STRUKTUR DASAR ALGORITMA

Oleh :

Yasinta Bella Fitriana, S.SI., M.Kom

PEMBELAJARAN DARING KOLABORATIF (PDK)
UNIVERSITAS MUHAMMADIYAH SIDOARJO & UNIVERSITAS MUHAMMADIYAH PAPUA

Struktur Runtunan *(Sequence)*

- Struktur yang digunakan untuk mengerjakan jenis program yang pernyataannya sequential atau berurutan
- struktur algoritma **paling dasar** yang berisi rangkaian instruksi yang diproses secara sekuensial, satu persatu, mulai dari intsruksi pertama sampai instruksi terakhir .
- tidak memuat lompatan atau pengulangan didalamnya
- dalam sains biasanya digunakan untuk melakukan perhitungan pada kasus yang melibatkan rumus-rumus sederhana dengan melibatkan operator penjumlahan, pengurangan, dan perkalian.

Karakteristik Struktur Runtunan

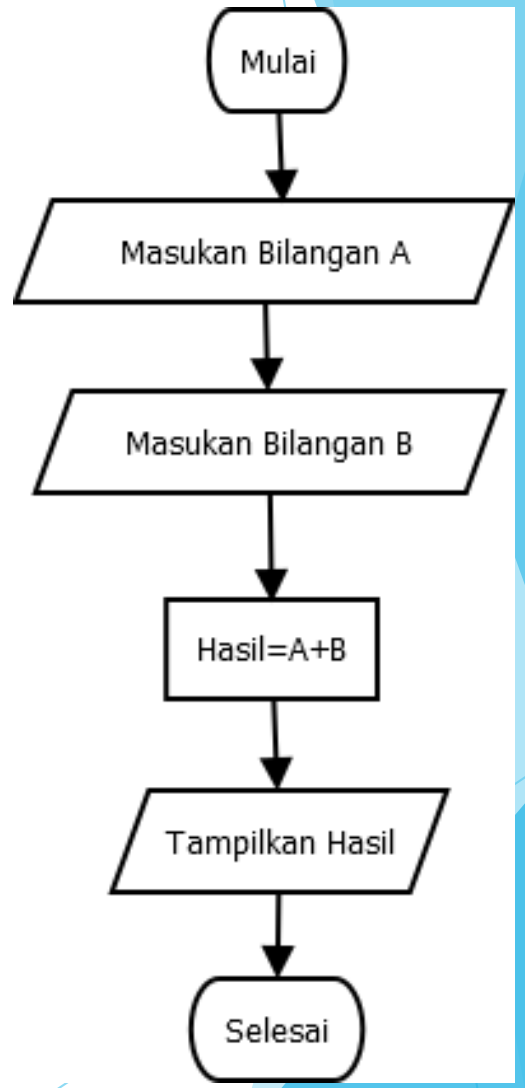
Intruksi dalam struktur urut memiliki karakteristik seperti:

1. Tiap perintah dikerjakan satu per satu sebanyak sekali
2. Pelaksanaan perintah dilakukan secara berurutan
3. Perintah terakhir merupakan akhir dari algoritma
4. Perubahan urutan dapat menyebabkan hasil yang berbeda

