1. **Algoritma**

* **Apa itu Algoritma?**
* Algoritma adalah serangkaian langkah-langkah logis dan tersruktur untuk menyelesaikan suatu masalah.
* Dalam pemograman,algoritma adalah “resep” yang memberitahu komputer apa yg harus dilakukan
* **Sejarah Algoritma**
* Istilah “algoritma” berasal dari nama matematikawan Persia, Muhammad Ibnu Musa Al-Khwarizmi (sekitar 780–850 M).
* Karyanya menjadi dasar bagi pengembangan aljabar dan konsep algoritma dalam komputasi modern.
* **Analogi Algoritma**
* Ada masalah →
* Buat algoritma (langkah-langkah solusi) →
* Ubah algoritma jadi program komputer
* Maka tercipta solusi nyata

Algoritma dibuat untuk menyelesaikan masalah, bukan menciptakan masalah baru

* **Contoh Algoritma**

**Bayangkan algoritma seperti resep makanan**

* Masalah : lapar, ingin nassi goreng
* Algoritma (resep) :

1. Siapkan bahan (nasi, bumbu, telur).
2. Panaskan minyak.
3. Masukkan bumbu, tumis.
4. Masukkan nasi, aduk rata.
5. Tambahkan telur, aduk.
6. Sajikan

* **Cara Mendeskripsikan Masalah**
* Menjabarkan masalah (dari klien/user)
* Merincikan masalah jadi bagian kecil
* Membuat sarana interaksi manusia & komputer  
  → Program harus mudah digunakan oleh manusia
* **Cara Mengubah Masalah ke Program**
* Identifikasi masalah : pahami kebutuhan dan tujuan program
* Rancang algoritma : buat langkah-langkah logis penyelesaian masalah
* Bahasa pemograman : tentukan bahasa yg sesuai (python : data, javascript : web).
* Tulis kode : terjemahkan algoritma ke dalam sintaks bahasa pemograman
* Uji & Debug : Pastikan program berjalan sesuai harapan
* **Kesimpulan Tentang Algoritma**
* Algoritma = proses **logis menyelesaikan masalah** tanpa bergantung pada bahasa pemrograman
* Urutan logika disebut juga **notasi algoritma**
* Bentuk algoritma bisa ditulis dalam:
  + ✏️ **Untaian kalimat deskriptif**
  + 🧭 **Flowchart**
  + 💻 **Pseudocode**
* **Notasi Algoritma**

**Algortima dapat dipresentasikan dalam beberapa bentuk**

* Kalimat Deskriptif : menggunakan bahasa sehari-hari
* Flowchart : menggunakan simbol grafis
* Pseudocode : mirip kode program, tapi tidak terikat sintaxs bahasa tertentu

1. **Kalimat Deskriptif**

* **Apa itu Kalimat Deskriptif?**
* Cara menuliskan intruksi algoritma dalam bentuk untaian kalimat
* Ditulis dalam bahasa yang jelas, baku, dan mudah dipahami.
* Mirip dengan procedure text (misalnya: resep atau panduan).
* **Tips Menulis Kalimat Deskriptif yang Baik**
* Gunakan **bahasa baku**
* Gunakan **bahasa yang mudah dimengerti**
* **Berkoresponden dengan logika pemrograman**
  + Artinya urutan kalimatnya harus sesuai dengan cara kerja program
* **Contoh Kalimat Deskriptif: Memasak Nasi**
* Langkah-langkah:

1. Ambil beras
2. Cuci beras
3. Masukkan ke rice cooker
4. Nyalakan rice cooker
5. Tunggu matang → Nasi siap dimakan

✅ Ini adalah bentuk **algoritma kehidupan nyata**

* **Contoh Kalimat Deskriptif di Pemrograman**
* **Topik:** Menghitung Luas Persegi Panjang
* **Langkah-langkah:**

1. Mulai,
2. Baca nilai panjang,
3. Baca nilai lebar,
4. Hitung luas dengan rumus : panjang dikali lebar,
5. Tampilkan hasil luas,
6. Selesai.

**📌 Catatan:**

* Urutan harus logis. Misalnya, Jangan tampilkan hasil sebelum input dimasukkan.
* **Pentingnya Bahasa Baku**

Gunakan bahasa yang formal dan konsisten untuk menghindari ambiguitas.

* **Contoh:**
  + ✅ *"Masukkan nilai panjang"* (baku)
  + ❌ *"Masukin panjangnya"* (tidak baku)
* Tujuannya agar algoritma lebih terstruktur, formal, dan profesional

### ****Apa itu Flowchart?****

### ****Apa itu Flowchart?****

* Flowchart adalah **representasi visual dari algoritma menggunakan simbol-simbol standart.**
* Mempermudah pemahaman alur logika program.
* **Setiap simbol memiliki makna dan fungsi spesifik**
* **Simbol-Simbol Dasar Flowchart**

| **Simbol** | **Nama** | **Fungsi** |
| --- | --- | --- |
| Oval atau kapsul | Terminator | Menandai awal dan akhir flowchart (Start/End) |
| Panah (Arrow) | Arah alur | Menunjukkan arah aliran proses |
| Segienam Horizontal (mirip trapesium) | Preparation | Menyatakan deklarasi/variabel awal |
| Jajar genjang | Input/Output | Memasukkan dan menampilkan data |
| Persegi panjang | Proses | Operasi atau Perhitungan |

Ini hanya sebagian simbol dasar. Ada banyak simbol lain untuk kasus yang lebih kompleks.

