

Menghitung rata-rata nilai. Program ini menerima sejumlah nilai, menghitung totalnya, kemudian membagi total tersebut dengan jumlah nilai untuk mendapatkan rata-rata.

Logika:

1. Mulai.
2. Terima nilai-nilai dari *user*.
3. Jumlahkan semua nilai tersebut.
4. Bagi total nilai dengan jumlah nilai yang diberikan.
5. Tampilkan hasil rata-rata.
6. Selesai.

Flowchart:



2. Logika Seleksi/Percabangan (*Selection Logic*)

Logika yang digunakan untuk membuat keputusan berdasarkan kondisi. Pengambilan keputusan berdasarkan kondisi ini, menggunakan struktur seperti *if*, *if-else*, atau *switch-case*. Kondisi ini dibentuk menggunakan operator logika aritmatika, perbandingan, dan boolean untuk mengevaluasi nilai dan menentukan jalur eksekusi.

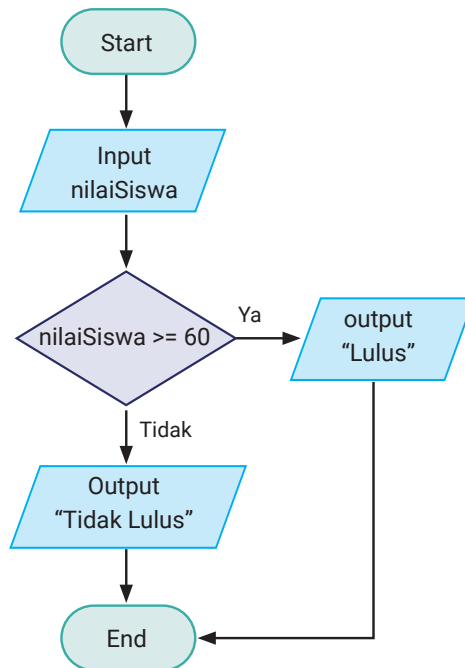
Contoh:

Menentukan kelulusan peserta didik berdasarkan nilai. Program ini memeriksa apakah nilai peserta didik di atas ambang batas kelulusan.

Logika:

1. Mulai.
2. Masukkan nilai peserta didik.
3. Jika nilai ≥ 60 , maka peserta didik dinyatakan lulus.
4. Jika tidak, peserta didik dinyatakan tidak lulus.
5. Tampilkan hasil kelulusan peserta didik.
6. Selesai.

Flowchart:



3. Logika Perulangan (Iteration/Looping Logic)

Logika ini digunakan untuk mengeksekusi blok kode secara berulang selama kondisi tertentu terpenuhi. Struktur perulangan yang digunakan adalah sebagai berikut.

- a. *For*
- b. *While do*
- c. *Repeat Until*

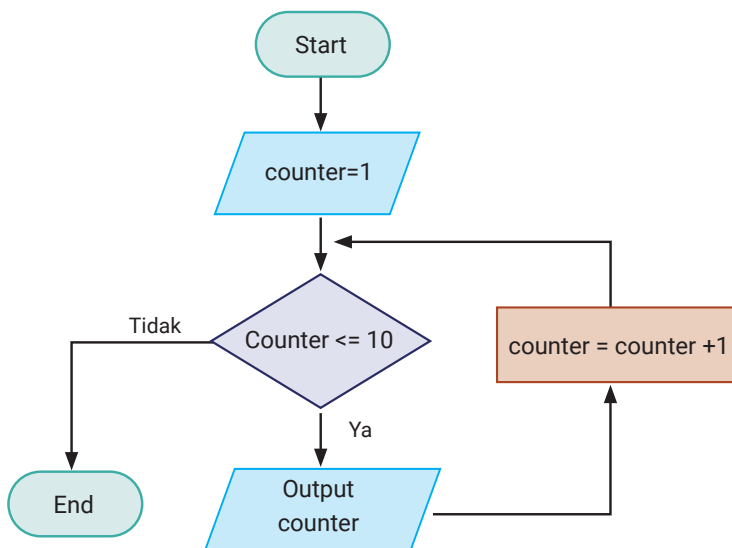
Contoh:

Mencetak angka dari 1 hingga 10. Program ini menggunakan *loop* untuk mencetak serangkaian angka.

Logika:

1. Mulai.
2. Mulai dengan angka 1.
3. Selama angka ≤ 10 , cetak angka tersebut.
4. Tambahkan 1 ke angka tersebut.
5. Ulangi proses hingga angka melebihi 10.
6. Selesai.

Flowchart:





Ayo Berlatih!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-05

Mengubah Teks ke *Flowchart*: Prosedur Menulis Surat Izin

1. Baca dan pahami teks prosedur berikut ini.

Cara Menulis Surat Izin Tidak Masuk Sekolah

- a. Siapkan kertas dan alat tulis.
 - b. Tulis tanggal surat di pojok kanan atas.
 - c. Jika surat ditulis oleh orang tua, gunakan format orang tua.
 - d. Jika ditulis olehmu, gunakan format pribadi.
 - e. Tulis nama lengkap dan kelasmu.
 - f. Jelaskan alasan tidak masuk (sakit, keperluan keluarga).
 - g. Tambahkan permohonan maaf atas ketidakhadiran.
 - h. Tutup surat dengan tanda tangan pengirim.
 - i. Serahkan surat kepada wali kelas atau TU.
2. Tandai bagian yang menurutmu penting:
 - a. Langkah tetap (urutan biasa)
 - b. Langkah yang menggunakan percabangan.
 3. Buatlah *flowchart* berdasarkan pada langkah pertama.
 4. Presentasikan hasil konversi teks menjadi bentuk *flowchart* yang telah kamu buat.



Ayo Berlatih!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-06

Mengubah Logika Pengenalan Jenis Kalimat Menjadi *Flowchart*

1. Pelajari dan pahami perbedaan antara kalimat perintah, kalimat tanya, dan kalimat deklaratif, lalu berikan masing-masing contohnya!”

Contoh Kalimat “Tutup pintunya!”

Kalimat di atas termasuk kalimat apa? Bagaimana kamu mengetahuinya?

2. Analisis logika klasifikasi kalimat berikut.

- a. Masukkan sebuah kalimat.
- b. Lihat akhir kalimat: Apakah menggunakan tanda tanya (?)
- c. Jika ya → kalimat tanya
- d. Jika tidak, cek apakah ada kata kerja bentuk perintah atau tanda seru (!).
- e. Jika ya → kalimat perintah
- f. Jika tidak → kalimat berita

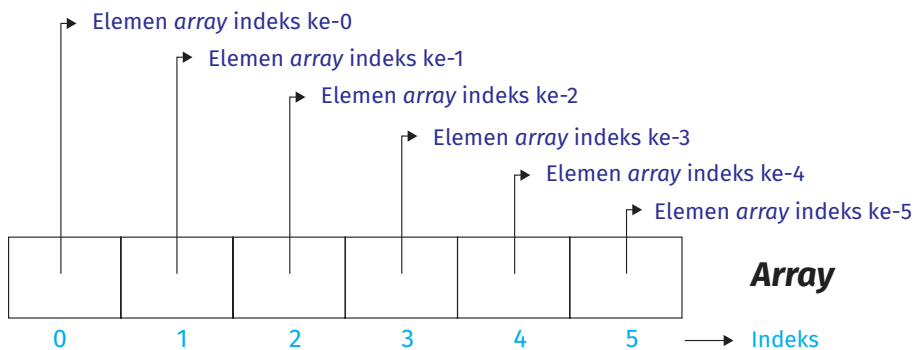
3. Gambarkan *flowchart* berdasarkan logika.
4. Temanmu dapat melakukan uji coba *flowchart* yang telah dibuat.
5. Persentasikan hasil *flowchat* yang telah kamu buat.

C. Algoritma Terstruktur dan Penerapannya

1. Array

Struktur data adalah cara mengatur dan menyimpan data agar informasi dapat digunakan secara efisien. Struktur data dapat berbentuk *array*, *list*, *queue*, atau *stack*. Ilustrasi struktur data pada subbab ini menggunakan *array*.

Array adalah struktur data yang digunakan untuk menyimpan sekumpulan nilai dalam satu variabel dengan tipe data yang sama. Setiap nilai yang disimpan dalam *array* disebut elemen, dan setiap elemen memiliki indeks atau posisi yang unik. Indeks *array* biasanya dimulai dari angka 0, sehingga elemen pertama berada pada indeks 0, elemen kedua di indeks 1, dan seterusnya. Jika indeks pertama dimulai dari 0, maka elemen kedua di indeks 2, dan seterusnya.



Gambar 2.1 Contoh Array

Membuat Kamus dalam Array

Kamus

A : array of Integer

Artinya *array* A setiap elemennya bernilai integer.

Kamus

A : array of String

Artinya *array* A setiap elemennya bernilai *string*.

Menulis isi *array*

Contoh:

$A = [10, 2, 30, 4]$

Artinya:

$A[0] = 10$

$A[1] = 2$

$A[3] = 30$

$A[4] = 4$

Operasi Penjumlahan dalam *Array*

Jumlah = $A[1] + A[4]$

Artinya menjumlahkan elemen *array* indeks ke-1 dan indeks ke-4, yaitu

Jumlah = $2 + 4 = 6$

2. Penggunaan Pengondisian Bertingkat (*Nested If*)

Nested if adalah sebuah struktur pengondisian di dalam pemrograman. Satu pernyataan *if* berada di dalam blok *if* atau *else* lainnya, sehingga memungkinkan pengambilan keputusan yang lebih kompleks dan bertingkat. Struktur ini berguna ketika sebuah keputusan bergantung pada hasil dari keputusan sebelumnya.

Cara kerja *nested if*, ketika program menemukan sebuah pernyataan *if*, program akan memeriksa apakah kondisi tersebut benar atau salah. Jika benar, maka kode di dalam blok *if* tersebut akan dijalankan. Jika ada pernyataan *if* lain di dalam blok ini, maka program akan lanjut memeriksa kondisi baru tersebut. Proses ini berlangsung bertingkat-tingkat sesuai dengan kebutuhan logika program. Jika kondisi pertama tidak terpenuhi, maka program dapat menjalankan blok *else*, yang juga bisa berisi *if* lain sebagai cabang pengondisian tambahan.

Bentuk Algoritma *Nested If*

```
if (kondisi pertama) then
    if (kondisi kedua) then
        .....
    endif
endif
```

Contoh Algoritma Prediksi Nilai Akhir

Algoritma PredikatNilaiAkhir

Kamus

```
nilai : integer
predikat : string
```

Deskripsi

```
Output("Masukkan nilai ujian:")
Input(nilai)

if nilai >= 90 then
    predikat = "A"
else if nilai >= 75 then
    predikat = "B"
else if nilai >= 60 then
    predikat = "C"
else
    predikat = "D"
endif

Output("Predikat nilai: ", predikat)
```

EndAlgoritma

3. Pengulangan Bersarang (Nested Loop)

Pengulangan bersarang atau *nested loop* adalah sebuah struktur kontrol yang terdiri atas satu pengulangan (*loop*) di dalam pengulangan lainnya. Pada saat satu *loop* berada di dalam blok kode *loop* lain, maka keduanya membentuk struktur bersarang. Konsep ini memungkinkan program menjalankan sejumlah besar perulangan yang kompleks dan digunakan secara luas dalam berbagai skenario, seperti mencetak pola, mengakses elemen *array* multidimensi, dan melakukan simulasi berulang.