Contoh Struktur Runtunan

- Algoritma menentukan Luas Persegi Panjang

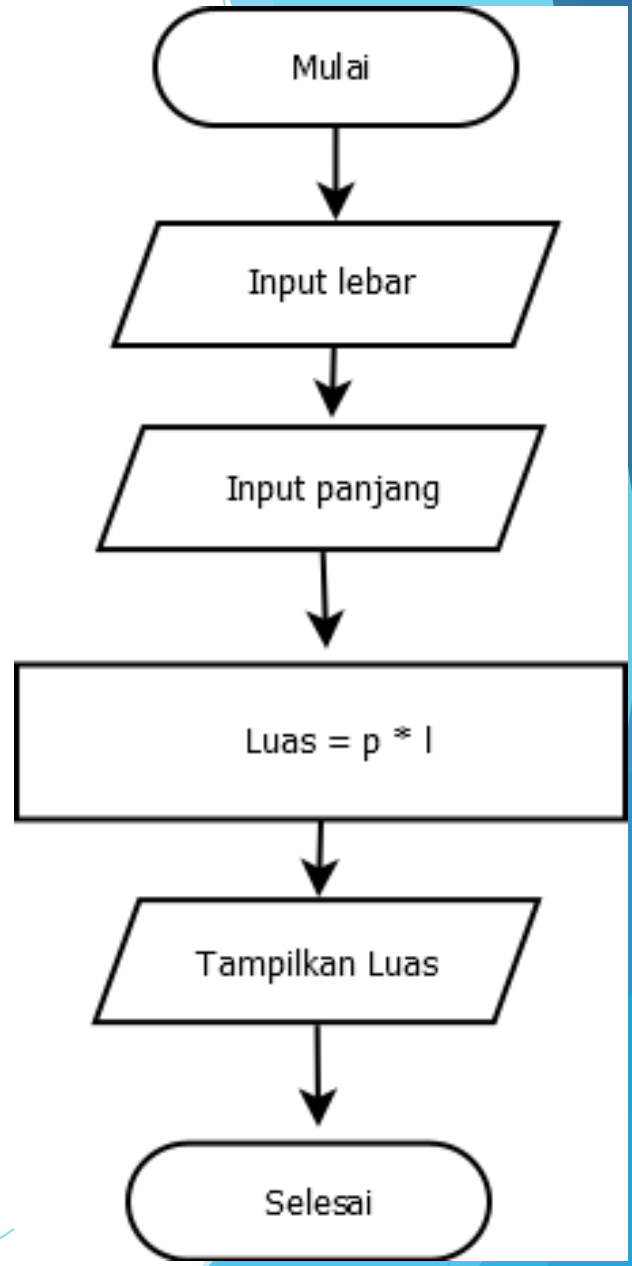
Langkah 1. Mulai
Langkah 2. Masukkan Bilangan A
Langkah 3. Masukkan Bilangan B
Langkah 4. Hitung Hasil = A+B
Langkah 5. Tampilkan Hasil
Langkah 6. Selesai



Contoh Struktur Runtunan

- Algoritma menjumlahkan dua bilangan

Langkah 1. Mulai
Langkah 2. Masukkan data leba
Langkah 3. Masukkan data panjang
Langkah 4. Hitung luas = panjang * lebar
Langkah 5. Tampilkan luas
Langkah 6. Selesai

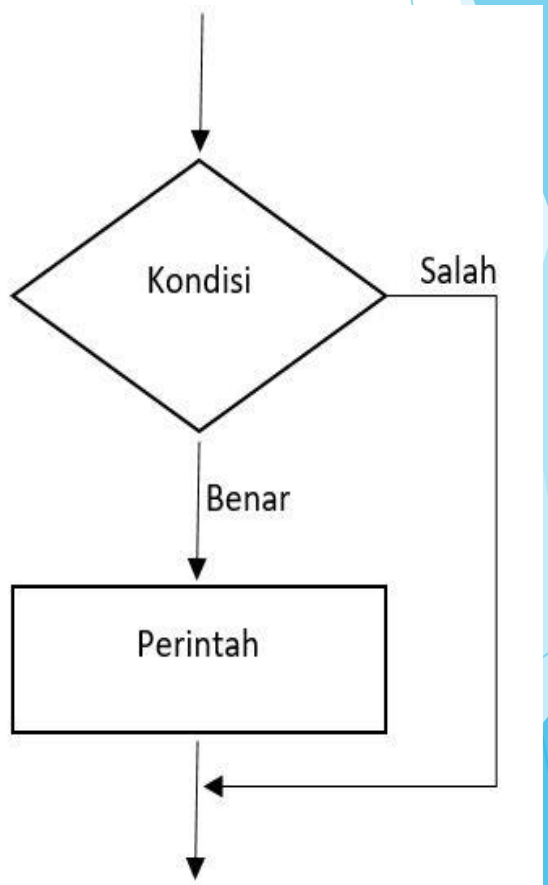


Struktur Dasar Pemilihan *(Selection)*

- Struktur yang digunakan pada program yang memerlukan proses pengujian kondisi untuk mengambil suatu keputusan apakah suatu baris perintah akan diproses atau tidak.
- Pengujian kondisi ini dilakukan untuk memilih salah satu dari beberapa alternatif yang tersedia.
- Tidak semua baris program akan dikerjakan pada struktur ini, melainkan hanya baris yang memenuhi syarat saja
- Notasi algoritma untuk struktur pemilihan menggunakan dua konstruksi berikut ini :
  1. IF – THEN
  2. CASE