* **Catatan Perancangan Flowchart**
* Tidak ada rumus mutlak.
* Urutan dan isi tergantung **studi kasus**.
* Bisa fleksibel, **asal logis**.
* **Contoh Flowchart: Menghitung Luas Persegi Panjang**

#### **Urutan logika:**

1. **Start** (Terminator)
2. **Preparation**
   * Deklarasi variabel: panjang, lebar, luas
3. **Input Data**
   * Masukkan nilai panjang dan lebar
4. **Proses**
   * Hitung: luas = panjang × lebar
5. **Output Data**
   * Tampilkan hasil luas
6. **End** (Terminator)

### 🔍 ****Penjelasan Ilustrasi Variabel****

* **Variabel = tempat menyimpan nilai**, ibarat “keranjang” untuk menyimpan “baju”.
* **Variabel = “wadah” dimemori komputer untuk menyimpan nilai. Sebelum digunakan, variabel perlu dideklarasikan (disiapkan wadahnya).**
* Harus **menyediakan tempat dulu** (deklarasi) sebelum memasukkan nilai (input).

Contoh analogi:

"Siapkan keranjang dulu sebelum dimasukkan bajunya."  
"Siapkan uang dulu sebelum beli barang."

* **Tips Membuat Flowchart**

1. **Gunakan simbol yg benar sesuai fungsinya.**
2. **Pastikan alur logis dan mudah diikuti.**
3. **Gunakan bahasa yang konsisten dan jelas disetiap simbol.**
4. **Setiap flowchart harus memiliki satu titik mulai dan satu titik selesai.**

### ****Kesimpulan****

* Flowchart adalah **cara visual** menuliskan algoritma.
* **Simbol harus sesuai fungsi**.
* Berguna untuk **mempermudah pemahaman alur program** sebelum dibuat dalam bentuk kode.

1. **Apa itu Pseudocode?**

* **Apa itu Pseudocode?**
* **Pseudocode** adalah deskripsi algoritma yg menyerupai kode program.
* Menggunakan bahasa yg sederhana (seringkali inggris) tanpa terikat sintaxs bahasa pemograman apapun.
* Fokus pada logika, buka detail implementasi.
* **Ciri-ciri Pseudocode**

1. Menggunakan kata kunci umum seperti **START, END, READ, WRITE, IF, ELSE, WHILE, FOR.**
2. Tidak memiliki standar baku yang ketat, namun harus konsisten.
3. Mudah diterjemahkan ke berbagai bahasa pemograman.

* **Contoh Pseudocode: Luas Persegi Panjang**

// Deklarasi variabel

| Keterangan istilah: |
| --- |
| * DECLARE → deklarasi variabel |
| * INPUT → masukan dari user |
| * SET → memberi nilai ke variabel |
| * DISPLAY → menampilkan output ke layar |

DECLARE panjang, lebar, luas AS INTEGER

// Input

INPUT panjang

INPUT lebar

// Proses

SET luas = panjang \* lebar

// Output

DISPLAY luas

END

* **Perbandingan: Deskriptif vs Flowchart vs Pseudocode**

| **Aspek** | **Kalimat Deskriptif** | **Flowchart** | **Pseudocode** |
| --- | --- | --- | --- |
| **Bentuk** | Untaian Kalimat | Diagram/simbol grafis | Mirip kode program |
| **Target Audiens** | Non-programer,awam | Visual thinker, desain sistem | Calon programmer |
| **Kelebihan** | Mudah dimengerti | Ringkas & visual | Mendekati kode nyata,mudah diimplementasi |
| **Kekurangan** | Kurang tersruktur untuk kompleksitas | Membutuhkan alat gambar, sulit untuk algoritma besar | Bisa rancu jika tidak konsisten |

* **Kapan Menggunakan Setiap Notasi?**
* **Kalimat Deskriptif** → untuk menjelaskan konsep dasar kepada non-teknis.
* **Flowchart** → untuk memvisualisasikan alur komplesk atau presentasi.
* **Pseudocode** → untuk merancang logika program sebelum menulis kode asli, atau berkomunikasi antar programer.

💡 Penting: Pilih notasi yang paling sesuai dengan audiens dan kompleksitas masalah.

## **Pengertian Variabel dan Konstanta**

## **Pengertian Variabel dan Konstanta**

* **Variabel : Sebuah wadah di memori komputer yang nilainya dapat berunah selama program berjalan.**
* **Konstanta :** Sebuah wadah di memori komputer yang nilainya tetap dan tidak dapat diubah setelah nilainya ditetapkan

### 📌 Contoh:

Misalnya rumus **luas lingkaran**:  
L = π × r²

* Nilai **π (pi)** biasanya ditetapkan sebagai 3.14 atau 22/7. Karena nilainya tetap, maka **π disebut sebagai konstanta**.
* Nilai **r (jari-jari)** tergantung dari soal. Bisa 1, 2, 3, dan seterusnya. Karena nilainya bisa berubah-ubah, maka **r disebut sebagai variabel**.