Pengulangan bersarang biasanya menggunakan jenis *loop* yang sama, seperti *for* di dalam *for*, atau kombinasi berbeda seperti *while* di dalam *for*. Saat *loop* luar berjalan sekali, *loop* dalam akan berjalan sepenuhnya sebelum *loop* luar melanjutkan ke iterasi berikutnya. Artinya, *loop* dalam akan mengulang sebanyak jumlah perulangannya untuk setiap satu kali iterasi *loop* luar.

Bentuk Algoritma *Nested Loop*

```
for i = 1 to n do  
    for j=1 to n do  
        .....  
        .....  
    endFor  
endFor
```

Contoh sederhana pengulangan bersarang adalah mencetak tabel perkalian. Sebagai contoh, untuk mencetak hasil perkalian dari 1 hingga 3 terhadap 1 hingga 5 akan menghasilkan seperti berikut.

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15

Algoritma CetakTabelPerkalian

Kamus

i, j : integer

Deskripsi

```
for i = 1 to 3 do
    for j = 1 to 5 do
        Hasil = i * j
        Output( Hasil )
    endfor
endfor
```

EndAlgoritma

4. Algoritma Pencarian (*Searching*)

a. *Sequential Search* atau *Linear Search*

Sequential search atau sering disebut juga sebagai *linear search* adalah salah satu algoritma pencarian paling sederhana yang digunakan untuk mencari suatu elemen dalam sebuah struktur data seperti *array*. Algoritma ini bekerja dengan cara memeriksa elemen satu per satu, dimulai dari indeks pertama sampai dengan elemen terakhir. Pencarian dimulai dengan membandingkan elemen pertama dengan nilai yang dicari. Jika berhasil, maka pencarian selesai. Jika tidak, algoritma melanjutkan ke elemen berikutnya dan mengulangi proses tersebut sampai dengan indeks terakhir. Jika sampai dengan indeks terakhir tidak ditemukan, maka hasilnya adalah “tidak ditemukan”.

Contoh algoritma pencarian *sequential search*.

Jika diketahui:

- data *array* sebagai berikut A =

1	5	10	7	15
---	---	----	---	----

- n adalah banyaknya data yaitu 5.
- x adalah nilai yang dicari.

- *i* adalah indeks *array*.
- *posisi* adalah posisi elemen *array*.

Algoritma SequentialSearch

Kamus

A : array of integer
 n : integer
 x : integer // nilai yang dicari
 i : integer // indeks array
 posisi : integer

Deskripsi

// Data A

A[0] = 1

A[1] = 5

A[2] = 10

A[3] = 7

A[4] = 15

// Jumlah Elemen A

n = 5

Output("Nilai yang dicari?")

Input(x)

posisi = -1

for i = 0 to n - 1 **do**

if A[i] = x **then**

 posisi = i

 Output("Ketemu, nilai yang dicari berada pada indeks ke-", **posisi**)

break

endIf

endFor

```

    if posisi = -1 then
        Output("Tidak ketemu")
    endIf

EndAlgoritma

```

b. Binary Search

Binary search adalah algoritma pencarian yang efisien untuk menemukan posisi suatu nilai dalam *array* yang telah diurutkan. Algoritma ini bekerja dengan membagi *array* menjadi dua bagian secara berulang, lalu membandingkan nilai yang dicari dengan elemen tengah *array*. Jika nilai yang dicari lebih kecil dari elemen tengah, pencarian dilanjutkan ke bagian kiri. Jika lebih besar, maka ke bagian kanan. Proses ini terus berulang hingga nilai ditemukan atau seluruh bagian sudah diperiksa.

Contoh algoritma pencarian dengan *binary search*.

Algoritma SequentialSearch

Kamus

```

A : array of integer
n : integer
x : integer // nilai yang dicari
i : integer // indeks array
awal : integer
akhir : integer
tengah : integer
posisi : integer

```

Deskripsi

```

// Data A
A[0] = 1
A[1] = 5
A[2] = 10
A[3] = 7
A[4] = 15

```

```

// Jumlah Elemen A
n = 5
awal = 0
akhir = n - 1
posisi = -1

Output("Nilai yang dicari?")
Input(x)

while (awal <= akhir) do
    tengah = (awal + akhir) div 2
    if A[tengah] = x then
        posisi = tengah
        Output("Ketemu, nilai yang dicari berada pada indeks
ke-", posisi)
        break
    else if x < A[tengah] then
        akhir = tengah - 1
    else
        awal = tengah + 1
    endIf
endWhile

if posisi = -1 then
    Output("Tidak ketemu")
endIf

EndAlgoritma

```

Tabel 2.1 Perbandingan Sequential Search dan Binary Search

Aspek Perbandingan	<i>Sequential Search</i>	<i>Binary Search</i>
Cara Kerja	Memeriksa elemen satu per satu dari awal.	Membagi <i>array</i> dan mencari di bagian tengah.

Aspek Perbandingan	<i>Sequential Search</i>	<i>Binary Search</i>
Syarat Data	Tidak perlu terurut.	Harus diurutkan terlebih dahulu.
Jumlah Perbandingan	Sebanyak jumlah elemen (misalnya 100 data = 100 kali).	Jauh lebih sedikit, cukup dibagi 2 terus-menerus hingga berhasil.
Jumlah <i>Loop</i> Maksimal	Jika 100 data, maka sebanyak jumlah elemen (100 kali).	Jika 100 data, cukup 7 kali. Jauh lebih sedikit <i>loop</i> nya dibanding dengan <i>Sequential Search</i> .

5. Algoritma Pengurutan (*Sorting*)

a. *Selection Sort*

Selection sort adalah salah satu algoritma pengurutan (*sorting*) sederhana yang bekerja dengan cara memilih elemen terkecil atau terbesar (tergantung urutan yang diinginkan) dari data yang belum terurut, lalu menukarnya dengan elemen di posisi awal. Proses ini diulangi untuk bagian data yang tersisa hingga seluruh data terurut.

Contoh *Selection Sort*:

Jika kamu memiliki *array*: [6, 3, 8, 5, 2]

Langkah-langkah *selection sort*:

- Cari nilai terkecil dari indeks 0 hingga akhir. Didapat hasil angka **2** yang paling kecil, maka tukar dengan posisi pertama.

Hasil: [2, 3, 8, 5, 6]

- Cari nilai terkecil dari indeks 1 hingga akhir, maka Didapat hasil angka **3**, tidak perlu ditukar.

Hasil: [2, 3, 8, 5, 6]

- c. Cari nilai terkecil dari indeks 2 hingga akhir.
Didapat hasil angka **5**, maka tukar dengan indeks ke-2.

Hasil: [2, 3, 5, 8, 6]

- d. Cari nilai terkecil dari indeks 3 hingga akhir.
Didapat hasil angka **6**, maka tukar dengan indeks ke-3.

Hasil: [2, 3, 5, 6, 8]

b. *Insertion Sort*

Insertion sort adalah salah satu algoritma pengurutan (*sorting*) yang bekerja seperti seseorang menyusun kartu di tangan saat bermain kartu. Setiap elemen dipilih satu per satu dan ditempatkan pada posisi yang sesuai dalam bagian data yang sudah terurut. Proses ini diulang hingga seluruh elemen dalam *array* berada dalam urutan yang benar.

Contoh *Insertion Sort*:

Misalkan kamu mempunyai *array*: [5, 2, 4, 6, 1]

Langkah-langkahnya:

- Mulai dari elemen ke-2 (yaitu 2), bandingkan dengan 5, karena $2 < 5$, geser 5 ke kanan, masukkan 2 ke indeks 0
Hasil: [2, 5, 4, 6, 1]
- Elemen berikutnya 4, bandingkan ke belakang, $5 > 4$, maka geser 5, masukkan 4 ke indeks 1.
Hasil: [2, 4, 5, 6, 1]
- Elemen berikutnya 6, ini nilainya sudah lebih besar dari sebelumnya, maka biarkan
Hasil: [2, 4, 5, 6, 1]
- Elemen berikutnya 1, bandingkan dengan sebelumnya, jika elemen sebelumnya lebih besar maka digeser. Jadi geser 6, 5, 4, 2 ke kanan, dan masukkan 1 ke indeks 0
Hasil: [1, 2, 4, 5, 6]

Tabel 2.2 Perbandingan *Selection Sort* dan *Insertion Sort*

Asepk Perbandingan	<i>Selection Sort</i>	<i>Insertion Sort</i>
Metode Pengurutan	Memilih elemen terkecil lalu menukarnya.	Menyisipkan elemen ke posisi yang tepat.
Cara Kerja	Seleksi elemen terkecil dan tukar ke depan.	Bandingkan ke belakang, geser dan sisipkan.
Jumlah Pertukaran	Lebih sedikit dari <i>insertion sort</i> .	Lebih banyak, karena sering menggeser elemen.



Ayo Berlatih!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-07

Membuat Algoritma menggunakan *Nested If*

- Buatlah program otomatis untuk mengetahui kelulusan peserta didik. Kelulusan dipengaruhi oleh nilai akhir dan kehadiran, dengan ketentuan sebagai berikut.
 - Nilai akhir minimal 75.
 - Kehadiran minimal 80%.
 - Jika nilai antara 70–74, maka peserta didik tetap lulus jika kehadiran di atas 90%, karena dianggap rajin walaupun nilainya kurang sedikit.
 - Jika **salah satu tidak memenuhi**, maka peserta didik **tidak lulus**.
- Buatlah algortimanya dalam bentuk *flowchart*.
- Minta temanmu untuk melakukan uji coba *flowchart* yang telah dibuat.
- Persentasikan hasil *flowchat* yang telah kamu buat.



Ayo Berlatih!



Jenis Aktivitas : Berpasangan

No Aktivitas : AP-K10-08

Melakukan Pengurutan Data

1. Kamu akan membuat program untuk mengurutkan data secara otomatis dari data dalam bentuk *array* berikut.

Data *array*: [15, 25, 13, 5, 30, 40, 7]

2. Lakukan simulasi dan catat setiap langkah apabila pengurutan menggunakan *selection sort*.
3. Jika data sudah terurut, ubah salah satu elemen *array* menjadi seperti berikut.

Data *array*: [1, 25, 13, 5, 2, 40, 7]

4. Lakukan analisis, apakah simulasi langkah yang sudah dibuat pada tahap 6 dapat langsung diterapkan? Jika tidak, berikan alasannya.
5. Presentasikan hasilnya dengan menjelaskan simulasi yang sudah dicatat dan hasil analisisnya.

6. Algoritma Rule Based

Algoritma *rule based* atau berbasis aturan adalah salah satu bentuk pengambilan keputusan paling dasar dalam kecerdasan artifial (KA) dan pemrograman. Sistem ini bekerja berdasarkan aturan yang ditulis oleh manusia, biasanya dalam bentuk *If-Then* atau *nested if*. Aturan-aturan ini digunakan untuk menentukan keputusan yang harus diambil ketika kondisi terpenuhi.

