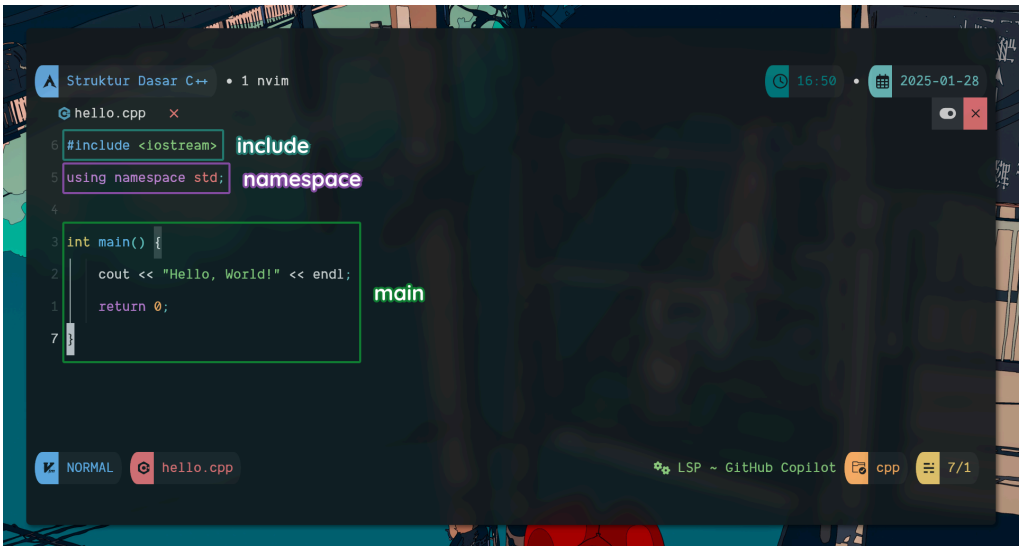
Modul Praktikum

Alpro Lanjut

1

PENGENALAN BAHASA PEMROGRAMAN C++

STRUKTUR DASAR C++



- **Include** : Bagian ini isinya adalah library apa aja yang kita akan gunain dalam program kita. Library ini bisa berupa library bawaan dari C++ atau library yang kita buat sendiri. Kalau di Python ini mirip-mirip sama import.
- **namespace** : Sifatnya opsional sih, tapi biasanya kita pake. Contoh di atas, kita pake std. Karena fungsi-fungsi yang ada di iostream dan string berada di dalam namespace std. Jadi, kalau kita mau pake fungsi-fungsi itu, kita harus nulis std:: dulu.

```
std::cout << "Hello, World!" << std::endl;
```

  Jadinya seperti ini:Pegelkan ngetik std:: terus?, makanya kita gunain namespace.
- **main()** : Bagian ini adalah bagian yang paling penting. Di dalamnya kita tulis semua kode program kita. Program kita akan dijalankan dari sini. Ini wajib ada di setiap program C++.
- { dan } : tempat dimulainya satu blok program.
- **Comment** : Comment adalah bagian dari kode program yang tidak akan dijalankan oleh compiler.
- Ada dua jenis Comment di C++, yaitu :
  - Single Line Comment : ‘ // ‘ berlaku untuk satu baris kode

- Multi Line Comment : ‘ /* ‘ dan ‘ */ ‘ Comment ini berlaku untuk beberapa baris kode

INPUT DAN OUTPUT

A. Output

Output adalah informasi yang udah kita proses dan kita tampilkan ke layar. Di C++, kita bisa menampilkan output ke layar dengan menggunakan ‘cout’.

Struktur fungsi ‘ cout ’ :

Setelah simbol ‘ << ‘, kita bisa menambahkan data yang mau kita tampilkan. Data ini bisa berupa string, angka, atau variabel.