Contoh lain, rumus **volume kubus**:  
V = sisi × sisi × sisi

Karena nilai sisi dapat berubah, maka sisi adalah **variabel**. Tidak ada konstanta yang digunakan dalam rumus ini.

* **Konsep Variabel dan Konstanta**

1. **Penyimpanan sementara :**  
   Data variabel/Konstanta disimpan di RAM dan akan hilang saat program berakhir.
2. **Nilai variabel fleksibel :** Nilai variabel dapat diubah berkali-kali.
3. **Nilai Konstanta tetap :**  
   Nilai konstanta tidak bisa diubah setelah diinisialisasi.
4. **Satu data per wadah :** Setiap variabel/konstanta hanya menyiman satu nilai pada satu waktu.
5. **Nilai terakhir berlaku : Jika variabel diberi nilai berulang, hanya nilai terakhir yang akan digunakan** **Contoh:** x = 10 x = 5 x = 50 **// nilai x sekarang adalah 50**

* **Aturan Penulisan Variabel dan Konstanta**

1. **Dimulai dengan huruf**  (A-Z atau a-z). Tidak boleh diawali dengan angka atau simbol seperti @, #, %, dll.
2. **Bersifat (case-sensitive)**, (misal : “nama” dan “Nama” adalah berbeda)
3. **Hanya boleh mengandung huruf, angka, dan garis bawah (underscore ” \_”).**
4. **Tidak boleh menggunakan spasi.**
   * Jika ingin menyusun dua kata, gunakan:
     + **Underscore**, contoh: panjang\_lebar
     + **CamelCase**, contoh: panjangLebar
5. **Nama variabel dan konstanta sebaiknya mencerminkan data yang disimpan.**
   * Contoh baik: panjang, lebar, luas
   * Contoh buruk: a, b, c (tidak jelas maknanya)

* **Notasi Pseudocode untuk Variabel dan Konstanta**
* Dalam algoritma dan pseudocode, penulisan variabel dan konstanta mengikuti aturan berikut:
* **Konstanta** ditulis dengan keyword:

CONSTANT Nama\_Konstanta = Nilai

* **Variabel** ditulis dengan keyword:

DECLARE Nama\_Variabel = Nilai

* + Contoh:

CONSTANT PI = 3.14

DECLARE radius = 5

# **Tipe Data dalam Pemrograman**

## **Apa itu Tipe Data?**

1. Tipe data adalah klasifikasi yang menentukan jenis nilai yang dapat disimpan oleh suatu variabel.Penting untuk:

* Mengoptimalkan penggunaan memori.
* Memastikan operasi yang benar pada data.
* Menccegah kesalahan program.
* **Empat Jenis Utama Tipe Data**

| **No** | **Jenis Tipe Data** | **Contoh** | **Keterangan Singkat** |
| --- | --- | --- | --- |
| 1 | INTEGER ( Bilangan Bulat ) | 5, -10, 200 | Angka tanpa desimal |
| 2 | FLOAT/DOUBLE (Bilangan Pecahan) | 3.14, -0.75 | Angka desimal |
| 3 | CHAR (Karakter) & STRING (Teks) | 'A', "Halo Dunia" | Huruf/simbol atau kumpulan karakter |
| 4 | Boolean (Logika) | true, false | Nilai benar atau salah |

* **Tipe Data Bilangan Bulat**

| **Tipe** | **Ukuran Bit** | **Jangkauan Nilai** |
| --- | --- | --- |
| byte | 8-bit | -128 hingga 127 |
| short | 16-bit | -32.768 hingga 32.767 |
| int | 32-bit | -2.147.483.648 hingga 2.147.483.647 |
| long | 64-bit | Nilai sangat besar (miliaran++) |

📌 Tips: Gunakan tipe data terkecil yang cukup agar program efisien.

* **Tipe Data Bilangan Pecahan**

| **Tipe** | **Ukuran Bit** | **Keterangan** |
| --- | --- | --- |
| float | 32-bit | Akurasi sedang |
| double | 64-bit | Akurasi tinggi |

* **Tipe Data Karakter & Teks**

| **Tipe** | **Keterangan** |
| --- | --- |
| char | Menyimpan satu karakter/simbol saja ('A') |
| String | Menyimpan banyak karakter ("Halo") |

📌 catatan: char ≠ String. char hanya bisa 1 huruf/simbol.

* **Tipe Data Boolean**

| **Tipe** | **Nilai Yang Diterima** |
| --- | --- |
| boolean | true atau false |

Cocok digunakan untuk kondisi if, while, atau perbandingan.

* **Kenapa Memilih Tipe Data Itu Penting?**
* **Efisiensi memori :** Memilih tipe data yg tepat menghemat ruang memori.
* **Validasi Data :** Memastikan data yang disimpan sesuai dengan yang diharapkan.
* **Operasi yang benar :** Mencegah kesalahan saat melalukan operasi.
* **Keterbacaan kode :** Membuat kode lebih mudah dipahami.