Contoh

Algoritma DiagnosaKesehatan

Kamus

suhubadan : float
batuk : string
sakitkepala : string

Deskripsi

```
Output("berapa suhu badan ? ")
Input(suhubadan)
Output("apakah terdapat batuk ? ya/tidak ")
Input(batuk)
Output("apakah terdapat sakit kepala ? ya/tidak ")
Input(sakitkepala)

if suhubadan > 37.5 then
    if batuk="ya" then
        if sakitkepala="ya" then
            Output("Diagnosa: terkena Flu atau Infeksi
Virus")
        else
            Output("Diagnosa: Demam dan Batuk")
        endIf
    else
        if sakitkepala="ya" then
            Output("Diagnosa: Demam biasa")
        else
            Output("Diagnosa: Demam ringan")
        endIf
    endIf
else
```

```

if batuk="ya" then
    if sakitkepala="ya" then
        Output("Diagnosa: kelelahan")
    else
        Output("Diagnosa: Batuk ringan")
    endIf
else
    if sakitkepala="ya" then
        Output("Diagnosa: Sakit kepala ringan")
    else
        Output("Kondisi sehat")
    endIf
endIf
endIf

EndAlgoritma

```

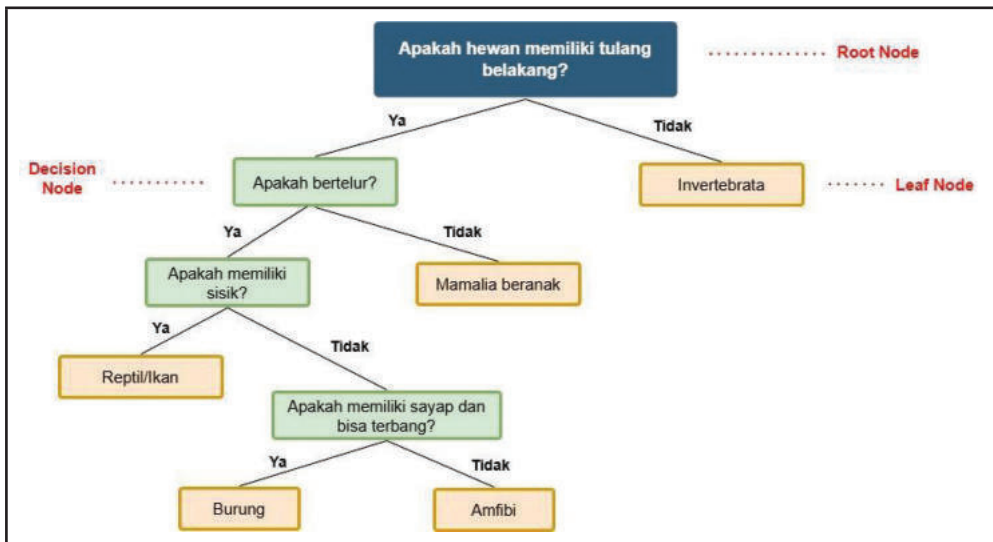
7. Algoritma Machine Learning (Pembelajaran Mesin)

a. Decision Tree

Decision tree adalah metode atau algoritma yang menggunakan struktur berbentuk pohon untuk membantu dalam pengambilan keputusan atau prediksi.

Setiap titik percabangan pada pohon disebut *node* yang mewakili suatu pertanyaan atau kondisi. Sedangkan cabang-cabangnya menyatakan kemungkinan hasil atau langkah berikutnya untuk mencapai hasil akhir atau kesimpulan akhir.

Contoh *Decision Tree* “Mengelompokkan Jenis Hewan”

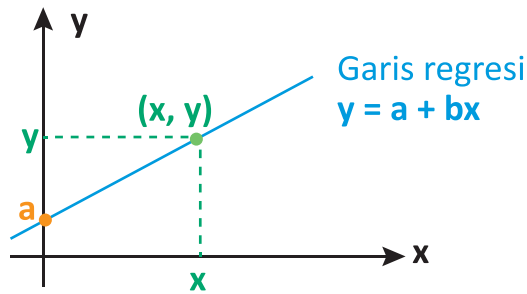


Keterangan:

- 1) **Root Node** adalah pertanyaan pertama pada pohon keputusan, tempat memulai proses memilih.
- 2) **Decision Node** adalah pertanyaan lanjutan yang muncul setelah jawaban dari *root node* atau pertanyaan sebelumnya.
- 3) **Leaf Node** adalah hasil akhir dari proses untuk menentukan keputusan.

b. *Linear Regression* (Regresi Linear)

Regresi linear adalah metode atau algoritma yang digunakan untuk memprediksi suatu nilai berdasarkan hubungan antara dua variabel X (independen) yang nilainya ditentukan atau diketahui lebih dulu untuk melihat pengaruhnya terhadap variabel lain dan Y (dependen) yang nilainya dipengaruhi berdasarkan variabel independen. Prinsip dari regresi linear sederhana adalah antara variabel Y (dependen) dengan variabel X (independen) harus memiliki sifat hubungan sebab akibat. Ilustrasi regresi linear dapat dilihat pada gambar berikut.



Gambar 2.2 Regresi Linear

Garis miring pada gambar menunjukkan hubungan variabel X dan Y. Setiap perubahan X akan memengaruhi nilai Y sesuai dengan kemiringan (*slope*) garis atau koefisien regresi. Secara matematis rumus regresi linear adalah sebagai berikut.

$$\bar{Y} = a + b\bar{X}$$

$\bar{Y}$ = variabel dependen atau nilai rata-rata variabel Y.

$\bar{X}$ = variabel independen atau nilai rata-rata variabel X.

a = konstanta (*intercept*) yang menyatakan titik potong garis dengan sumbu Y.

b = kemiringan garis (*slope*), yang menunjukkan besarnya perubahan nilai Y untuk setiap kenaikan satu satuan X atau dapat disebut juga dengan koefisien regresi.

Contoh: Prediksi Berat Badan dari Tinggi Badan

Urutan langkah untuk memprediksi “berat badan dari tinggi badan” dengan menggunakan regresi linear adalah sebagai berikut.

1. Persiapan Data

- a. Menentukan variabel X dan Y. X adalah tinggi badan dengan satuan cm dan Y adalah berat badan dengan satuan kg.
- b. Mengumpulkan data tinggi badan dan berat badan. Setelah data terkumpul buatlah tabel data seperti berikut.

No.	X (Tinggi Badan)	Y (Berat Badan)
1.	150 cm	45 kg
2.	155 cm	50 kg
3.	160 cm	52 kg
4.	165 cm	55 kg
5.	170 cm	60 kg

2. Hitung rata-rata!

a. Hitung rata-rata X

$$\text{Rata-Rata } X(\bar{X}) = \frac{(150 + 155 + 160 + 165 + 170)}{5} = 160$$

b. Hitung rata-rata Y

$$\text{Rata-Rata } Y(\bar{Y}) = \frac{(45 + 50 + 52 + 55 + 60)}{5} = 52,4$$

3. Hitung slope (kemiringan) b!

Untuk menghitung slope gunakan rumus:

$$b = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sum(X - \bar{X})^2}$$

No.	X	Y	$(X - \bar{X})$	$(Y - \bar{Y})$	$(X - \bar{X})(Y - \bar{Y})$	$(X - \bar{X})^2$
1.	150 cm	45 kg	150-160 = -10	45-52,4= -7,4	74	100
2.	155 cm	50 kg	155-160 = -5	50-52,4= -2,4	12	25
3.	160 cm	52 kg	160-160 = 0	52-52,4= -0,4	0	0
4.	165 cm	55 kg	165-160 = 5	55-52,4= 2,6	13	25
5.	170 cm	60 kg	170-160 = 10	60-52,4= 7,6	76	100
Jumlah (Σ)					175	250

$$b = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sum(X - \bar{X})^2} = \frac{175}{250} = 0,7$$

4. Hitung *intercept* a !

$$\bar{Y} = a + b\bar{X}$$

$$a = \bar{Y} - b\bar{X} = 52,4 - (0,7 \times 160) = -59,6$$

5. Prediksi berat badan dengan persamaan regresi!

Dengan menggunakan persamaan regresi, kamu dapat melakukan prediksi berat badan jika tinggi badan diketahui. Misalkan jika diketahui tinggi badan seseorang 175 cm, prediksi berat badannya adalah sebagai berikut.

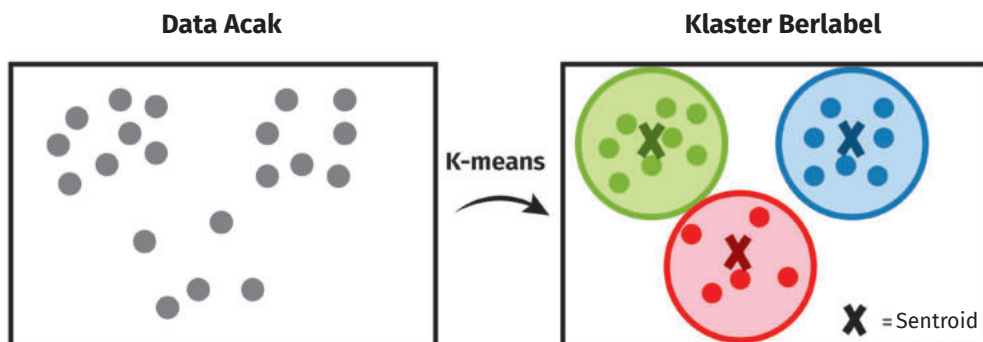
$$\bar{Y} = a + b\bar{X}$$

$$\bar{Y} = -59,6 + (0,7 \times 175) = 62,9 \text{ kg}$$

Jadi prediksi berat badannya adalah 62,9 kg.

c. *K-Means Clustering*

K-means clustering adalah metode atau algoritma yang digunakan untuk mengelompokkan data menjadi beberapa kelompok (klaster) berdasarkan kemiripan datanya. K menunjukkan jumlah kelompok (klaster) yang akan dibuat. Ilustrasi *K-means clustering* dapat dilihat pada gambar berikut.



Gambar 2.3 K-Means Clustering

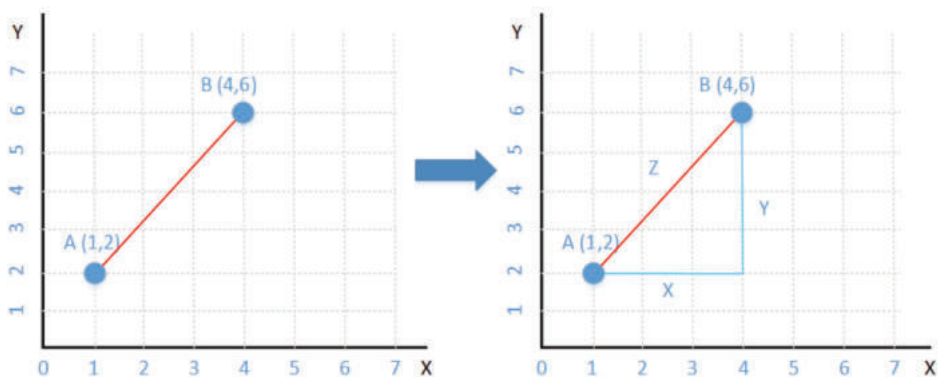
Pada gambar 2.3 titik-titik abu-abu merupakan representasi data yang belum dikelompokkan yang masih tersusun secara acak. Kemudian dilakukan proses pengelompokan dengan menggunakan algoritma *K-means* dengan membagi menjadi tiga kelompok (klaster) yaitu hijau, merah, biru. X di tengah adalah sentroid masing-masing klaster, yang mewakili pusat kelompok. Setiap klaster mengelompokkan titik-titik yang memiliki jarak terdekat terhadap sentroidnya dibandingkan sentroid lain. Untuk lebih jelasnya, perhatikan urutan langkah algoritma *K-means clustering* sebagai berikut.

- 1) Tentukan jumlah klaster.
- 2) Pilih titik pusat awal (sentroid) secara acak.
- 3) Hitung jarak setiap data ke masing-masing sentroid.

Contoh di 2 dimensi ($n = 2$) dengan titik $A=(1,2)$ dan $B=(4,6)$, jika titik A dan B dibuat tabel akan terlihat seperti berikut.

Titik	x	y
A	1	2
B	4	6

Jika titik A dan B dibuat grafik dan disambungkan setiap titiknya akan terlihat sebagai berikut.