Contoh:

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

int main() {
    cout << "Hello, World! ";
    cout << "Belajar C++ sangat seru😀";
    return 0;
}
```

Maka outputnya akan seperti ini:

```
Hello, World! Belajar C++ sangat seru😀
```

Kalau mau buat baris baru, kita bisa gunain endl.

Contoh:

```
cout << "Hello, World!" << endl;
cout << "Belajar C++ sangat seru😀";
```

Maka outputnya akan nampilin dua baris:

```
Hello, World!
Belajar C++ sangat seru😀
```

Nahh simbol << ini bisa kita sebut sebagai operator insertion atau operator masukan. Karena dia nge-insert data ke dalam cout. Kalau di kasus kita di atas, dia nge-insert string ke dalam cout. Kita juga bisa nampilin angka atau variabel.

Contoh:

```
string nama = "Alvito";
int umur = 20;
cout << "Halo, nama saya " << nama << " dan umur saya " << umur << "
tahun";
```

Maka outputnya akan seperti ini:

```
Halo, nama saya Alvito dan umur saya 20 tahun
```

B. Input

Input adalah data yang kita terima dari user. Di C++, kita bisa menerima input dari user dengan menggunakan 'cin'.

Struktur fungsi 'cin' :

Setelah simbol >>, kita bisa menambahkan variabel yang akan kita isi dengan data yang diinputkan oleh user.

Contoh:

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

int main() {
    string nama;
    cout << "Masukkan nama kamu: ";
    cin >> nama;
    cout << "Halo, " << nama;
    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nama kamu: Raana
Halo, Raana
```

Tapi, fungsi cin ini hanya nerima inputan sampai ketemu spasi. Jadi, kalau kita inputkan Raana Matcha, dia cuma nerima Raana aja. Kalau kita mau nerima inputan sampai baris baru, kita bisa gunain fungsi getline.

Contoh:

```
string nama;
cout << "Masukkan nama kamu: ";
getline(cin, nama);
cout << "Halo, " << nama;
```

Maka outputnya akan seperti ini:

```
Masukkan nama kamu: Raana Matcha
Halo, Raana Matcha
```

Catatan: Perhatikan simbol << itu untuk output dan >> itu untuk input.

TIPE DATA

Tipe data adalah jenis data yang bisa disimpan di dalam variabel. Di C++, ada beberapa tipe data yang bisa kita gunain. Pada umumnya bahasa pemrograman C++, terdapat tiga jenis tipe data, yaitu :

A. Tipe data primitif

Tipe data primitif adalah tipe data dasar yang digunakan untuk operasi dasar. Contohnya int, boolean, float, double, char, dll. Masing masing tipe data mempunyai karakteristik dan *range* yang berbeda.

Tipe Data	Ukuran (byte)	Nilai
int	4	9
float	4	9.9
double	8	9.9
char	1	a'
bool	1	true/false
string	-	"Hello, World!"

Contoh:

```
int umur = 20;
int sumbuY = -45;
float berat_badan = 50.5;
double tinggi_badan = 170.5;
char jenis_kelamin = 'L';
bool is_menikah = false;
```

Untuk mencari ukuran *byte* setiap tipe data kita dapat menggunakan operator **sizeof** diikuti dengan tipe data yang ingin di cari ukuran bytenya.

Berikut implementasi operator sizeof untuk menentukan jumlah *byte* dari setiap tipe data.

```
cout << "Size of char: " << sizeof(char) << endl;
cout << "Size of int: " << sizeof(int) << endl;
cout << "Size of float: " << sizeof(float) << endl;
cout << "Size of double: " << sizeof(double) << endl;
cout << "Size of long: " << sizeof(long);
```

Output:

```
Size of char: 1
Size of int: 4
```

```
Size of float: 4
Size of double: 8
Size of long: 8
```

B. Tipe data kolektif

Tipe data kolektif adalah tipe data yang berupa rangkaian atau kumpulan data yang memiliki indeks, seperti array atau list.

```
int angka[5] = {1, 2, 3, 4, 5};
string nama = "Raana";
```