# **Apa Itu Operator?**

### Apa Itu Operator?

**Operator** adalah simbol-simbol khusus yang digunakan untuk melakukan operasi pada nilai (operand) dalam program.

### Jenis-Jenis Operator

### ****Operator Penugasan (Assignment Operator)****

* + **Simbol :** =
  + **Fungsi :** Memberikan nilai ke variabel.
  + **Contoh :** umur = 25 (variabel umur diisi nilai 25).
  + **Catatan :** bukan berarti "sama dengan" dalam matematis, tapi merupakan **tanda penugasan “diisi dengan”**.

### ****Operator Aritmatika (Arithmetic Operator)****

Digunakan untuk melakukan perhitungan matematis.

| **Operasi** | **Simbol** | **Contoh** |
| --- | --- | --- |
| Penjumlahan | + | x + y |
| Pengurangan | - | x - y |
| Perkalian | \* | x \* y |
| Pembagian | / | x / y |
| Modulus (sisa bagi) | % | x % y |

### ****Operator Perbandingan (Comparison Operator)****

Digunakan untuk membandingkan dua nilai dan menghasilkan nilai **boolean** (true/false).

| **Operasi** | **Simbol** |
| --- | --- |
| Sama dengan | == |
| Tidak sama dengan | != |
| Lebih besar dari | > |
| Kurang dari | < |
| Lebih besar atau sama dengan | >= |
| Kurang dari atau sama dengan | <= |

### ****Operator Logika (Logical Operator)****

Digunakan untuk **menggabungkan ekspresi boolean**.

| **Operasi** | **Simbol** | **Keterangan** |
| --- | --- | --- |
| AND | && | Bernilai **true** jika semua kondisi benar  Jika ada **satu saja kondisi false**, hasilnya akan false |
| OR | || | Hasilnya **true jika salah satu atau kedua kondisi bernilai true**.  Hanya akan menghasilkan false jika **semua kondisi bernilai false** |
| NOT | ! | Membalik nilai boolean |

#### 

| **A** | **B** | **A && B** | **A || B** | **! (A)** | **! (B)** |
| --- | --- | --- | --- | --- | --- |
| B (True) | B (True) | B (True) | B (True) | S (False) | S (False) |
| B (True) | S (False) | S (False) | B (True) | S (False) | B (True) |
| S (False) | B (True) | S (False) | B (True) | B (True) | S (False) |
| S (False) | S (False) | S (False) | S (False) | B (True) | B (True) |

* + **Tabel Kebenaran**

### Penutup

Operator adalah bagian penting dalam pemrograman. Mereka digunakan untuk:

* Menyimpan nilai
* Melakukan perhitungan
* Membandingkan ekspresi
* Mengontrol alur logika program

### ****Logika Percabangan (Conditional Statement)****

Logika percabangan digunakan untuk **menjalankan perintah tertentu berdasarkan kondisi**. Jika kondisi bernilai benar (true), maka perintah akan dijalankan.

### ****Contoh Kehidupan Sehari-hari:****

**Situasi:**  
Kamu ingin pergi ke luar rumah.

**Percabangannya:**

* Jika **hujan**, maka **bawa payung**.
* Jika **tidak hujan**, maka **tidak perlu payung**.

### Dalam simbol flowchart (dessicion)

### ****Contoh Kasus: Penerimaan Karyawan****

**Syarat:**

1. Umur ≥ 17 tahun
2. Jenis kelamin = Laki-laki / Perempuan (L || P)
3. Pendidikan ≥ SLTA

### ****Contoh Inputan Kemampuan :****

| **Kondisi** | **Hasil** |
| --- | --- |
| Umur = 17 tahun | Lolos syarat umur |
| Jenis kelamin = L | Lolos syarat gender |
| Pendidikan = SLTP | ❌ Tidak lolos pendidikan |

* **Kondisi dengan Operator AND (**&&**)** Semua syarat harus terpenuhi

| **Kondisi** | **Hasil** |
| --- | --- |
| Umur = 17 tahun | Lolos syarat umur |
| Jenis kelamin = L | Lolos syarat gender |
| Pendidikan = SLTP | ❌ Tidak lolos pendidikan |

📌 **Hasil (AND):** ❌ **Tidak lolos**, karena satu saja tidak terpenuhi (pendidikan).

## **Kondisi dengan Operator OR (**||**)**

Cukup satu syarat yang terpenuhi agar hasilnya true.

| **Kondisi** | **Hasil** |
| --- | --- |
| Umur = 17 tahun | Lolos syarat umur |
| Jenis kelamin = L | Lolos syarat gender |
| Pendidikan = SLTP | ❌ Tidak lolos pendidikan |

📌 **Hasil (OR):** ✅ **Lolos**, karena **ada dua syarat yang terpenuhi**.