Untuk mencari jarak A ke B, dapat menggunakan pendekatan rumus Phytagoras.

$$Z^2 = X^2 + Y^2 = 3^2 + 4^2$$

$$z = \sqrt{9 + 16} = \sqrt{25} = 5$$

Jadi jarak dari titik A ke titik B adalah 5.

4. Tentukan setiap data masuk ke kluster mana.
5. Hitung pusat (sentroid) baru untuk setiap kluster.
6. Cek perubahan pusat kluster, jika pusat sentroid berubah, ulangi proses dari langkah 3. Jika pusat tidak berubah, hentikan proses.
7. Selesai.

Contoh: Pengelompokan Teman Berdasarkan 3 Hobi

1. Persiapan Data, diketahui data yang diperoleh melalui pendataan, dengan angka 1 menyatakan hobi dan angka 0 menyatakan bukan hobinya.

Teman	Hobi		
	Olahraga	Membaca	Musik
A	1	0	1
B	1	1	0
C	0	1	1
D	0	1	0
E	1	0	0
F	0	0	1

2. Tentukan jumlah kluster.
Misalkan ditentukan jumlah kluster (K) = 2
3. Pilih titik pusat awal (sentroid) secara acak.
 - Titik pusat (sentroid) untuk kluster 1 (C_1) adalah Teman A (1,0,1)
 - Titik pusat (sentroid) untuk kluster 2 (C_2) adalah Teman D (0,1,0)

4. Hitung jarak setiap data ke masing-masing sentroid.
- Kamu harus menghitung semua jarak teman ke kluster 1 (C1) dan kluster 2 (C2).

Misalkan menghitung jarak dari Teman B (1,1,0) ke kluster 1 ($C1=(1,0,1)$)

i	x = B	y = C1
1	1	1
2	1	0
3	0	1

Jadi jarak antara titik B ke kluster 1 (C1) adalah 1,41.

- Dengan melakukan perhitungan yang sama, maka akan diperoleh jarak setiap data sebagai berikut.

Jarak Teman	Ke C1	Ke C2
A (1,0,1)	0	1,73
B (1,1,0)	1,41	1
C (0,1,1)	1,41	1
D (0,1,0)	1,73	0
E (1,0,0)	1	1,41
F (0,0,1)	1	1,41

5. Tentukan setiap data masuk ke kluster mana dengan cara membandingkan nilai jarak ke C1 dan jarak ke C2.

Jarak Teman	Ke C1	Ke C2	Jarak terdekat (yang paling kecil antara C1 dan C2)
A (1,0,1)	0	1,73	masuk ke C1
B (1,1,0)	1,41	1	masuk ke C2
C (0,1,1)	1,41	1	masuk ke C2
D (0,1,0)	1,73	0	masuk ke C2
E (1,0,0)	1	1,41	masuk ke C1
F (0,0,1)	1	1,41	masuk ke C1

Kelompok Cluster 1 (C1) adalah A, E dan F.

Kelompok Cluster 2 (C2) adalah B, C dan D.

6. Hitung pusat (sentroid) baru untuk setiap klaster.

- Kelompok klaster 1 (C1) adalah A, E dan F.

Teman	Dimensi 1 = Olahraga	Dimensi 2 = Membaca	Dimensi 3 = Musik
A (1,0,1)	1	0	1
E (1,0,0)	1	0	1
F (0,0,1)	0	0	1
Jumlah	2	0	3

Untuk menghitung sentroid baru, hitung rata-rata setiap dimensinya.

Sentroid baru C1 = $(2/3, 0/3, 3/3) = (0.67, 0, 1)$

- Kelompok klaster 2 (C2) adalah B, C dan D.

Teman	Dimensi 1 = Olahraga	Dimensi 2 = Membaca	Dimensi 3 = Musik
B (1,1,0)	1	1	0
C (0,1,1)	0	1	1
D (0,1,0)	0	1	0
Jumlah	1	3	1

Untuk menghitung sentroid baru, hitung rata-rata setiap dimensinya.

Sentroid baru C2= (1/3 , 3/3 , 1/3) = (0.33, 1, 0.33)

7. Cek perubahan pusat klaster, jika pusat sentroid berubah, ulangi proses dari langkah 3 dengan menggunakan hasil sentroid baru C1 dan C2. Jika pusat tidak berubah, hentikan proses. Setelah dihitung jarak ke sentroid baru C1 dan sentroid baru C2, maka dihasilkan jarak sebagai berikut.

Jarak Teman	Ke Sentroid Baru C1	Ke Sentroid Baru C2	Jarak Terdekat (yang Paling Kecil antara C1 dan C2)
A (1,0,1)	0,47	1,37	masuk ke C1
B (1,1,0)	1,24	0,74	masuk ke C2
C (0,1,1)	1,24	0,74	masuk ke C2
D (0,1,0)	1,37	0,74	masuk ke C2
E (1,0,0)	0,74	1,24	masuk ke C1
F (0,0,1)	0,74	1,24	masuk ke C1

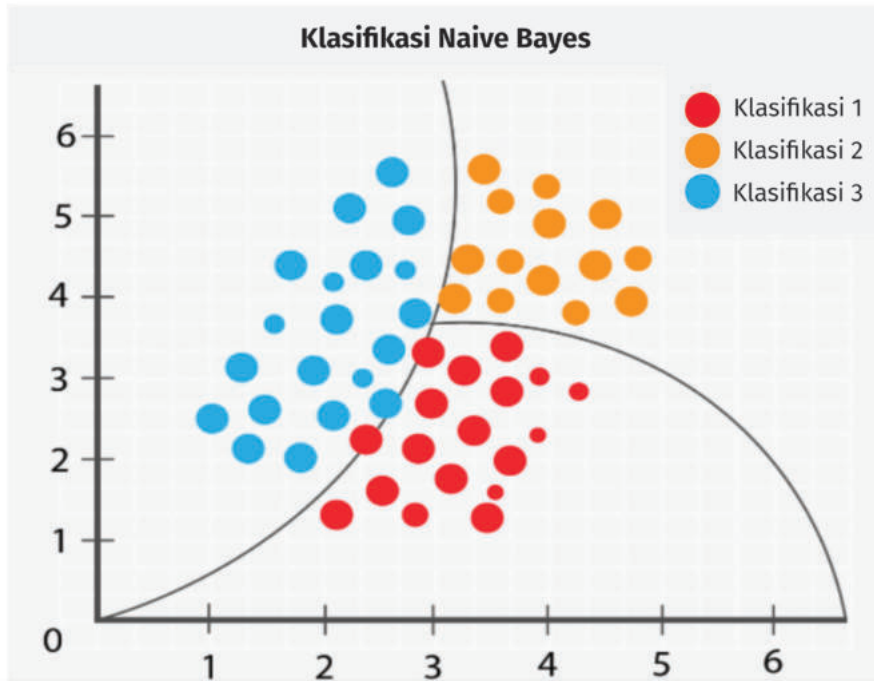
Kelompok klaster 1 (C1) adalah A, E dan F.

Kelompok klaster 2 (C2) adalah B, C dan D.

8. Selesai.

d. Naive Bayes

Naive Bayes adalah metode atau algoritma yang digunakan untuk melakukan prediksi atau klasifikasi. Algoritma ini didasarkan pada teorema Bayes, yaitu rumus matematika yang menghitung kemungkinan suatu kejadian terjadi (probabilitas) berdasarkan informasi yang sudah kita ketahui sebelumnya (*klasifier*/kelas).



Gambar 2.4 Klasifikasi Naive Bayes dengan 3 Kelas (Merah, Oranye, Biru)

Dalam contoh sebelumnya diketahui terdapat 3 kelas (merah, oranye, dan biru) sebagaimana gambar 2.4. Setiap titik baru akan diklasifikasikan ke kelas dengan nilai probabilitas paling besar. Untuk lebih jelasnya, perhatikan urutan langkah algoritma naive Bayes berikut.

- 1) Pengumpulan data.
- 2) Hitung probabilitas awal (*prior probability*).
- 3) Hitung probabilitas bersyarat (*conditional probability*).
- 4) Untuk data baru yang ingin diprediksi, gunakan rumus naive Bayes.
- 5) Kelas dengan probabilitas tertinggi adalah hasil prediksi akhir.
- 6) Selesai.

Contoh Klasifikasi Jenis Buah

1. Pengumpulan data, sebagai contoh data telah dikumpulkan sebagai berikut.

Buah	Ukuran	Rasa
Apel	Besar	Manis
Apel	Besar	Asam
Apel	Kecil	Manis
Jeruk	Besar	Asam
Jeruk	Besar	Asam
Jeruk	Kecil	Asam
Anggur	Kecil	Manis
Anggur	Kecil	Manis
Anggur	Besar	Manis

2. Hitung probabilitas awal (*prior probability*).
 - $P(\text{Apel}) = \text{JumlahApel} / \text{TotalData} = 3/9 = 0,33$
 - $P(\text{Jeruk}) = \text{JumlahJeruk} / \text{TotalData} = 3/9 = 0,33$
 - $P(\text{Anggur}) = \text{JumlahAnggur} / \text{TotalData} = 3/9 = 0,33$
3. Hitung probabilitas bersyarat (*conditional probability*).

$$P(A|B) = P(A \cap B) / P(B)$$

Keterangan:

$P(A|B)$ = Probabilitas bersyarat A terhadap B.

$P(A \cap B)$ = Probabilitas terjadinya irisan antara kejadian A dan kejadian B atau probabilitas kedua kejadian secara bersamaan

$P(B)$ = Probabilitas B

Buah	Ukuran		Rasa	
	Besar (A)	Kecil (A)	Manis (A)	Asam (A)
	$P(A B)$	$P(A B)$	$P(A B)$	$P(A B)$
Apel (B)	$P(A B) = \frac{2}{3} = 0.67$	$P(A B) = \frac{1}{3} = 0.33$	$P(A B) = \frac{2}{3} = 0.67$	$P(A B) = \frac{1}{3} = 0.33$
Jeruk (B)	$P(A B) = \frac{2}{3} = 0.67$	$P(A B) = \frac{1}{3} = 0.33$	$P(A B) = \frac{0}{3} = 0$	$P(A B) = \frac{3}{3} = 1$
Anggur (B)	$P(A B) = \frac{1}{3} = 0.33$	$P(A B) = \frac{2}{3} = 0.67$	$P(A B) = \frac{3}{3} = 1$	$P(A B) = \frac{0}{3} = 0$

4. Untuk data baru yang ingin diprediksi, gunakan rumus naive Bayes.

Sebagai contoh, terdapat buah baru yang memiliki ukuran = Kecil dan Rasa = Manis. Klasifikasikan termasuk ke dalam buah apa?