C. Tipe data abstrak

Contoh dari tipe data abstrak adalah **struct**. Struct adalah sekumpulan variabel bertipe data abstrak yang dinyatakan dengan sebuah nama dan bisa diciptakan secara dinamis.

```
struct Mahasiswa {
    string nama;
    int umur;
    float ipk;
};

Mahasiswa mhs = {"Raana", 20, 3.5};
```

VARIABEL

Segala program komputer yang sedang berjalan akan nyimpen data sementara di dalam RAM (Random Access Memori). Data-data yang tersimpan dalam RAM punya alamat yang direpresentasikan dalam bentuk hexadecimal. Nah, biar gampang, kita kasih nama ke alamat-alamat ini. Nama ini kita sebut sebagai variabel.

A. Cara Buat Variabel

Untuk bikin variabel, kita harus tentuin tipe datanya dulu dan kasih nama variabelnya. Struktur penulisan variabel di C++:

```
tipe_data nama_variabel;
// atau
tipe_data nama_variabel = nilai;
```

Contoh:

```
string nama = "Raana";
int umur;
```

Nahh pada variabel pertama yaitu **nama** memiliki nilai awal yaitu **Raana**, sedangkan variabel kedua yaitu **umur** tetap memiliki nilai awal tetapi isinya **null** (kosong).

B. Aturan penulisan variabel

- Penulisan variabel harus diawali dengan huruf atau underscore '_', tidak boleh menggunakan angka atau simbol tertentu seperti $, ?, %, #, !, &, *, (,), -, +, =, dll.
- Nama variabel tidak boleh dimulai dengan angka.
- Nama variabel tidak boleh ada spasi.
- Nama variabel tidak boleh ada karakter spesial kecuali underscore.
- Variabel hanya terdiri dari satu kata (tidak boleh lebih dari satu kata), jika terdapat dua kata sebaiknya gunakan penghubung underscore (_).
- Penulisan variabel dalam c++ bersifat *case sensitive* artinya bahwa huruf besar kecil dibedakan. Contohnya a dan A adalah dua variabel yang berbeda.
- Nama variabel tidak boleh mengandung *keyword* atau kata kunci tertentu yang digunakan pada C++ seperti *for, if, break, continue*, dll

OPERATOR

Operator aritmatika digunakan untuk melakukan operasi matematika.

A. Operator Aritmatika

Penjumlahan	+
Pengurangan	-
Perkalian	*
Pembagian	/
Sisa bagi / Modulus	%

increment	++
decrement	--

Contoh

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

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

    cout << "Masukkan nilai a: ";
    cin >> a;
    cout << "Masukkan nilai b: ";
    cin >> b;

    // Penjumlahan
    hasil = a + b;
    cout << "Hasil penjumlahan: " << hasil << endl;

    // Pengurangan
    hasil = a - b;
    cout << "Hasil pengurangan: " << hasil << endl;

    // Perkalian
    hasil = a * b;
    cout << "Hasil perkalian: " << hasil << endl;

    // Pembagian
    hasil = a / b;
    cout << "Hasil pembagian: " << hasil << endl;

    // Modulus
    hasil = a % b;
    cout << "Hasil modulus: " << hasil << endl;