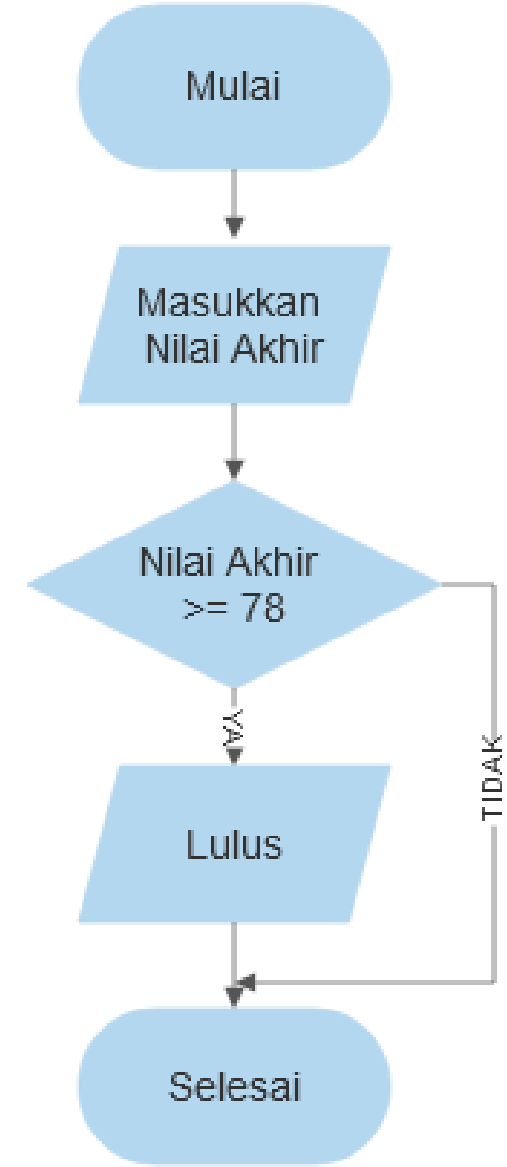
Struktur Pemilihan 1 kasus (IF-THEN)

- Perintah if….then.. digunakan jika struktur pemilihan hanya memuat satu pilihan. Bila kondisi yang diseleksi terpenuhi maka aksi atau *statement* yang mengikuti “then” yang akan diproses. Dan jika sebaliknya maka tidak diminta aksi apapun.