* **jenis-jenis logika percabangan**

| **Jenis Percabangan** | **Cocok Untuk** |
| --- | --- |
| IF Tunggal | Satu kondisi, satu aksi jika benar |
| IF-ELSE (Ganda) | Dua kemungkinan aksi (true lakukan a, false lakukan b) |
| IF-ELSE IF-ELSE (Majemuk) | Banyak kondisi berurutan |
| SWITCH CASE | Memilih aksi berdasarkan nilai pasti dari satu variabel |

## **Jenis Percabangan: IF Tunggal dan IF Ganda**

1. **IF Tunggal**

**📌 Pengertian:**  
Percabangan **IF tunggal** digunakan jika hanya ada **satu aksi** yang dilakukan saat kondisi **bernilai benar**.  
Jika kondisi salah, maka **tidak ada aksi** yang dilakukan.

**💡 Analogi Kehidupan:**

"Jika hujan, maka saya akan membawa payung."  
(Tidak hujan? Tidak ada aksi.)

**🔁 Flowchart Sederhana:**

[Mulai]

|

[hujan AS STRING]

|

[Apakah hujan?]

|

(jawaban (Ya))

|

[Bawa payung]

|

[Selesai]

**💻 Pseudocode IF Tunggal:**

DECLARE hujan AS BOOLEAN

IF hujan == TRUE THEN

DISPLAY "Bawa payung"

ENDIF

1. **IF Ganda (IF - ELSE)**

**📌 Pengertian:**  
Percabangan **IF ganda** digunakan jika ada **dua aksi yang berbeda**:

* Jika kondisi benar, maka lakukan aksi 1
* Jika kondisi salah, maka lakukan aksi 2

**💡 Analogi Kehidupan:**

"Jika hujan, maka saya membawa payung, jika tidak hujan, maka saya memakai kacamata hitam."

**🔁 Flowchart Sederhana:**

[Mulai]

|

[hujan AS STRING]

|

[Apakah hujan?]

/ \

(Ya) (Tidak)

| |

[payung] [Kacamata]

\ /

[Selesai]

**💻 Pseudocode IF Ganda:**

DECLARE hujan AS BOOLEAN

IF hujan == TRUE THEN

DISPLAY "Bawa payung"

ELSE

DISPLAY "Pakai kacamata"

ENDIF

## **If Majemuk / If Bersarang (Nested If)**

**📌 Pengertian:**

If bersarang digunakan saat **ada lebih dari dua pilihan**. Artinya, **kita perlu mengecek satu kondisi**, lalu **di dalamnya ada pengecekan lagi**.

**💡 Analogi Kehidupan:**

Adi pergi ke rumah Bagas  
→ Kalau Bagas ada → main dengan Bagas  
→ Kalau Bagas **tidak ada**, pergi ke rumah Budi  
→ Kalau Budi ada → main dengan Budi  
→ Kalau Budi juga tidak ada → pulang ke rumah

**🔁 Flowchart Sederhana:**

[Mulai]

|

[jawaban,jawaban2 AS STRING]

|

Cek Bagas ada?

/ \

Ya Tidak

/ \

"Main dgn Bagas" Cek Budi ada?

/ \

Ya Tidak

/ \

"Main dgn Budi" "Pulang"

\ /

[Selesai]

**💻 Pseudocode IF Majemuk:**

DECLARE Bagas,Budi AS Boolean

IF Bagas\_ada THEN

OUTPUT "Main dengan Bagas"

ELSE

IF Budi\_ada THEN

OUTPUT "Main dengan Budi"

ELSE

OUTPUT "Pulang ke rumah"

ENDIF

ENDIF

## **Percabangan Case?**

Percabangan case digunakan saat kita ingin **memilih salah satu dari banyak kondisi** (lebih dari 2) **berdasarkan satu nilai tertentu**.

✅ Cocok saat:

* Pilihan banyak tapi hanya berdasarkan **satu variabel**
* if-else jadi terlalu rumit atau panjang

**💻 Pseudocode IF Ganda:**

SWITCH (variabel)

CASE nilai1:

// Aksi untuk nilai1

BREAK

CASE nilai2 :

// Aksi untuk nilai2

BREAK

DEFAULT :

// Aksi jika tidak ada nilai yang cocok

END SWITCH

### 📊 ****Contoh Kasus: Pilih Hari****

Misalnya kamu input angka 1–7, lalu program menampilkan nama hari.

#### ✅ **Pseudocode:**

INPUT hari

CASE hari OF

1 : OUTPUT "Senin"

2 : OUTPUT "Selasa"

3 : OUTPUT "Rabu"

4 : OUTPUT "Kamis"

5 : OUTPUT "Jumat"

6 : OUTPUT "Sabtu"

7 : OUTPUT "Minggu"

ELSE : OUTPUT "Hari tidak valid"