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nilai a: 4
```

```
Masukkan nilai b: 2
Hasil penjumlahan: 6
Hasil pengurangan: 2
Hasil perkalian: 8
Hasil pembagian: 2
Hasil modulus: 0
```

B. Operator Penugasan

Atau disebut juga *Assignment Operator* berfungsi untuk memberikan tugas pada variabel atau untuk mengisikan nilai pada variabel tersebut.

Penugasan	=
Penambahan	+=
pengurangan	-=
Perkalian	*=
Pembagian	/=
Sisa bagi	%=
Pergeseran bit ke kiri	>>=
Pergeseran bit ke kanan	<<=
Bitwise AND	&=
Bitwise OR Eksklusif	^=
Bitwise OR Inklusif	|=

Contoh

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

int main() {
    int a, b;

    cout << "Masukkan nilai a: ";
    cin >> a;
    cout << "Masukkan nilai b: ";
    cin >> b;

    // Penjumlahan
    a += b;
    cout << "Hasil penjumlahan: " << a << endl;

    // Pengurangan
    a -= b;
    cout << "Hasil pengurangan: " << a << endl;

    // Perkalian
    a *= b;
    cout << "Hasil perkalian: " << a << endl;

    // Pembagian
    a /= b;
    cout << "Hasil pembagian: " << a << endl;

    // Modulus
    a %= b;
    cout << "Hasil modulus: " << a << endl;

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nilai a: 4
Masukkan nilai b: 2
Hasil penjumlahan: 6
Hasil pengurangan: 4
```

```
Hasil perkalian: 8
Hasil pembagian: 4
Hasil modulus: 0
```

C. Operator Perbandingan

Operator yang digunakan untuk membandingkan dua *operand*, kemudian menghasilkan nilai *boolean*, benar (1) dan juga nilai salah (0). Operator ini sering digunakan dalam struktur kontrol seperti if, else, dan switch.

Sama dengan	==
Tidak sama dengan	!=
Kurang dari	<
Lebih besar dari	>
Kurang dari sama dengan	<=
Lebih dari sama dengan	>=

Contoh

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

int main() {
    int a, b;

    cout << "Masukkan nilai a: ";
    cin >> a;
    cout << "Masukkan nilai b: ";
    cin >> b;

    // Sama dengan
    cout << "Apakah a sama dengan b? " << (a == b) << endl;
```

```
    // Tidak sama dengan
    cout << "Apakah a tidak sama dengan b? " << (a != b) << endl;

    // Lebih besar dari
    cout << "Apakah a lebih besar dari b? " << (a > b) << endl;

    // Lebih kecil dari
    cout << "Apakah a lebih kecil dari b? " << (a < b) << endl;

    // Lebih besar sama dengan
    cout << "Apakah a lebih besar sama dengan b? " << (a >= b) << endl;

    // Lebih kecil sama dengan
    cout << "Apakah a lebih kecil sama dengan b? " << (a <= b) << endl;

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nilai a: 4
Masukkan nilai b: 2
Apakah a sama dengan b? 0
Apakah a tidak sama dengan b? 1
Apakah a lebih besar dari b? 1
Apakah a lebih kecil dari b? 0
Apakah a lebih besar sama dengan b? 1
Apakah a lebih kecil sama dengan b? 0
```

D. Operator Logika

Operator logika digunakan untuk menggabungkan dua atau lebih ekspresi logika. Operator ini menghasilkan nilai true atau false. Operator ini sering digunakan dalam struktur kontrol seperti if, else, dan switch.

AND	&&
OR	||
NOT	!

Contoh

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

int main() {
    int a, b;

    cout << "Masukkan nilai a: ";
    cin >> a;
    cout << "Masukkan nilai b: ";
    cin >> b;

    // Dan
    cout << "Apakah a lebih besar dari 0 dan b lebih kecil dari 10? " <<
(a > 0 && b < 10) << endl;

    // Atau
    cout << "Apakah a lebih besar dari 0 atau b lebih kecil dari 10? " <<
(a > 0 || b < 10) << endl;

    // Tidak
    cout << "Apakah a lebih besar dari 0? " << !a << endl;

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nilai a: 4
Masukkan nilai b: 2
Apakah a lebih besar dari 0 dan b lebih kecil dari 10? 1
Apakah a lebih besar dari 0 atau b lebih kecil dari 10? 1
Apakah a lebih besar dari 0? 0
```

E. Operator Bitwise

Operator bitwise adalah operasi matematika yang mengoperasikan pada bilangan biner berbasis 2.