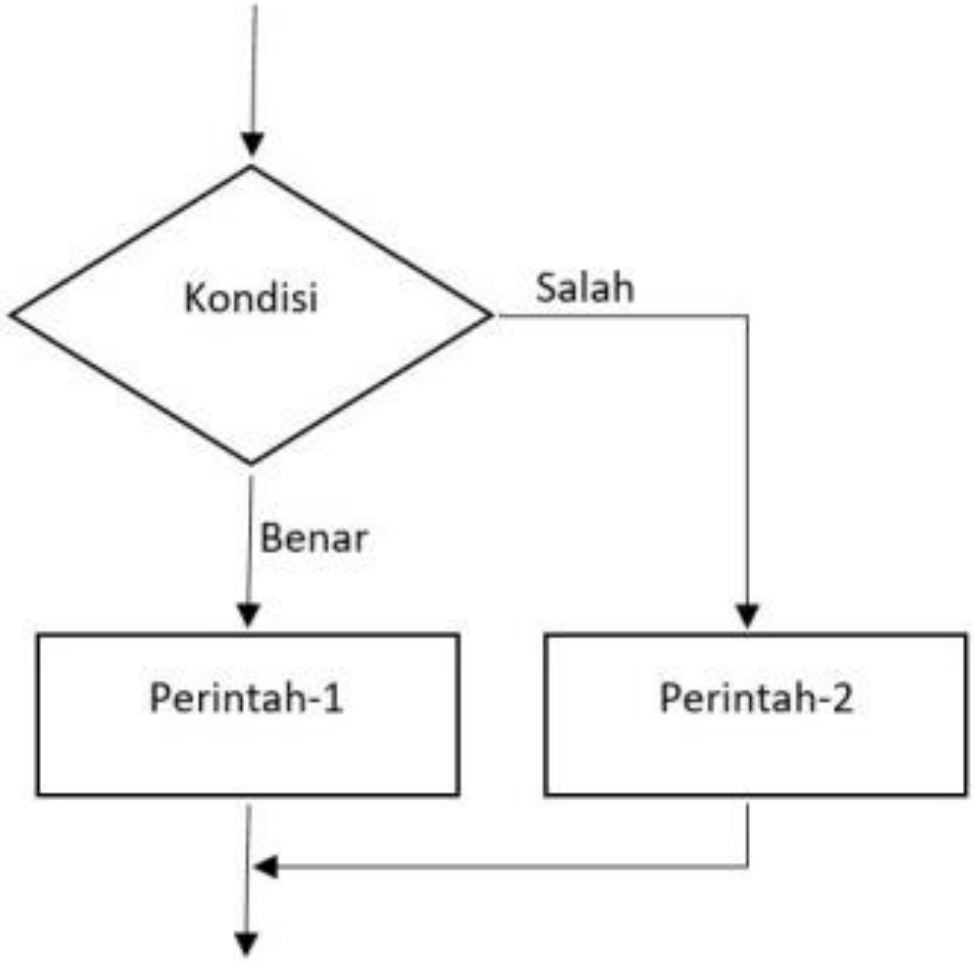
Contoh IF-ELSE

- Algoritma mengecek kriteria kelulusan dengan syarat Nilai Akhir >=78
  1. Mulai
  2. Masukan Nilai Akhir
  3. Cek apakah Nilai Akhir >=78
  4. Jika ya cetak "Lulus"
  5. Selesai



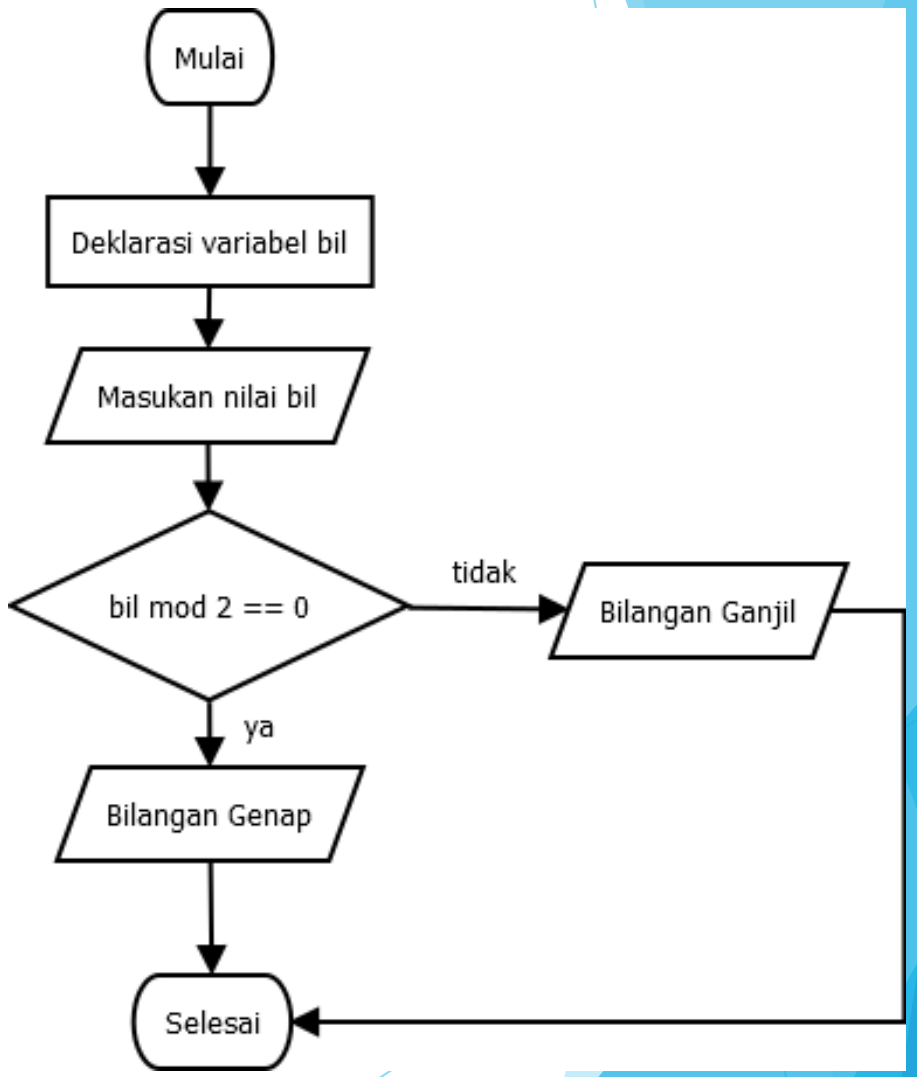
Struktur Pemilihan 2 Kasus (IF-THEN-ELSE)