ENDCASE

### ****Perbedaan**** IF ****vs**** CASE

| **Aspek Perbandingan** | **IF** | **CASE** |
| --- | --- | --- |
| **Jumlah Kondisi** | Bisa digunakan untuk kondisi lebih dari satu:  >, <, >=, <=, !=, == | Hanya digunakan untuk satu ekspresi dengan beberapa nilai tetap (konstan) |
| **Jenis Kondisi** | Fleksibel (Perbandingan, logika kompleks, range nilai) | Nilai pasti (konstan) dari satu variabel |
| **Bentuk Kondisi** | Bisa ekspresi kompleks:  bilangan, string, char, boolean, logika gabungan | Nilai tetap seperti bilangan, string, char, atau boolean sederhana |
| **Keterbacaan** | Bisa panjang dan rumit jika banyak kondisi | Lebih rapi untuk banyak pilihan nilai pasti |
| **Penggantian** | Tidak selalu bisa diganti dengan CASE  (terutama jika logika kompleks) | Bisa diganti dengan IF |

### 📌 ****Kesimpulan Singkat:****

* Gunakan IF jika kondisi logikanya **beragam dan kompleks**
* Gunakan CASE jika ingin **memilih 1 dari banyak berdasarkan 1 variabel tetap**

## **Logika Perulangan (Looping)**

* **Pengertian:**  
  Logika perulangan adalah proses mengulang eksekusi suatu pernyataan (statement) secara berulang hingga mencapai batas atau kondisi tertentu yang telah ditentukan.

### ****Analogi Sederhana:****

Misalnya, seseorang ingin naik tangga 10 anak tangga. Ia akan mulai dari anak tangga pertama (nilai awal), terus melangkah (step) selagi belum mencapai anak tangga ke-10 (kondisi).

### ****Dalam Pemrograman, perulangan terdiri dari 3 komponen penting:****

1. **Inisialisasi** – Nilai awal variabel kontrol.
2. **Kondisi** – syarat agar perulangan terus berjalan.
3. **Iterasi** – perubahan nilai variabel kontrol (increment/decrement)setiap putaran.

### ****Jenis-Jenis Logika Perulangan:****

1. **For Loop**

## **Pengertian For Loop**

**For loop** adalah salah satu jenis perulangan dalam pemrograman yang digunakan untuk mengulang blok kode dengan **jumlah iterasi yang sudah diketahui sebelumnya**.

### ****Struktur Umum:****

FOR (inisialisasi; kondisi; iterasi) DO

// Blok kode yang diulang

END FOR

* **Nilai awal**: inisialisasi variabel sebelum perulangan dimulai.
* **Kondisi**: syarat agar perulangan terus berjalan.
* **Step**: perubahan nilai variabel tiap putaran (misalnya, +1 atau -1)

## **Analogi Sederhana:**

Bayangkan kamu ingin menghitung dari 1 sampai 5. Kamu mulai dari angka 1, lalu naik satu per satu sampai 5. Ini bisa dilakukan dengan for loop.

## **Contoh : Menampilkan angka 1 sampai 5**

FOR i = 1 TO 5 DO

PRINT i

END FOR

### ****Output:****

1

2

3

4

5

## **Penjelasan:**

* int i = 1 → memulai dari angka 1
* i <= 5 → perulangan terus berjalan selama i kurang dari atau sama dengan 5
* i++ → setiap kali perulangan selesai, nilai i ditambah 1
* System.out.println(i) → menampilkan nilai i ke layar

1. **While Loop**

## **Pengertian While Loop**

**While loop** adalah salah satu jenis perulangan dalam pemrograman yang digunakan untuk **menjalankan kode selama kondisi bernilai benar (true)**.

Berbeda dengan for, pada while **jumlah perulangannya belum tentu diketahui**, tergantung kondisi.

### ****Struktur Umum:****

WHILE kondisi DO

// Blok kode yang diulang

// Pastikan ada perubahan yang membuat kondisi suatu saat menjadi false

END WHILE

* Perulangan akan terus dijalankan **selama kondisi terpenuhi (true)**.
* Jika kondisi **salah (false)**, maka perulangan berhenti.

## **Contoh 1: Menampilkan angka 1 sampai 5**

DECLARE i = 1

WHILE i <= 5 DO

PRINT i

i = i + 1

END WHILE

### ****Output:****

1

2

3

4

5

### ****Penjelasan:****

* int i = 1 → nilai awal
* while (i <= 5) → perulangan terus berjalan selama i kurang dari atau sama dengan 5
* i++ → setiap iterasi, nilai i ditambah 1

1. **Do While Loop**

## **Pengertian Do While Loop**

**Do while loop** adalah jenis perulangan yang **menjalankan kode minimal satu kali**, lalu akan terus mengulang **selama kondisi bernilai benar (true).**

Perbedaannya dengan while loop:

* **while** → kondisi dicek di awal
* **do while** → kondisi dicek di akhir → jadi **pasti dijalankan sekali**, walau kondisi awalnya salah.

## **Struktur Umum:**

WHILE kondisi DO

// Blok kode yang diulang

// Pastikan ada perubahan yang membuat kondisi suatu saat menjadi false

END WHILE

## **Contoh 1: Menampilkan angka 1 sampai 5**

### ****Penjelasan:****

* Baris dalam do {} dijalankan **terlebih dahulu**.
* Setelah itu kondisi i <= 5 dicek.
* Jika kondisi masih benar, perulangan dilanjutkan.