AND	&

OR	|
XOR	^
NOT	~
Pergeseran bit ke kiri	<<
Pergeseran bit ke kanan	>>

F. Operator Ternary

Operator ternary adalah operator yang memungkinkan kita untuk menulis kode if-else dalam satu baris. Operator ini terdiri dari tiga bagian, yaitu kondisi, ekspresi jika benar, dan ekspresi jika salah.

Struktur operator ternary

```
kondisi ? ekspresi_jika_benar : ekspresi_jika_salah
```

Contoh:

```
bool sudahMenikah = true;
string status = sudahMenikah ? "Sudah menikah" : "Belum menikah";
cout << status << endl;
```

Maka outputnya akan seperti ini:

```
Sudah menikah
```

G. Operator Increment dan Decrement

Operator increment dan decrement digunakan untuk menambah atau mengurangi nilai variabel sebanyak satu. Operator ini terdiri dari:

Nama Operator	Operator	Bentuk Lain	Contoh
Increment	++	a++	a++ atau ++a
Decrement	--	a--	a-- atau --a

Perhatikan increment a++ dan ++a berbeda ya. a++ akan menampilkan nilai a terlebih dahulu, baru nilai a ditambah satu. Sedangkan ++a akan menambahkan nilai a terlebih dahulu, baru nilai a ditampilkan. Begitu juga untuk decrement.

Contoh

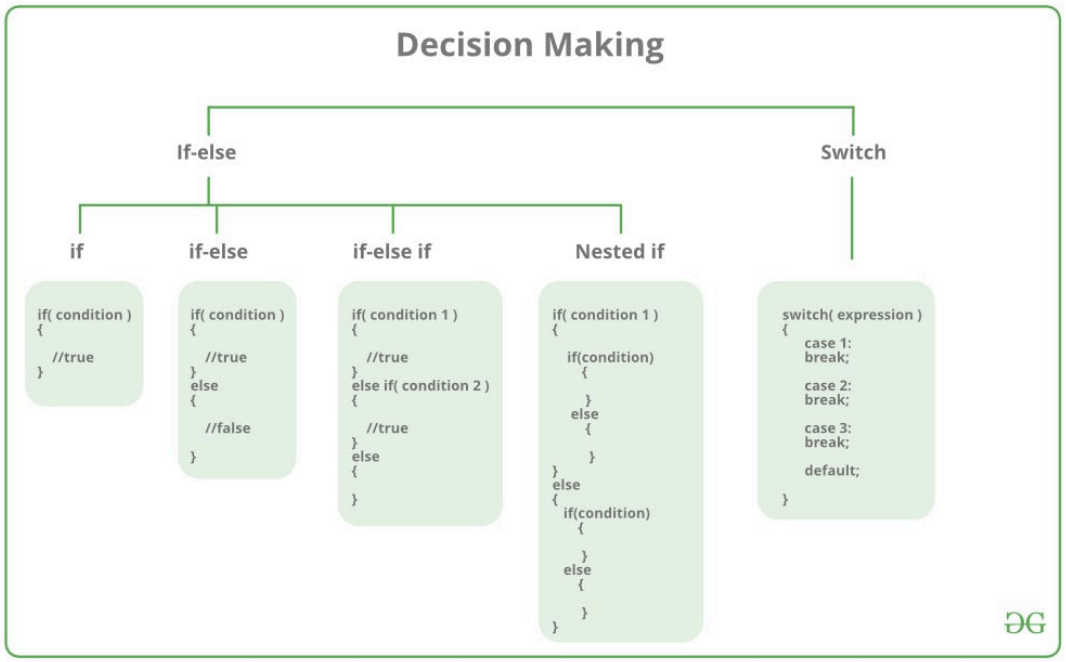
```
int z = 1;

cout << "Sebelum increment " << z << endl;
cout << "Increment di depan " << ++z << ", sudah bertambah" << endl;
cout << "Increment di belakang " << z++ << ", belum bertambah" << endl;
cout << "Hasil akhir " << z << endl;
```

Maka Outputnya akan seperti ini :

```
Sebelum increment 1
Increment di depan 2, sudah bertambah
Increment di belakang 2, belum bertambah
Hasil akhir 3
```

PERCABANGAN

Percabangan adalah sebuah struktur kontrol yang memungkinkan kita untuk memeriksa kondisi tertentu dan menjalankan kode tertentu berdasarkan apakah kondisi tersebut benar atau salah.