- Perintah ini digunakan jika kasus yang dihadapi memuat 2 pilihan. Bila kondisi terpenuhi maka aksi atau statement 1 yang akan diproses, jika tidak maka statement 2 yang akandiproses.



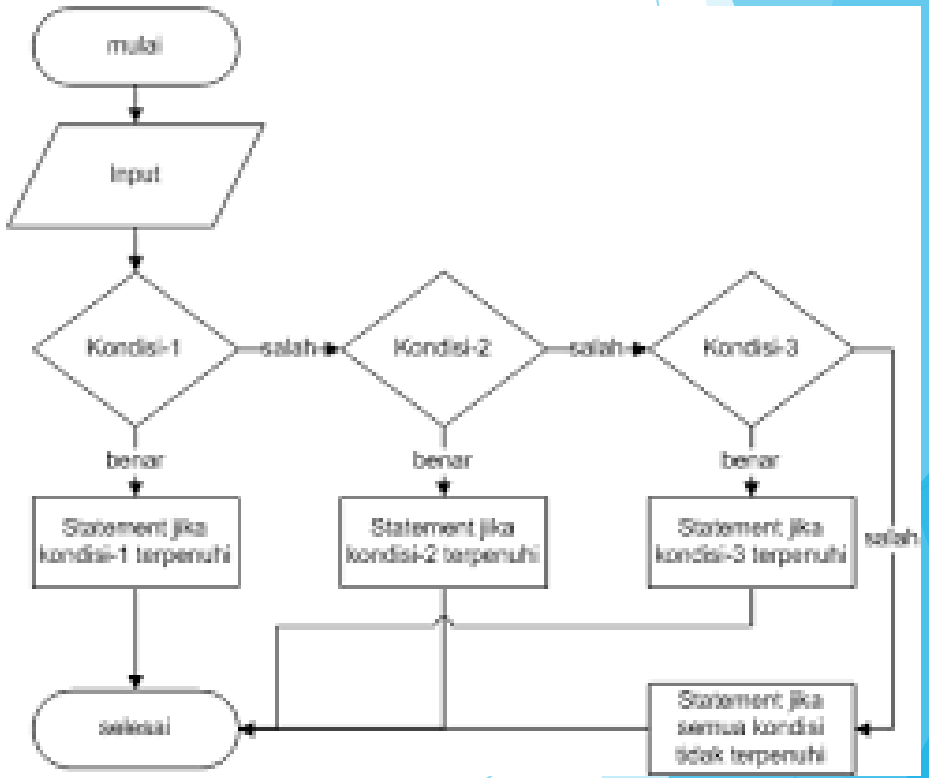
Contoh IF-ELSE-THEN

- Algoritma menentukan bilangan ganjil genap
  1. Mulai
  2. Deklarasikan variable Bil
  3. Masukan nilai Bil
  4. Periksa apakah Bil habis dibagi 2 ?
  5. Jika Ya, Tampilkan “Bilangan Genap”
  6. Jika Tidak, Tampilkan “Bilangan Ganjil”
  7. Selesai