## **Contoh :**

DECLARE i = 10

DO

PRINT “halo”

i = i + 1

WHILE i < 5

### ****Output:****

Halo

Perbedaannya dengan while loop :

1. While = kondisi dicek diawal
2. Do while = kondisi dicek diakhir -> jadi pasti dijalankan sekali, walau kondisi awalnya salah
3. **Foreach Loop**
   * Digunakan untuk **iterasi langsung** pada elemen-elemen dalam array, list, atau koleksi.

## **Konsep Array dalam Pemrograman**

### Pengertian:

**Array** adalah tipe data yang digunakan untuk menyimpan **kumpulan elemen (nilai atau variabel)** dalam satu variabel tunggal, di mana setiap elemen memiliki **index (posisi)**.

### Kenapa Menggunakan Array?

#### **Manajemen Memori**

* **Sebelum array:**

binatang1 = "ayam"

binatang2 = "sapi"

binatang3 = "kambing"

* **Dengan array:**

binatang = ["ayam", "sapi", "kambing"]

**✅ Lebih hemat, efisien, dan mudah dikelola.**

#### **Mempermudah Pengelolaan Data**

Anggap array seperti **lemari dengan banyak laci**. Tiap laci punya nomor (index), dan isinya bisa kamu ambil atau ganti dengan cepat.

### ****Sifat-Sifat Array****

#### **Variabel Jamak dengan Nama yang Sama**

* Semua elemen diakses lewat satu nama, dibedakan lewat indeks.
* 📌 **Contoh (java):**

Int [] angka = {10, 20, 30};

System.out.println (angka [0]); // Output: 10

System.out.println (angka [1]); // Output: 20

#### 🔑 **Pasangan Key dan Value (Key-Value Pair)**

* Di array biasa (list), index adalah **angka urut otomatis**.
* Dalam array yang berbentuk dictionary (Python), key bisa ditentukan sendiri.
* 📌 **Contoh:**

hewan = {"a": "ayam", "s": "sapi", "k": "kambing"}

print(hewan["s"]) # Output: sapi

#### 🔄 **Bisa Menyimpan Tipe Data Berbeda (di bahasa tertentu seperti Python)**

* 📌 **Contoh:**

campuran = ["apel", 123, True, 45.6]

print(campuran[2]) # Output: True

Di bahasa seperti Java, C, C++, **semua elemen array harus satu jenis (homogen).**

* 📌 **Contoh Java:**

Object [ ] campuran = {"apel", 123, True, 45.6};

System.out.println(campuran [2];) // Output: True

#### 📚 **Array Bersarang (Array di dalam Array)**

* Array yang elemen-elemennya juga berupa array.
* 📌 **Contoh:**

data = [

["apel", "jeruk"],

["ayam", "bebek"]

]

print(data[1][0]) # Output: ayam

* 📌 **Contoh Java:**

String [ ] [ ] data = {

{"apel", "jeruk"},

{"ayam", "bebek"}

};

System.out.println(data[1][0]); // Output: ayam

### 🔧 ****Contoh Flowchart Array Sederhana****

**Tujuan:** Menyimpan dan menampilkan isi array

Start

↓

Deklarasi array [3]

↓

Input array[0]

Input array[1]

Input array[2]

↓

Output array[0]

Output array[1]

Output array[2]

↓

End

### 💻 ****Pseudocode Array****

DECLARE buah [ 3 ] AS STRING

FOR i = 0 TO 2

INPUT buah [ i ]

END FOR

FOR i = 0 TO 2

DISPLAY buah [ i ]

END FOR

### ****Cara Membuat Array Bersarang****

📌 Contoh Python:

siswa = [

[ "Adit", 90 ],

[ "Budi", 85 ],

[ "Cici", 95 ]

]

Print ( siswa [ 0 ] [ 0 ] ) # Output: Adit

Print ( siswa [ 2 ] [ 1 ] ) # Output: 95

📌 Contoh Java:

String [ ] nama = { “Adit”, “Budi”, “Cici” };

Int [ ] nilai = { 90, 85, 95 };

System.out.println ( nama [ 0 ] ); // Output : Adit

System.out.println ( nama [ 2 ] ); // Output : 95

## **Konsep Function dalam Pemrograman**

### 🔍 ****Apa itu Function?****

Function adalah **program kecil atau blok kode terpisah** dalam program utama yang bisa dipanggil kapan saja. Function dapat memiliki **input (parameter)** dan dapat **mengembalikan output (return value)**.

### ****Analogi Function****

Function itu seperti **mesin atau alat**:

* Kita **masukkan bahan (input)**,
* Mesin **memprosesnya (logic)**,
* Lalu **menghasilkan sesuatu (output)**.

**Contoh:**

* Mesin kopi:
  + Input: air + bubuk kopi
  + Output: kopi panas
* Function:

def tambah ( a, b ):

return a + b

### ****Karakteristik Function****

1. Program kecil yang berdiri sendiri dalam program utama.
2. Bisa menerima input (argumen/parameter).
3. Bisa menghasilkan output (return).
4. Output function bisa menjadi input untuk function lain.
5. Wajib dibuat dulu sebelum dipanggil.