- Apel

$$P(A|B) = P(\text{databaru} | \text{Apel}) = P(\text{Apel}) * P(\text{Ukuran} | \text{Apel}) * P(\text{Rasa} | \text{Apel})$$

$$P(A|B) = P(\text{databaru} | \text{Apel}) = P(\text{Apel}) * P(\text{Kecil} | \text{Apel}) * P(\text{Manis} | \text{Apel})$$

Berdasarkan dari proses 2 dan 3 dapat ditentukan hasilnya sebagai berikut.

$$P(\text{databaru} | \text{Apel}) = 0,33 * 0,33 * 0,67 = 0,072$$

- Jeruk

$$P(A|B) = P(\text{databaru} | \text{Jeruk}) = P(\text{Jeruk}) * P(\text{Ukuran} | \text{Jeruk}) * P(\text{Rasa} | \text{Jeruk})$$

$$P(A|B) = P(\text{databaru} | \text{Jeruk}) = P(\text{Jeruk}) * P(\text{Kecil} | \text{Jeruk}) * P(\text{Manis} | \text{Jeruk})$$

Berdasarkan dari proses 2 dan 3 dapat ditentukan hasilnya sebagai berikut.

$$P(\text{databaru} | \text{Jeruk}) = 0,33 * 0,33 * 0 = 0$$

- Anggur

$$P(A|B) = P(\text{databaru} | \text{Anggur}) = P(\text{Anggur}) * P(\text{Ukuran} | \text{Anggur}) * P(\text{Rasa} | \text{Anggur})$$

$$P(A|B) = P(\text{databaru} | \text{Anggur}) = P(\text{Anggur}) * P(\text{Kecil} | \text{Anggur}) * P(\text{Manis} | \text{Anggur})$$

Berdasarkan dari proses 2 dan 3 dapat ditentukan hasilnya sebagai berikut.

$$P(\text{databaru} | \text{Anggur}) = 0,33 * 0,67 * 1 = 0,22$$

5. Kelas dengan probabilitas tertinggi adalah hasil prediksi akhir.

Berdasarkan hasil dari proses ke 4, maka untuk data buah baru dengan ukuran=kecil dan rasa=manis, hasil prediksinya masuk ke dalam kelas Anggur karena memiliki probabilitas paling tinggi yaitu 0,22

6. Selesai.

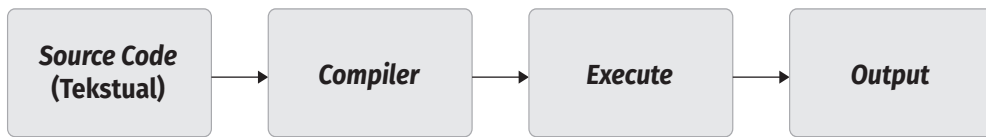
D. Pengenalan Pemrograman Teks

1. Pemrograman Teks

Pemrograman tekstual adalah pemrograman yang instruksi dan kode program ditulis secara manual dalam bentuk teks menggunakan sintaks dan aturan tertentu dari bahasa pemrograman yang digunakan. Penulisan kode program menggunakan *tool* editor teks, seperti notepad, notepad++, atau editor lainnya. Berbeda dengan antarmuka grafis (GUI) yang menggunakan ikon dan tombol, pemrograman tekstual menggunakan baris perintah (*command line*) sebagai media komunikasi antara pengguna dan komputer. Perintah-perintah dalam pemrograman tekstual dijalankan melalui terminal (pada sistem operasi Linux dan macOS) atau *Command Prompt* (pada Windows), yang memungkinkan pengguna memberikan instruksi langsung kepada sistem. Contoh bahasa pemrograman tekstual antara lain C, C++, Pascal, Python, Java, dan lain-lain.

Agar komputer dapat menjalankan program yang telah ditulis secara tekstual, diperlukan proses yang disebut *compiler*. *Compiler* adalah perangkat lunak khusus yang bertugas menerjemahkan kode sumber tersebut menjadi

bahasa mesin, yaitu bahasa yang dapat langsung dipahami dan dieksekusi oleh komputer. Ilustrasi proses pemrograman tekstual dapat dilihat di bawah ini.



a. **Source Code (Tekstual)**

Tahapan pertama dalam pemrograman tekstual adalah menulis kode menggunakan editor teks atau terminal. Pada tahap ini, kamu dapat menuliskan instruksi program dalam bahasa pemrograman tertentu seperti C, C++, Java, atau Python. Kode tersebut disimpan dalam fail teks dengan ekstensi sesuai bahasa yang digunakan. Kode ini masih dalam bentuk yang dapat dibaca manusia dan belum dapat dijalankan langsung oleh komputer.

b. **Compiler**

Setelah kode selesai ditulis, langkah berikutnya adalah proses *compiler*. *Compiler* bertugas menerjemahkan kode sumber (*source code*) menjadi kode mesin yang dapat dipahami oleh komputer. Selama proses ini, *compiler* akan memeriksa kesalahan sintaks dalam kode. Jika tidak terdapat kesalahan, maka akan terbentuk fail *executable*.

c. **Execute**

Tahap selanjutnya adalah menjalankan program dari fail *executable* yang telah dihasilkan. Ketika program dijalankan, komputer akan melaksanakan semua instruksi yang telah diterjemahkan sebelumnya oleh *compiler*. Pada tahap ini, program melakukan proses sesuai dengan kode yang dituliskan, seperti menghitung angka dan menerima input dari pengguna.

d. **Output**

Tahap ini adalah tahap terakhir untuk menampilkan hasil dari program yang dijalankan. *Output* biasanya ditampilkan pada layar atau terminal dalam bentuk teks, angka, atau tampilan lainnya, tergantung pada tujuan program tersebut. Sebagai contoh, jika kode berisi perintah untuk mencetak teks “Selamat Datang”, maka pesan tersebut akan muncul pada layar sebagai hasil akhir proses pemrograman.

Ciri Pemrograman Teksual:

- 1) Kode program diketik secara manual.
- 2) Memerlukan pemahaman mendalam mengenai sintaks dan logika bahasa pemrograman yang digunakan.
- 3) Lebih fleksibel dan dapat digunakan untuk membuat program yang kompleks.

Berbeda dengan pemrograman visual yang menggunakan elemen grafis atau blok-blok kode yang dapat disusun secara *drag-and-drop*, pemrograman tekstual menuntut seseorang untuk memahami dan menuliskan kode secara detail.

Perbedaan bahasa pemrograman tekstual dan visual dapat dilihat sebagai berikut.

Tabel 2.3 Perbedaan Bahasa Pemrograman Teksual dan Visual

Aspek	Pemrograman Teksual	Pemrograman Visual
Cara Membuat Program	Menulis kode secara manual dalam bentuk teks.	Menggunakan antarmuka grafis, menyusun blok/elemen.
Sintaks	Harus memahami cara penulisan sintaks dengan benar.	Tidak perlu menulis sintaks, cukup menyusun blok.
Kemudahan Belajar	membutuhkan pemahaman yang mendalam.	Lebih mudah dipahami, cocok untuk pemula.
Fleksibilitas	Sangat fleksibel, cocok untuk program kompleks.	Terbatas pada fitur yang disediakan visual editor.
Contoh Bahasa	C, C++, Python, Java, JavaScript, PHP	Scratch, Blockly, MIT App

2. Tool Pemrograman Teks

Terdapat banyak *tool* yang digunakan untuk mengimplementasikan pemrograman tekstual, baik dengan cara menginstal dengan penyesuaian bahasa pemrogramannya, ataupun menggunakan *tool online* yang tidak perlu penginstalan terlebih dahulu. Beberapa cara praktis untuk

mengimplementasikan pemrograman tekstual tanpa perlu instalasi adalah OnlineGDB dan Google Colab. Google Colab merupakan aplikasi berbasis Cloud milik Google yang mendukung penulisan dan eksekusi kode terutama bahasa pemrograman Python, tetapi Google Colab juga dapat digunakan untuk bahasa lain seperti C dan Java dengan sedikit konfigurasi. Sedangkan OnlineGDB, merupakan *compiler* dan *debugger online* yang mendukung berbagai bahasa pemrograman, sehingga memudahkan pengguna untuk menulis, menguji, dan melakukan *debugging* program tanpa perlu menginstal perangkat lunak tambahan. Proses menulis, mengompilasi, dan mengeksekusi program dapat dilakukan dalam satu antarmuka yang sederhana dan interaktif. *Tool* aplikasi ini mendukung berbagai bahasa pemrograman seperti C, C++, Java, dan Python. *Output* dari program ditampilkan langsung untuk memudahkan pengguna melihat hasil eksekusi secara instan tanpa harus menggunakan *compiler* lagi.



Untuk mengenal fitur Google Colab dan OnlineGDB, kamu dapat mempelajarinya pada QR Code berikut.

<https://buku.kemdikbud.go.id/s/kka10ay>



E. Sintaks Dasar Pemrograman

Pemrograman adalah proses menyusun instruksi untuk dijalankan komputer. Setiap bahasa pemrograman memiliki aturan sintaks (tata bahasa) masing-masing yang harus dipahami oleh pemrogram. Pada subbab ini, materi yang akan dibahas adalah sintaks dasar dari tiga bahasa pemrograman populer, yaitu C, Python, dan Java.

1. Sintaks Pemrograman C

Bahasa C adalah bahasa pemrograman prosedural yang banyak digunakan untuk pengembangan sistem dan aplikasi tingkat rendah. Sintaks dasar pemrograman C adalah sebagai berikut.

```
#include <stdio.h>

int main() {
    printf("Halo, dunia!\n");
    return 0;
}
```

Tabel 2.4 Kode dan Keterangan Sintaks Dasar Pemrograman C

Kode	Keterangan
<code>#include <stdio.h></code>	Berfungsi untuk menyertakan pustaka <i>input-output</i> standar.
<code>int main() { ... }</code>	Merupakan fungsi utama tempat program dieksekusi.
<code>printf(...)</code>	Merupakan fungsi untuk menampilkan <i>output</i> pada layar.
<code>return 0</code>	Mengembalikan nilai 0 sebagai tanda program selesai.

Beberapa aturan dalam pemrograman C.

- Penulisan kode logika pemrograman harus berada pada fungsi *main()*.
- Pemberian komentar dapat menggunakan kode `//` untuk komentar satu baris, dan `/* ... */` untuk komentar banyak baris. Setiap komentar tidak akan dijalankan oleh program.
- Setiap pernyataan diakhiri dengan tanda titik koma (;).
- Dalam penulisan variabel tidak boleh menggunakan spasi, tidak boleh diawali dengan karakter/angka, jika akan menggunakan variabel dua kata gunakan tanda *underscore* (`_`).
- Bersifat *case-sensitif*, huruf kapital dan huruf kecil berpengaruh.

2. Pemrograman Python

Python adalah bahasa pemrograman tingkat tinggi yang terkenal karena sintaksnya yang sederhana dan mudah dibaca. Sintaks dasar pemrograman Python adalah sebagai berikut.

```
printf("Halo, dunia!\n");
```

Pada pemrograman Python, tidak perlu fungsi `main()`, Setiap kode program langsung dituliskan di editor. Beberapa kode standar yang terdapat di Python adalah sebagai berikut.

Tabel 2.5 Kode, Keterangan, dan Contoh Sintaks Dasar Pemrograman Python.

Kode	Keterangan	Contoh
<code>print(...)</code>	Untuk menampilkan <i>output</i> pada layar.	<code>print("Selamat Datang")</code>
<code>input()</code>	untuk menerima input dari pengguna.	<code>nama = input("Masukkan nama: ")</code> <code>print("Halo,", nama)</code>
<code>int(), float(), str()</code>	Untuk konversi tipe data: Int : Mengubah <i>string</i> menjadi integer. Float: Mengubah <i>string</i> menjadi float. Str : Mengubah angka menjadi <i>string</i> .	<code>a = "123"</code> <code>print(int(a))</code> <code>print(float(a))</code> <code>print(str(123))</code>

Berikut beberapa aturan dalam pemrograman Python.

- Penulisan kode logika pemrograman langsung tidak membutuhkan fungsi `main()`.
- Pemberian komentar dapat menggunakan kode `#`. Setiap komentar tidak akan dijalankan oleh program.
- Setiap pernyataan tidak wajib diakhiri dengan tanda titik koma (;).