Struktur Pemilihan 3 kasus atau lebih (IF-THEN-IF ELSE)

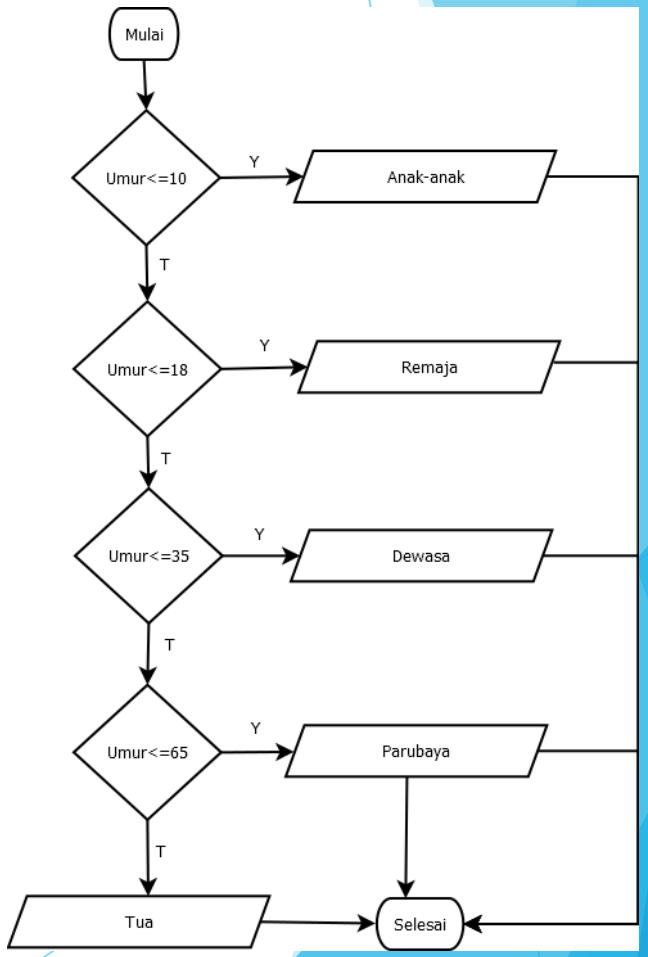
- Struktur pemilihan dengan tiga kasus atau lebih menggunakan perintah if…then…if else…then… artinya perintah ini digunakan jika masalah yang dihadapi memuat tiga pilihan atau lebih. Dapat juga
- disebut IF Bersarang atau IF Bertingkat. Dimana terdapat perintah IF di dalam IF (*nested if*).



Contoh Pemilihan 3 kasus atau lebih

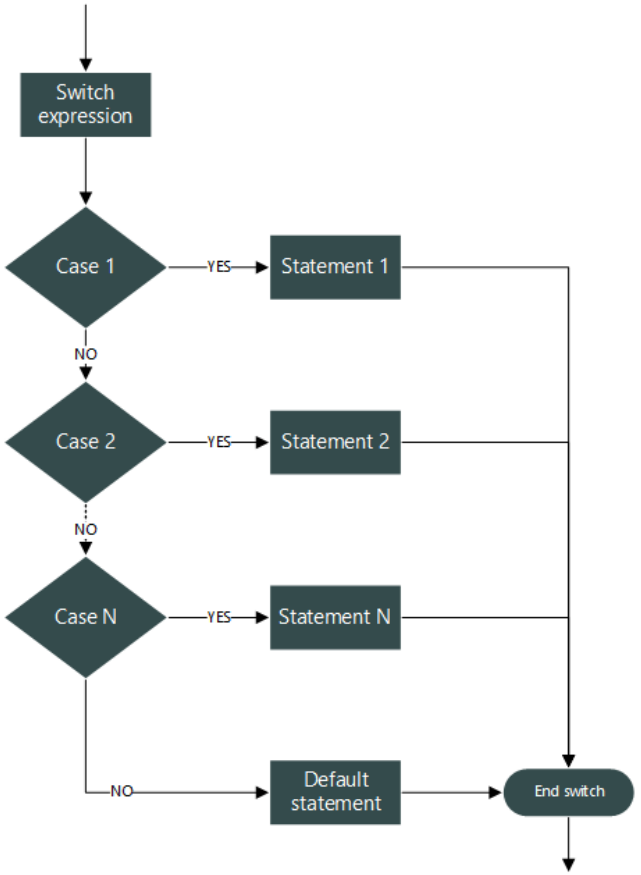
- Contoh Menentukan Apakah pengguna merupakan anak, remaja, dewasa, parubaya atau tua