A. Percabangan If

Percabangan if adalah percabangan yang cuman punya satu blok pilihan ketika kondisinya terpenuhi atau bernilai benar. Contoh dalam kehidupan sehari-hari, jika saya malas, saya scroll fesbuk.

Struktur penulisan percabangan *if* di C++:

```
if (kondisi) {
    // kode yang akan dijalankan jika kondisi bernilai benar
}
```

Contoh:

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

int main() {
    bool malas = true;

    if (malas) {
        cout << "Saya scroll fesbuk";
    }

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Saya scroll fesbuk
```

B. Percabangan If ... else

Percabangan if else adalah percabangan yang punya dua blok pilihan. Satu blok pilihan ketika kondisinya terpenuhi atau bernilai benar, dan satu blok pilihan ketika kondisinya tidak terpenuhi atau bernilai salah. Contoh dalam kehidupan sehari-hari, jika saya malas, saya scroll fesbuk. Jika tidak, saya belajar. Nahh gitu.

Struktur penulisan percabangan *if else* di C++:

```
if (kondisi) {
    // kode yang akan dijalankan jika kondisi bernilai benar
} else {
    // kode yang akan dijalankan jika kondisi bernilai salah
}
```

Contoh:

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

int main() {
    bool malas = false;

    if (malas) {
        cout << "Saya scroll fesbuk";
    } else {
        cout << "Saya belajar";
    }
    return 0;
}
```

Maka outputnya akan seperti ini:

```
Saya belajar
```

C. Else if

Percabangan else if adalah percabangan yang punya lebih dari dua blok pilihan. Kita bisa menambahkan blok pilihan baru ketika kondisi sebelumnya tidak terpenuhi. Contoh dalam kehidupan sehari-hari, jika saya malas, saya scroll fesbuk. Jika tidak, saya belajar. Jika saya lapar, saya makan. Nahh gitu.

Struktur penulisan percabangan *else if* di C++:

```
if (kondisi1) {
    // kode yang akan dijalankan jika kondisi1 bernilai benar
} else if (kondisi2) {
    // kode yang akan dijalankan jika kondisi2 bernilai benar
} else {
    // kode yang akan dijalankan jika semua kondisi salah
}
```

Contoh

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

int main() {
    bool malas = false;
    bool lapar = true;

    if (malas) {
        cout << "Saya scroll fesbuk";
    } else if (lapar) {
        cout << "Saya makan";
    } else {
        cout << "Saya belajar";
    }

    return 0;
}
```

Maka outputnya akan seperti ini:

```
Saya makan
```

D. Switch Case

Percabangan switch/case adalah bentuk lain dari percabangan if/else/if.

Struktur penulisan switch case:

```
switch(variabel) {
    case<value>:
        // blok kode
        break;
```

```
    case<value>:
        // blok kode
        break;
    default:
        // blok kode
}
```

Untuk bagian case, kita bisa buat sebanyak mungkin sesuai dengan kebutuhan. Jika kita tidak menemukan nilai yang sesuai, kita bisa gunakan default. Setiap case harus diakhiri dengan break. Tujuannya agar berhenti ngecek case selanjutnya ketika case saat ini sudah terpenuhi. Untuk default tak perlu diakhiri break.

Contoh:

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

int main() {
    int nilaiUjian;

    cout << "Masukkan nilai: ";
    cin >> nilaiUjian;

    switch(nilaiUjian) {
        case 100:
            cout << "Nilai sempurna";
            break;
        case 90:
            cout << "Nilai sangat baik";
            break;
        case 80:
            cout << "Nilai baik";
            break;
        case 70:
            cout << "Nilai cukup";
            break;
        case 60:
            cout << "Nilai kurang";
            break;
        default:
            cout << "Nilai tidak valid";
    }
```

```
    return 0;
}
```

Maka outputnya akan seperti ini:

```
Masukkan nilai: 90
Nilai sangat baik
```

E. Nested If

Nested if adalah percabangan if yang ada di dalam if.