- d. Dalam penulisan variabel tidak boleh menggunakan spasi, tidak boleh diawali dengan karakter/angka, jika akan menggunakan variabel dua kata gunakan tanda *underscore* (_).
- e. Bersifat *case-sensitif*, huruf kapital dan huruf kecil berpengaruh.

3. Pemrograman Java

Java adalah bahasa pemrograman berorientasi objek yang dirancang agar kode program yang ditulis dapat dijalankan di berbagai *platform* sistem operasi. Kode bahasa pemrograman Java bersifat *case-sensitive* yang membedakan huruf kapital dan huruf kecil. Contoh “NAMA” dengan “nama” merupakan hal yang berbeda, meskipun cara bacanya sama. Sintaks dasar pemrograman Java adalah sebagai berikut.

```
public class Halo {
    public static void main(String[] args) {
        System.out.println("Halo, dunia!");
    }
}
```

Tabel 2.6 Kode dan Keterangan Sintaks Dasar Pemrograman Java

Kode	Keterangan
<code>public class Halo</code>	Mendeklarasikan kelas bernama <code>Halo</code> . Nama kelas harus sama dengan nama fail, misalnya nama failnya <code>Halo.java</code> .
<code>public static void main(String[] args)</code>	Program utama, yang merupakan titik awal untuk eksekusi program. Tanpa baris ini, program Java tidak akan tahu awalan untuk mulai menjalankan kode.
<code>System.out.println(...)</code>	Untuk menampilkan <i>output</i> pada layar.

Berikut beberapa aturan dalam pemrograman Java.

- Semua kode harus ditulis di dalam kelas.
- Nama fail harus sama dengan nama kelas.
- Penulisan kode logika pemrograman membutuhkan fungsi *main()*.
- Pemberian komentar dapat menggunakan kode `//` untuk satu baris, dan `/* ... */` untuk komentar banyak baris. Setiap komentar tidak akan dijalankan oleh program.
- Setiap pernyataan tidak wajib diakhiri dengan tanda titik koma (;).
- Dalam penulisan variabel tidak boleh menggunakan spasi, tidak boleh diawali dengan karakter/angka, jika akan menggunakan variabel dua kata gunakan tanda *underscore* (_).
- Bersifat *case-sensitif*, huruf kapital dan huruf kecil berpengaruh.

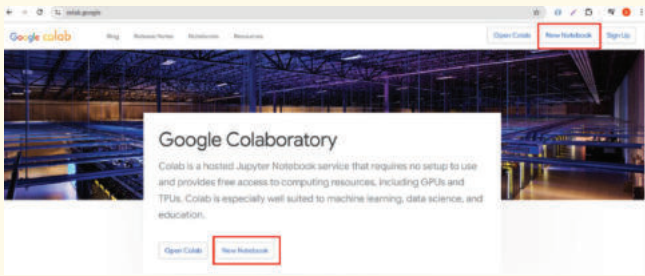


Ayo Mencoba!



Jenis Aktivitas : Mandiri

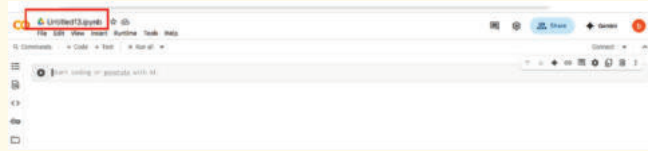
No Aktivitas : AP-K10-09

Aktivitas	Keterangan Aktivitas
Program untuk mengetahui sebuah bidang persegi panjang atau sama sisi.	kamu akan membuat program untuk mengenali sebuah bidang termasuk persegi panjang atau sama sisi. Langkah-Langkah Kegiatan: Buka aplikasi https://colab.google/. 

Aktivitas

Keterangan Aktivitas

2. Pilih *New Notebook*!



3. Ubah nama fail **Untitled13.ipynb** menjadi **latihanprogram1.ipynb**

4. Tulis kode program pada *code cell*.

▶ start coding or generate with AI.

5. Buat masukan panjang dan lebar.

```
▶ panjang = int(input("Panjang: "))  
lebar = int(input("Lebar: "))
```

Dalam penulisan variabel pada Python tidak perlu dibuat deklarasi terlebih dahulu, tuliskan secara langsung seperti di atas, yaitu panjang dan lebar karena menggunakan `int()`, maka variabel tersebut bernilai integer.

6. Buat kondisi pengecekan, apakah nilai panjang sama dengan nilai lebar? Jika sama, maka tampilkan “sama sisi”, jika tidak maka tampilkan “persegi panjang”.

```
if panjang == lebar:  
    print("Sama sisi")  
else:  
    print("Persegi panjang")
```

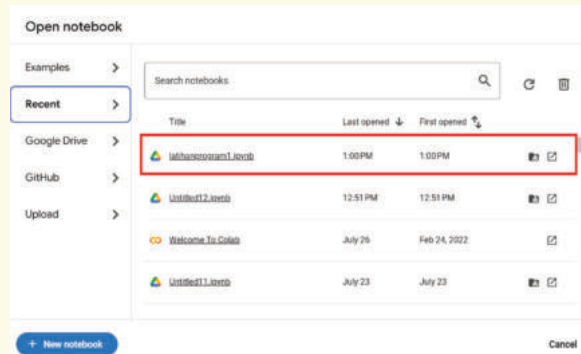
Aktivitas	Keterangan Aktivitas
	7. Kode program lengkapnya akan terlihat seperti berikut. <pre data-bbox="423 399 1068 675"> ▶ panjang = int(input("Panjang: ")) lebar = int(input("Lebar: ")) if panjang == lebar: print("Sama sisi") else: print("Persegi panjang") </pre> 8. Jalankan program dengan cara klik tombol <i>Run Cell</i>. ▶ 9. Masukkan nilai panjang, contoh: 10, lalu tekan <i>enter</i>. <pre data-bbox="423 913 1068 980"> ... Panjang: </pre> 10. Masukkan nilai lebar, contoh: 10, lalu tekan <i>enter</i>. <pre data-bbox="423 1056 1068 1161"> ... Panjang: 10 Lebar: </pre> 11. Hasil akhir akan terlihat seperti berikut. <pre data-bbox="517 1228 833 1351"> ➞ Panjang: 10 Lebar: 10 Sama sisi </pre>
Menampilkan umur dan tahun lahir.	kamu akan membuat program menghitung dan mencetak tahun lahir (asumsi tahun sekarang 2025).

Aktivitas

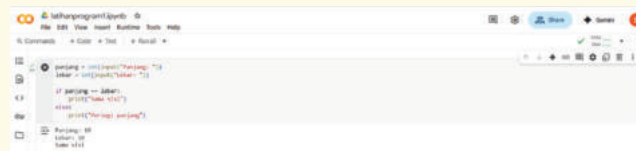
Keterangan Aktivitas

Langkah-Langkah Kegiatan:

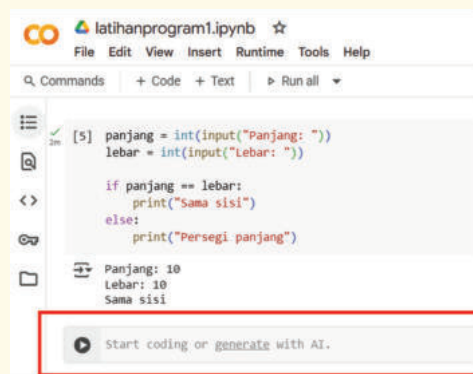
1. Gunakan fail **latihanprogram1.ipynb**.
2. Jika Google Colab sudah ditutup, buka kembali dengan cara membuka aplikasi <https://colab.google/>
3. Pilih *Open Colab*.



4. Klik fail **latihanprogram1.ipynb**.



5. Buat *code cell* baru dengan cara klik tombol + **Code**, maka *code cell* akan terbentuk di bawahnya seperti berikut.



Aktivitas

Keterangan Aktivitas

6. Buat masukan nama dan umur.

```
nama = input("Nama : ")
umur = int(input("umur: "))
```

7. Buat proses perhitungan tahun lahir dengan rumus
tahunlahir = 2025 - umur.

```
nama = input("Nama : ")
umur = int(input("umur: "))

tahunlahir= 2025 - umur
```

8. Menampilkan hasil dengan tampilan
“Halo, <nama>, kamu lahir tahun <tahunlahir>.”

<nama> berisi nama yang diinputkan,

<tahunlahir> berisi hasil tahun lahir berdasarkan
dari perhitungan.

```
print("Halo, "+nama+", kamu lahir tahun "+str(tahunlahir)+".")
```

Dapat juga menggunakan format berikut.

```
print("Halo, {}, kamu lahir tahun {}".format(nama, tahunlahir))
```

9. Code lengkapnya seperti berikut.

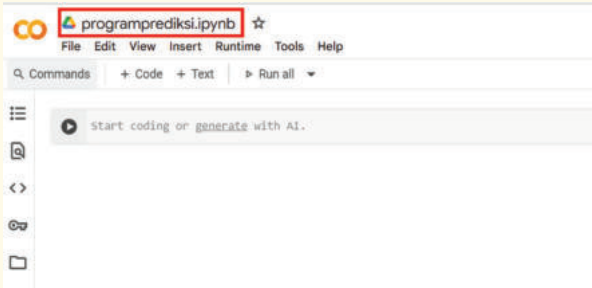
```
nama = input("Nama : ")
umur = int(input("umur: "))

tahunlahir= 2025 - umur

print("Halo, "+nama+", kamu lahir tahun "+str(tahunlahir)+".")
```

10. Jalankan program dengan cara klik tombol *run cell*.



Aktivitas	Keterangan Aktivitas
Prediksi berat badan dengan Linier Regresi.	kamu akan membuat program prediksi berat badan dengan menggunakan algoritma regresi linear. Langkah-Langkah Kegiatan: 1. Buat fail baru di Google Colab dengan nama programprediksi.ipynb.  2. Tulis <i>import library</i> pada <i>code cell</i>.  3. Jalankan <i>code cell</i>, jika terdapat tanda centang hijau artinya kode program berjalan dengan baik atau tidak ada eror.  4. Buat <i>code cell</i> baru dengan pilih + code. 

Aktivitas

Keterangan Aktivitas

5. Tulis kode program untuk memasukkan data pada *code cell* yang baru dibuat.

```
# Contoh Data
tinggi = np.array([150, 155, 160, 165, 170, 175, 180]).reshape(-1, 1)
berat = np.array([50, 53, 56, 60, 65, 68, 72])
```

6. Jalankan *code cell*.
7. Buat *code cell* baru dengan pilih + **code**.
8. Tulis kode program model linier regresi pada *code cell* yang baru dibuat.

```
model = LinearRegression()
model.fit(tinggi, berat)
```

9. Jalankan *code cell*.

```
model = LinearRegression()
model.fit(tinggi, berat)
```

LinearRegression

LinearRegression()

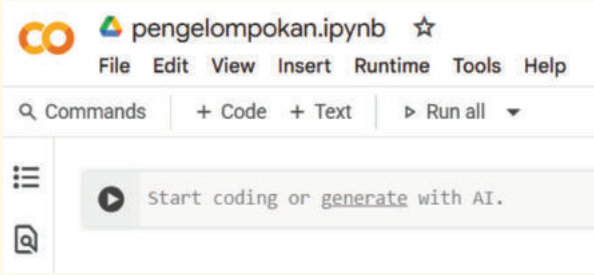
10. Buat *code cell* baru dengan pilih + **code**.
11. Tulis kode program untuk menampilkan nilai koefisien regresi (slope) dan *intercept*.

```
print("Koefisien (m):", model.coef_[0])
print("Intercept (b):", model.intercept_)
```

12. Jalankan *code cell*.

```
print("Koefisien (m):", model.coef_[0])
print("Intercept (b):", model.intercept_)
```

Koefisien (m): 0.7500000000000001
Intercept (b): -63.178571428571445

Aktivitas	Keterangan Aktivitas
	13. Buat <i>code cell</i> baru dengan pilih + code. 14. Tulis kode program untuk melakukan uji coba prediksi. Sebagai contoh diketahui tinggi badan 172, berapa prediksi berat badannya? <pre data-bbox="458 464 1009 611"> tinggi_uji = np.array([[172]]) prediksi = model.predict(tinggi_uji) print(prediksi) </pre> 15. Jalankan <i>code cell</i>, untuk mengetahui hasil prediksi berat badan jika diketahui tinggi badan 172 cm. <pre data-bbox="458 734 1016 929"> tinggi_uji = np.array([[172]]) prediksi = model.predict(tinggi_uji) print(prediksi) </pre> [65.82142857]
Penge- lompokan menggu- nakan K-Means.	Kamu diminta untuk membuat pengelompokan berdasarkan hobi dengan menggunakan algoritma K-Means. Langkah-Langkah Kegiatan: 1. Buat fail baru di Google Colab dengan nama pengelompokan.ipynb. 