1. Mulai
2. Input Umur
3. Cek Umur apakah sesuai kondisi 1?
4. Jika Ya maka Tampilkan Anak-anak
5. Jika tidak, maka lanjut ke kondisi berikutnya
6. .....
7. Selesai



Struktur Pemilihan (Switch-Case)

- Konstruksi CASE digunakan untuk masalah dengan dua kasus atau lebih. Konstruksi ini dapat menyederhanakan perintah if-then-else yang bertingkat-tingkat ataupun bersarang untuk kasus tertentu.
- Perintah ini digunakan untuk memilih suatu proses dari beberapa kemungkinan proses berdasarkan nilai dari kondisi variabel kontrol.



Perbedaan Penggunaan Switch-Case dan If-Else

If-Else

```
#include <iostream>
 using namespace std;
 int main()
{
  char nilai;
 cout << "Input Nilai Anda (A - E): ";
  cin >> nilai;
 if (nilai == 'A' ) {
    cout << "Pertahankan!" << endl;
  }
  else if (nilai == 'B' ) {
    cout << "Harus lebih baik lagi" << endl;
  }
  else if (nilai == 'C' ) {
    cout << "Perbanyak belajar" << endl;
  }
  else if (nilai == 'D' ) {
    cout << "Jangan keseringan main" << endl;
  }
  else if (nilai == 'E' ) {
    cout << "Kebanyakan bolos..." << endl;
  }
  else {
    cout << "Maaf, format nilai tidak sesuai" << endl;
  }
 return 0;
}
```

Switch-Case

```
#include <iostream>
using namespace std;
int main()
{
char nilai;
cout << "Input Nilai Anda (A - E): ";
cin >> nilai;
switch (nilai) {
  case 'A':
    cout << "Pertahankan!" << endl;
    break;
  case 'B':
    cout << "Harus lebih baik lagi" << endl;
    break;
  case 'C':
    cout << "Perbanyak belajar" << endl;
    break;
  case 'D':
    cout << "Jangan keseringan main" << endl;
    break;
  case 'E':
    cout << "Kebanyakan bolos..." << endl;
    break;
  default:
    cout << "Maaf, format nilai tidak sesuai" << endl;
  }
return 0;
}
```

Struktur Perulangan

Digunakan untuk mengulang statemen berulang kali sejumlah yang ditentukan

Format perulangan for

```
for (start; condition; increment)
{
    statement
}
```

- **Start**

  kondisi pada saat awal perulangan

- **Condition**

  kondisi yang harus dipenuhi agar perulangan dijalankan. Selama kondisi ini terpenuhi, maka C++ akan terus melakukan perulangan

- **Increment**

  bagian yang digunakan untuk memproses variabel agar bisa memenuhi kondisi akhir perulangan.

- **Statement**

  bagian kode program yang akan diproses secara terus-menerus selama proses perulangan berlangsung.

Struktur Perulangan

Digunakan untuk mengulang statemen berulang kali sejumlah yang ditentukan

Format perulangan for

```
for (start; condition; increment)
{
    statement
}
```

- **Start**

  kondisi pada saat awal perulangan

- **Condition**

  kondisi yang harus dipenuhi agar perulangan dijalankan. Selama kondisi ini terpenuhi, maka C++ akan terus melakukan perulangan

- **Increment**

  bagian yang digunakan untuk memproses variabel agar bisa memenuhi kondisi akhir perulangan.

- **Statement**

  bagian kode program yang akan diproses secara terus-menerus selama proses perulangan berlangsung.