### ****Manfaat Function****

1. ✅ **Reusability** – Bisa dipakai berulang-ulang di bagian lain program.
2. ✅ **Manajemen Memori** – Fungsi dipanggil hanya saat dibutuhkan.
3. ✅ **Efisiensi Kode** – Tidak perlu menulis kode yang sama berulang-ulang.
4. ✅ **Mudah Dipahami** – Kode terstruktur dan jelas bagi programmer lain.

### ****Kategori Function****

### 1. ****Built-in Function (fungsi bawaan)****

* Sudah disediakan oleh bahasa pemrograman.
* Tinggal dipakai.
* 📌 Contoh (Python):

Print ( "Halo" )

Len ( "abc" ) // menghitung jumlah karakter string, hasil : 3

* 📌 Contoh (Java):

String teks = “halo” ;

System.out.println ( teks.length ( ) ) ; // menghitung jumlah karakter string, hasil : 3

### 2. ****User-Defined Function (fungsi buatan sendiri)****

* Dibuat oleh programmer sendiri sesuai kebutuhan.
* 📌 Contoh (Python):

def luas\_persegi ( sisi ) :

return sisi \* sisi

# Pemanggilan fungsi

hasil = luas\_persegi ( 5 )

print ( "Luas persegi:", hasil ) # Output: 25

* 📌 Contoh (Java):

public class FungsiLuasPersegi {

// Fungsi user-defined

public static int luasPersegi ( int sisi ) {

return sisi \* sisi;

}

public static void main ( String [ ] args ) {

// Memanggil fungsi dan menyimpan hasilnya

int hasil = luasPersegi ( 5 );

// Menampilkan hasil

System.out.println ( "Luas persegi: " + hasil );

Print ( "Luas persegi:", hasil ) // Output: 25

}

}

### ****Pseudocode Membuat Function****

FUNCTION namaFunction ( parameter1, parameter2 )

DEKLARASI variabel

LAKUKAN proses

RETURN nilai

END FUNCTION

PROGRAM UTAMA

PANGGIL namaFunction ( dengan\_argumen )

END PROGRAM

📌 **Contoh Pseudocode:**

FUNCTION Tambah ( a, b )

RETURN a + b

END FUNCTION

PROGRAM UTAMA

HASIL ← Tambah ( 4, 5 )

DISPLAY HASIL

END PROGRAM

### ****Flowchart Function Sederhana****

Tujuan: Membuat fungsi untuk menjumlahkan 2 angka

+-----------------+

| Start |

+-----------------+

|

v

+--------------------+

| Input A, B |

+--------------------+

|

v

+--------------------------+

| Call Function |

| Jumlahkan (A, B ) |

+---------------------------+

|

v

+--------------------+

| Output Hasil |

+--------------------+

|

v

+------------------+

| End |

+------------------+

📌 Di dalam function Jumlahkan ( A, B ):

* Proses: C = A + B
* Return: C

### ****Parameter dan Argumen dalam Pemrograman****

#### 📌 **Pengertian:**

* **Parameter** adalah variabel yang ditulis di dalam tanda kurung () saat sebuah **fungsi (function/procedure)** dibuat. Fungsinya untuk **menampung nilai** yang akan dikirim saat fungsi itu dipanggil.
* **Argumen** adalah nilai **yang dikirimkan ke fungsi** saat fungsi tersebut dipanggil. Nilai ini nantinya **diterima oleh parameter** untuk digunakan dalam proses di dalam fungsi.

#### 🔍 **Analogi Sederhana (Mesin Jagung):**

Bayangkan kita punya **mesin pengolah jagung**.

* **Parameter = Lubang input mesin**  
  Mesin butuh input jagung untuk diproses. Lubang ini bisa diibaratkan sebagai **parameter**.
* **Argumen = Jagung yang dimasukkan ke mesin**  
  Jagung yang kita masukkan ke dalam lubang adalah **argumen**. Mesin tidak bisa bekerja jika tidak diberi jagung.

➡️ Jadi: **Fungsi = Mesin**, **Parameter = Lubang masuk**, **Argumen = Jagung yang dimasukkan**

#### **Contoh Pseudocode:**

// DECLARE

DECLARE integer x,y

// WRITE FUNCTION (User Defined Function)

Function name ( x,y )

RETURN x\*y

Endfunction

// CALL FUNCTION

CALL : name ( 10,2 )

DISPLAY ( call : name ( 10,2 ) )

**📝 Keterangan:**

* x,y adalah **parameter**
* 10,2 radalah **argumen**

#### 🔁 **Contoh Flowchart (Deskripsi Teks)**

**Judul: Fungsi dengan Parameter**

1. **Start 1.start**
2. **angka** (10,2) 2.deklarasi x,y
3. **Panggil Prosedur angka (10,2) 3.baca x,y**
4. **Tampilkan "20" 4.kembali x\*y**
5. **End**

✅ **Kesimpulan**

* Parameter = Tempat menerima data saat fungsi dibuat
* Argumen = Nilai yang dikirim saat fungsi dipanggil
* Pemisahan ini membantu fungsi agar bisa digunakan ulang dengan data yang berbeda-beda