Aktivitas

Keterangan Aktivitas

2. Tulis *import library* pada *code cell*.

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

3. Jalankan *code cell*, jika terdapat tanda centang hijau artinya kode program berjalan dengan baik atau tidak ada error.

```
✓ 4s ▶ import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

4. Buat *code cell* baru dengan pilih + **code**.

```
✓ 4s ▶ import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

▶ Start coding or generate with AI.
```

5. Tulis kode program untuk memasukkan data pada *code cell* yang baru dibuat.

```
▶ data = {
    'Teman': ['A', 'B', 'C', 'D', 'E', 'F'],
    'Olahraga': [1, 1, 0, 0, 1, 0],
    'Membaca': [0, 1, 1, 1, 0, 0],
    'Musik': [1, 0, 1, 0, 0, 1]
}

df = pd.DataFrame(data)

x = df[['Olahraga', 'Membaca', 'Musik']]

print(X)
```

6. Jalankan *code cell*.

```
data = {
    'Teman': ['A', 'B', 'C', 'D', 'E', 'F'],
    'Olahraga': [1, 1, 0, 0, 1, 0],
    'Membaca': [0, 1, 1, 1, 0, 0],
    'Musik': [1, 0, 1, 0, 0, 1]
}

df = pd.DataFrame(data)

X = df[['Olahraga', 'Membaca', 'Musik']]

print(X)
```

	Olahraga	Membaca	Musik
0	1	0	1
1	1	1	0
2	0	1	1
3	0	1	0
4	1	0	0
5	0	0	1

7. Buat *code cell* baru dengan pilih + **code**.

8. Tulis kode program untuk menentukan sentroid awal pada *code cell* yang baru dibuat.

```
initial_centroids = X.loc[[0, 3]]
```

0 artinya indeks ke-0 pada *array* Teman atau disebut juga A.

3 artinya indeks ke-3 pada *array* Teman atau disebut juga D.

9. Tulis kode program model K-Means pada *code cell* yang baru dibuat.

```
# Model K-Means dengan 2 cluster

kmeans = KMeans(n_clusters=2, init=initial_centroids, n_init=1)
df['cluster'] = kmeans.fit_predict(X)

print(df)
```

10. Jalankan *code cell*.

```

# Model K-Means dengan 2 cluster

kmeans = KMeans(n_clusters=2, init=initial_centroids, n_init=1)
df['cluster'] = kmeans.fit_predict(X)

print(df)

```

	Teman	Olahraga	Membaca	Musik	Cluster
0	A	1	0	1	0
1	B	1	1	0	1
2	C	0	1	1	1
3	D	0	1	0	1
4	E	1	0	0	0
5	F	0	0	1	0

11. Buat *code cell* baru dengan pilih + **code**.

12. Tulis kode program untuk menampilkan hasil kelompok.

```

cluster_0 = df[df['cluster'] == 0]
cluster_1 = df[df['cluster'] == 1]

print("Anggota Cluster 0:")
print(cluster_0)
print("\n")
print("Anggota Cluster 1:")
print(cluster_1)

```

13. Jalankan *code cell*.

```

cluster_0 = df[df['cluster'] == 0]
cluster_1 = df[df['cluster'] == 1]

print("Anggota Cluster 0:")
print(cluster_0)
print("\n")
print("Anggota Cluster 1:")
print(cluster_1)

```

Anggota Cluster 0:					
	Teman	Olahraga	Membaca	Musik	Cluster
0	A	1	0	1	0
4	E	1	0	0	0
5	F	0	0	1	0

Anggota Cluster 1:					
	Teman	Olahraga	Membaca	Musik	Cluster
1	B	1	1	0	1
2	C	0	1	1	1
3	D	0	1	0	1

14. Selesai.

F. Analisis Kesalahan (*Debugging*) dalam Kode Program

Debugging adalah proses mencari, menemukan, dan memperbaiki kesalahan dalam program agar program dapat berjalan sebagaimana mestinya. Kesalahan dalam kode dapat muncul karena berbagai faktor, seperti kesalahan logika, penulisan sintaks yang salah, atau kesalahan konfigurasi. Proses *debugging* ini dapat membantu pengembang memahami kesalahan, alasan terjadinya kesalahan, dan cara memperbaikinya. Kesalahan dalam kode (*bug*) adalah hal yang sangat umum dan tak terhindarkan. Bahkan programmer berpengalaman sekalipun pasti pernah mengalami kesalahan saat menulis kode. Oleh karena itu, kemampuan untuk menganalisis dan memperbaiki kesalahan (*debugging*) menjadi keterampilan penting yang harus dipahami.

Jenis-Jenis Kesalahan dalam Kode Program

1. Kesalahan Sintaks (*Syntax Error*)

Kesalahan sintaks terjadi ketika kode program tidak sesuai dengan aturan penulisan (tata bahasa) yang ditentukan oleh bahasa pemrograman. *Compiler* atau interpreter akan langsung menolak menjalankan program sebelum semua kesalahan sintaks diperbaiki.

Contoh Sintaks Java

```
System.out.println("Halo dunia")
```

Error terjadi karena tidak ada tanda titik koma (;) di akhir pernyataan.

Contoh Sintaks Python

```
pint("Halo dunia")
```

Error terjadi karena kesalahan penulisan `pint` harusnya `print`.

Contoh Sintaks C

```
print("Halo dunia")
```

Error terjadi karena pada C tidak ada fungsi `print` selayaknya python, yang ada adalah `printf`.

2. Kesalahan Logika (*Logic Error*)

Kesalahan logika terjadi ketika program berjalan tanpa error, tetapi hasil yang dihasilkan tidak sesuai dengan yang diharapkan. Kesalahan ini tidak akan terdeteksi oleh *compiler*/interpreter.

Contoh kesalahan logika “Program Menghitung Penjumlahan” dalam bahasa C.

```
#include <stdio.h>

int main()
{
    int a, b, hasil;

    a = 15;
    b = 5;

    hasil = a - b;

    printf("%d", hasil);

    return 0;
}
```

Jika kode program dijalankan akan berhasil, tidak terjadi error. Akan tetapi hasilnya akan salah. Hasil yang diharapkan adalah penjumlahan, tapi hasil menunjukkan pengurangan. Untuk mengetahui *logic error* ini perhatikan pada proses perhitungannya.

3. Kesalahan *Runtime* (*Runtime Error*)

Kesalahan *runtime* terjadi ketika program dijalankan.

Contoh kesalahan *runtime* dalam bahasa pemrograman C.

```
#include <stdio.h>

int main() {
    int a = 10;
    int b = 0;
    int hasil = a / b; // Runtime error: Division by
    zero

    printf("Hasil: %d\n", hasil);
    return 0;
}
```

Bahasa pemrograman di atas ketika dijalankan akan terjadi eror, karena proses pembagian $hasil = a/b$, karena pembagiannya dengan angka 0, maka hasilnya akan *Division by zero* yang menyebabkan *runtime error*.

Contoh kesalahan *runtime* dalam bahasa pemrograman Python

```
a = "10"
b = 5
print(a + b) # Runtime error: TypeError (tidak
bisa menjumlahkan str + int)
```

Bahasa pemrograman di atas ketika dijalankan akan terjadi eror, karena proses penjumlahan antara *string* dan angka yang menyebabkan *runtime error*.

Berikut beberapa langkah dalam proses *debugging*.

- Membaca pesan error. Pesan error dari interpreter/compiler biasanya menunjukkan letak dan jenis kesalahan pada *line* dan baris.
- Menambahkan *output* sementara (*print debugging*), untuk melihat hasilnya.
- Menggunakan *debugger tools* dengan menambahkan proses *breakpoint* pada baris tertentu.
- Memeriksa kode dengan saksama.

Contoh membaca error kode program pada bahasa Python.

```
▶ nilai = input("Masukkan nilai ujian: ")
if nilai >= 75:
    print("Lulus")
else:
    print("Tidak lulus")
```

Jika dijalankan program akan menampilkan error berikut.

```
↳ Masukkan nilai ujian: 20

-----
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1421913686.py in <cell line: 0>()
      1 nilai = input("Masukkan nilai ujian: ")
----> 2 if nilai >= 75:
      3     print("Lulus")
      4 else:
      5     print("Tidak lulus")

TypeError: '>=' not supported between instances of 'str' and 'int'
```

Next steps: [Explain error](#)

Perhatikan pesan error, terdapat kode program yang ditunjukan error.

```
----> 2 if nilai >= 75:
```

Dan perhatikan pesan `TypeError`.

```
TypeError: '>=' not supported between instances of 'str' and 'int'
```

Pesan error tersebut berarti, bahwa proses yang didalam kondisi IF itu tidak boleh *string*, tetapi nilai yang dibandingkan harus bernilai integer. Letak kesalahannya berarti nilai harus bertipe integer. Perhatikan kode program pada baris pertama `nilai = input("Masukkan nilai ujian: ")`, artinya variabel nilai akan selalu bernilai string karena tidak ada koneversi ke integer, Ubah baris pertama menjadi `nilai = int(input("Masukkan nilai ujian: "))`

Contoh membaca error kode program pada bahasa C.

```
1 |
2 | #include <stdio.h>
3 |
4 | int main()
5 | {
6 |     printf("Hello World")
7 |
8 |     return 0;
9 | }
10 |
```

Compilation failed due to following error(s).

```
main.c: In function 'main':
main.c:6:26: error: expected ';' before 'return'
6 |     printf("Hello World")
  |                        ^
7 |
8 |     return 0;
```

Ditunjukkan bahwa erornya tidak ada tanda titik koma

Ditunjukkan pada baris kode ke 6, pada posisi setelah `Printf("Hello World")`



Ayo Menganalisis!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-10

1. Di bawah ini adalah potongan kode program yang memiliki kesalahan sintaks atau logika.