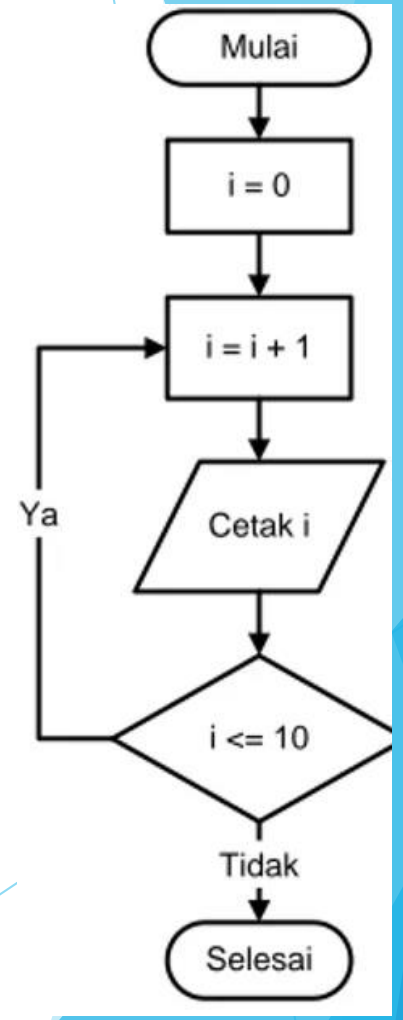
Contoh perulangan for

```
#include <iostream>
using namespace std;

void CetakAngka (int i, int j){
   for (i = 1; i <= j; i++){
      cout << i << '\n';
   }
}

int main ()
{
   CetakAngka (1,10);
}
```

- **Hasil keluaran :**
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10



Struktur while … do

- Dalam perulangan ***while***, program akan terus melakukan perulangan dengan mengeksekusi pernyataan target selama kondisi tertentu bernilai benar
- Format perulangan while…do

```
while (condition)
  {
    statement
  }
```

- **Condition**
  - kondisi yang harus dipenuhi agar perulangan berlangsung
  - Kondisi ini mirip seperti dalam perulangan for.
  - Condition ini akan diperiksa pada tiap perulangan, dan hanya jika hasilnya FALSE, maka proses perulangan berhenti. Artinya kita tidak tahu berapa banyaknya perulangan.
  - Karena, selama condition bernilai TRUE, maka perulangan akan terus dilakukan.
- **Statement**

  Kode program yang akan diulang

Contoh perulangan while

```
#include <iostream>
using namespace std;

int main () {
  // Local variable declaration:
  int a = 1;

  // while loop execution
  while( a < 10 ) {
    cout << "value of a: " << a << endl;
    a++;
  }
return 0;
}
```

- **Hasil keluaran :**
- Value of a : 1
- Value of a : 2
- Value of a : 3
- Value of a : 4
- Value of a : 5
- Value of a : 6
- Value of a : 7
- Value of a : 8
- Value of a : 9

Perulangan do … while

- Perulangan while hampir sama dengan do…while
- Format perulangan Do … While

```
do
{
   statement;
}
While (condition);
```

- Perbedaan terletak pada ’lokasi’ pengecekan kondisi perulangan.
- Dalam struktur while, pengecekan untuk kondisi perulangan di lakukan di awal, sehingga jika kondisi tidak terpenuhi, maka perulangan tidak akan pernah dijalankan.
- Namun pada perulangan do-while:
- Pengecekan kondisi akan dilakukan di akhir perulangan, sehingga walaupun kondisi adalah FALSE, perulangan akan tetap berjalan minimal 1 kali.
- **Statement** adalah kode program yang akan diulang & condition adalah kondisi yang harus dipenuhi agar perulangan berlangsung.

Contoh perulangan do...while

```
#include <iostream>
using namespace std;

int main () {
  // Local variable declaration:
  int a = 1;

  // while loop execution
  do {
    cout << "value of a: " << a << endl;
    a = a + 1;
  } while ( a < 10 );

  return 0;
}
```

- **Hasil keluaran :**
- Value of a : 1
- Value of a : 2
- Value of a : 3
- Value of a : 4
- Value of a : 5
- Value of a : 6
- Value of a : 7
- Value of a : 8
- Value of a : 9

TERIMA KASIH