Contoh :

```
bool malas = true;
bool lapar = true;

if (malas) {
    if (lapar) {
        cout << "Saya makan";
    }
} else {
    cout << "Gak ngapa-ngapain";
}
```

Maka outputnya akan seperti ini:

```
Saya makan
```

F. Ternary Operator

Ternary operator ini nulis if-else dalam satu baris.

Struktur penulisan ternary operator:

```
(kondisi) ? nilaiJikaBenar : nilaiJikaSalah;
```

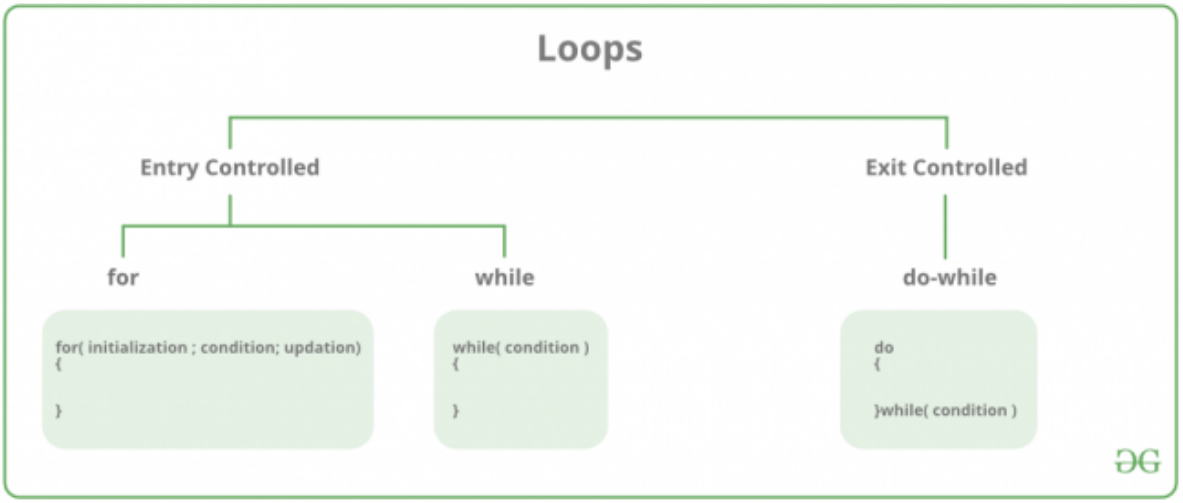
Contoh:

```
bool malas = true;
string aktivitas = (malas) ? "Scroll fesbuk" : "Belajar";
cout << aktivitas;
```

Maka outputnya akan seperti ini:

```
Scroll fesbuk
```

PERULANGAN



A. For

Perulangan *for* merupakan perulangan yang sudah ditentukan dan jelas berapa kali ia akan mengulang.

Struktur dari for loop adalah sebagai berikut:

```
for (initialization; condition; update) {
    // kode yang akan diulang
}
```