```
usia = int(input("Masukkan usia anda: "))  
print(f"Usia anda adalah {} tahun",usia)
```

2. Baca kode program dan temukan kesalahan kode program tersebut. Kamu dapat melakukan hal berikut.
 - Menyalin kode ke Google Colab.
 - Menjalankan program.
 - Mencatat pesan kesalahan (*error message*).
 - Menjelaskan bagian yang menyebabkan eror.
3. Perbaiki kode program dan jalankan kembali hingga tidak terdapat eror.
4. Tuliskan dan jelaskan letak kesalahan yang kamu temukan di depan kelas.



Ayo Mengidentifikasi!



Jenis Aktivitas : Berpasangan

No Aktivitas : AP-K10-11

1. Pilih salah satu teman untuk melakukan aktivitas ini yang berfungsi sebagai *debugging* berpasangan.
2. Setiap pasangan mendapat soal yang dibagikan secara acak oleh guru.
3. Buat program sesuai dengan soal dengan 2 kesalahan kecil.

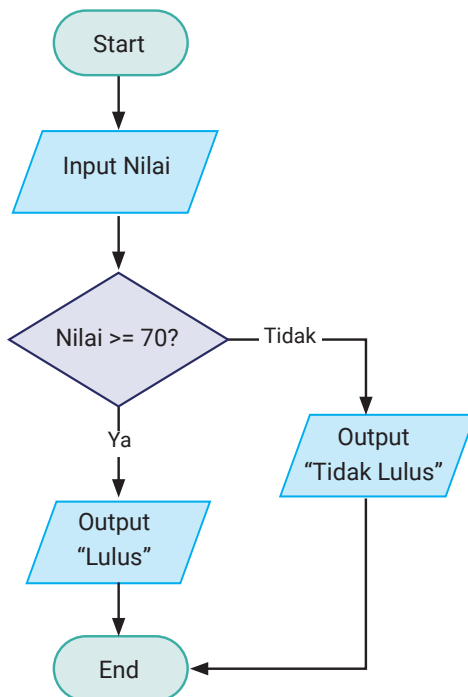
4. Periksa kode program yang dibuat oleh kelompok yang lain.
5. Catat kesalahan yang kalian temukan pada program dan berikan penjelasannya.
6. Presentasikan hasil temuan kesalahan kalian.



Uji Kompetensi

A. Pilihlah salah satu jawaban yang paling tepat!

1. Perhatikan *flowchart* berikut ini.



Berdasarkan gambar di atas, jika nilai yang dimasukkan adalah 85, maka *output* yang dihasilkan adalah

- a. eror
 - b. tidak lulus
 - c. lulus
 - d. gagal
 - e. tidak menghasilkan *output*
2. Simbol persegi panjang dalam *flowchart* berfungsi untuk...
- a. menandakan dimulainya program
 - b. input/output
 - c. proses
 - d. pengecekan
 - e. menandakan eror
3. Sintaks dalam bahasa pemrograman adalah
- a. cara menyimpan data di dalam fail
 - b. kumpulan aturan penulisan kode program agar dapat dimengerti oleh komputer
 - c. aturan penamaan variabel yang sama
 - d. proses mengubah kode program menjadi grafik
 - e. cara membaca bahasa pemrograman
4. Contoh penulisan struktur IF yang benar dalam Python adalah
- a. if x > 10 then:
 - b. if (x > 10)
 - c. if x > 10 {
 - d. if x > 10:
 - e. if x < 10;
5. Perintah `print("Loop")` dalam potongan kode Python berikut dijalankan sebanyak

```
for i in range(3):  
    for j in range(2):  
        print("Loop")
```

- a. 6 kali
- b. 3 kali
- c. 2 kali
- d. 5 kali
- e. 4 kali

B. Kerjakan soal-soal berikut!

1. Sebuah toko memberikan diskon 10% untuk pembelian di atas Rp100.000. Buatlah *flowchart* yang menunjukkan proses perhitungan total bayar setelah diskon.

2.

Wati memiliki kebiasaan membuat jadwal belajar harian sebelum tidur. Wati ingin menyusun urutan kegiatan belajarnya secara efisien agar dapat menyelesaikan semua tugas sekolah dan masih memiliki waktu istirahat yang cukup. Untuk itu, Wati menentukan beberapa langkah berikut.

- a. Mengecek tugas atau PR yang harus dikumpulkan esok hari.
- b. Mengurutkan tugas berdasarkan prioritas (batas waktu pengumpulan terdekat).
- c. Menentukan lama waktu pengerjaan untuk setiap tugas.
- d. Menyusun jadwal belajar dengan memperhitungkan waktu istirahat.
- e. Mengeksekusi jadwal tersebut.

Kerjakan soal berikut berdasarkan cerita di atas.

- a. Bagaimana Wati menerapkan konsep algoritma dalam kehidupan sehari-harinya? Jelaskan!
- b. Jelaskan alasan pentingnya menyusun langkah-langkah tersebut secara runtut!

3. Perhatikan potongan kode berikut ini!

Kode Python	Kode C
<pre>a = 5 b = 3 hasil = a + b print("Hasil:", hasil)</pre>	<pre>#include <stdio.h> int main() { int a = 5, b = 3; int hasil; hasil = a + b; printf("Hasil: %d", hasil); return 0; }</pre>

Berdasarkan kode tersebut, tentukan yang disebut kode untuk input, proses, dan *output*.

4. Perhatikan potongan kode berikut ini.

```
#include <stdio.h>

int main() {
    int umur;
    printf("Masukkan umur Anda: ");
    scanf("%d", umur);
    printf("Umur Anda adalah: %d\n", umur);
    return 0;
}
```

a. Jelaskan kesalahan yang terdapat pada kode program tersebut.

b. Jelaskan alasan kesalahan tersebut dapat terjadi!

5. Apa yang dimaksud dengan *debugging*? Bagaimana peran *debugging* dapat membantu dalam proses pembuatan program?



Pengayaan

Dalam dunia pemrograman, pemahaman terhadap algoritma merupakan langkah awal yang sangat penting. Algoritma adalah serangkaian instruksi logis dan sistematis yang digunakan untuk menyelesaikan suatu masalah. Algoritma yang baik mampu mengubah masalah kompleks menjadi langkah-langkah yang sederhana dan terstruktur. Kemampuan ini sangat dibutuhkan dalam merancang solusi, baik dalam dunia bisnis, kesehatan, pendidikan, maupun bidang lainnya.

Untuk memvisualisasikan algoritma, kamu dapat menggunakan *flowchart*, yaitu diagram alir yang menggambarkan alur logika penyelesaian masalah menggunakan simbol-simbol standar. *Flowchart* membantu seorang programmer dalam memahami cara sebuah sistem bekerja sebelum diubah menjadi kode program. Dengan bantuan *flowchart*, kesalahan logika dapat dideteksi lebih awal sebelum diubah menjadi bahasa pemrograman.

Berikut contoh penerapan dalam bahasa pemrograman Python.

Seorang petugas administrasi di sebuah klinik kesehatan akan membuat program sederhana untuk membantu mengelompokkan pasien berdasarkan kategori usia. Petugas hanya perlu memasukkan usia pasien, program akan otomatis menampilkan kategori usia yang sesuai.

Tabel 2.7 Tampilan Contoh Hasil Bahasa Pemrograman Python

Usia (tahun)	Kategori
0 – 1	Bayi
2 – 5	Balita
6 – 12	Anak-Anak
13 – 17	Remaja
18 – 59	Dewasa
di atas 60	Lansia

Untuk menyelesaikan program tersebut, berikut beberapa langkah yang harus dilakukan.

1. Membuka program aplikasi <https://www.onlinegdb.com/>.
2. Pilih bahasa pemrograman Python.
3. Hapus kode *default* yang ada di editor agar siap untuk mengetik program sendiri.
4. Menuliskan kode program seperti berikut.

```
1  # Program Kategori Usia Pasien
2
3  # Input usia dari pengguna
4  usia = int(input("Masukkan usia pasien: "))
5
6  # Logika pengelompokan usia
7  if usia >= 0 and usia <= 1:
8      print("Kategori: Bayi")
9  elif usia <= 5:
10     print("Kategori: Balita")
11  elif usia <= 12:
12     print("Kategori: Anak-anak")
13  elif usia <= 17:
14     print("Kategori: Remaja")
15  elif usia <= 59:
16     print("Kategori: Dewasa")
17  elif usia >= 60:
18     print("Kategori: Lansia")
19  else:
20     print("Usia tidak valid.")
21
```

5. Jalankan program dengan klik *Run*  **Run** .

Setelah kamu mengerti bagaimana logika percabangan bekerja dalam program Python untuk mengelompokkan data seperti kategori usia, kini saatnya kamu menerapkannya dalam konteks yang lebih interaktif dan dekat dengan kehidupan sehari-hari, yaitu dengan membuat *chatbot* sederhana. *Chatbot* ini akan merespons pertanyaan-pertanyaan pengguna berdasarkan kata kunci tertentu, dan dirancang menggunakan logika dasar *if-then* yang telah kamu pelajari sebelumnya.



Ayo Mencoba!



Jenis Aktivitas : Mandiri

No Aktivitas : AP-K10-12

Membuat *Chatbot* Sederhana Asisten Sekolah

Kamu akan membuat *chatbot* sederhana untuk membantu teman-teman menjawab beberapa pertanyaan umum seputar aktivitas sekolah.

- Jadwal Pelajaran
- Guru Pengampu
- Waktu Istirahat
- Ucapan Salam

Chatbot ini tidak menggunakan KA canggih, tapi hanya membalas berdasarkan kata kunci yang dikenali.

Langkah-Langkah Kegiatan:

1. Membuka program aplikasi <https://www.onlinegdb.com/>
2. Pilih bahasa pemrograman Python.
3. Hapus kode *default* yang ada pada editor agar siap untuk mengetik program sendiri.
4. Menulis program dasar *chatbot*.

```
1 print("Halo! Saya Bot Asisten Sekolah.")
2 print("Ketik pertanyaanmu, atau ketik 'keluar' untuk mengakhiri.")
3
4 while True:
5     pertanyaan = input("Kamu: ").lower()
6
7     if "jadwal" in pertanyaan:
8         print("Bot: Jadwal pelajaran hari ini bisa dilihat di papan pengumuman.")
9     elif "guru" in pertanyaan:
10        print("Bot: Guru pengampu Matematika adalah Ibu Rina.")
11    elif "istirahat" in pertanyaan:
12        print("Bot: Waktu istirahat dimulai pukul 10.15 selama 15 menit.")
13    elif "hai" in pertanyaan or "halo" in pertanyaan:
14        print("Bot: Hai juga! Ada yang bisa saya bantu?")
15    elif "keluar" in pertanyaan:
16        print("Bot: Terima kasih, sampai jumpa!")
17        break
18    else:
19        print("Bot: Maaf, saya belum mengerti pertanyaan itu.")
```

5. Uji coba *chatbot* dengan memberikan pertanyaan berikut.

- a. Siapa nama gurumu?
- b. Halo
- c. Kapan waktu istirahat?
- d. Keluar

Perhatikan bagaimana *chatbot* merespons berdasarkan kata kunci.

6. Kamu juga dapat menambahkan fitur untuk menjawab pertanyaan mengenai nama kepala sekolah, lokasi kelas, atau waktu pulang sekolah.



Refleksi

Setelah mempelajari bab ini, luangkan waktu sejenak untuk merenungkan perjalanan belajarmu. Jawablah pertanyaan berikut dengan jujur sesuai dengan perasaanmu setelah mempelajari bab ini. Tidak ada jawaban benar atau salah, poin penting di sini adalah kejujuran dan perasaanmu.

1. Pengalaman apa yang kamu dapatkan setelah mempelajari pengenalan algoritma dan pemrograman?

.....

2. Apakah ada kesulitan yang kamu rasakan selama mempelajari materi pada bab ini? Subbab mana yang menurutmu sulit? jelaskan!

.....

3. Apakah kamu mendapatkan materi dan pemahaman baru setelah mempelajari materi ini? jelaskan!

.....

4. Ceritakan pengalamanmu setelah mempelajari materi pada bab ini!

.....

5. Apa yang akan kamu lakukan untuk mengaplikasikan materi pada bab ini?

.....