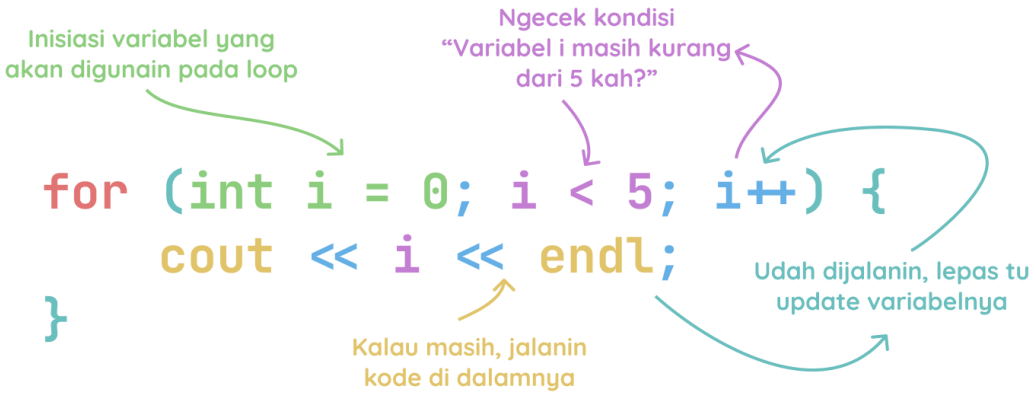
- Initialization adalah bagian yang akan dieksekusi pertama kali ketika loop dimulai. Biasanya kita gunain tuk deklarasikan variabel yang akan digunakan dalam loop.
- Condition adalah bagian yang akan dievaluasi sebelum setiap iterasi loop. Jika hasil evaluasi ini adalah true, maka loop lanjut. Jika false, maka loop udahan.
- Update adalah bagian yang akan dieksekusi setelah setiap iterasi loop. Biasanya kita gunain tuk update variabel yang digunakan dalam loop.

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

int main() {
    for (int i = 0; i < 5; i++) {
        cout << i << endl;
    }
    return 0;
}
```

Output:

```
0
1
2
3
4
```

Ilustrasi for loop:



B. While

Perulangan *while* merupakan perulangan *uncounted loop* atau tidak ditentukan berapa kali mengulangnya. Perulangan ini akan terus dilakukan selama kondisinya terpenuhi.

Struktur dari for loop adalah sebagai berikut:

```
while (condition) {
    // kode yang akan diulang
}
```

Condition adalah bagian yang akan dievaluasi sebelum setiap iterasi loop. Jika hasil evaluasi ini adalah true, maka loop lanjut. Jika false, maka loop udahan.

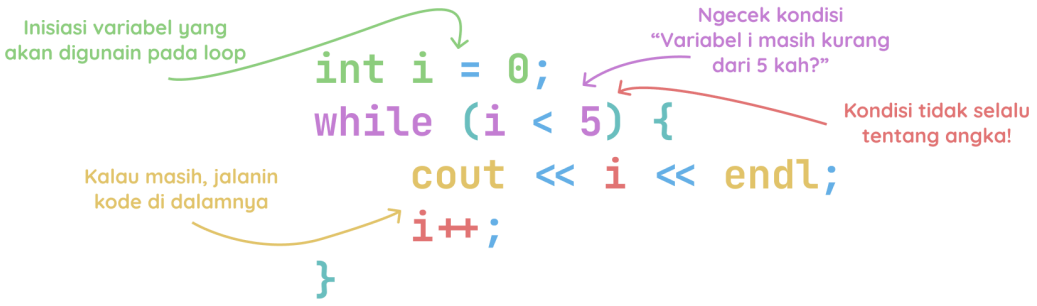
Contoh:

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

int main() {
    int i = 0;
    while (i < 5) {
        cout << i << endl;
        i++;
    }
    return 0;
}
```

Output :

```
0
1
2
3
4
```

Ilustrasi while loop:



C. Do-While

Mirip seperti *while*, perbedaannya terdapat pada perintah *do* di awal yang berguna untuk mengeksekusi perintah satu kali terlebih dahulu, kemudian memeriksa kondisi perulangan, apabila benar perintah akan dieksekusi lagi dan jika salah maka perulangan akan berhenti.

Struktur dari do-while loop adalah sebagai berikut:

```
do {
    // kode yang akan diulang
} while (condition);
```

Condition adalah bagian yang akan dievaluasi setelah setiap iterasi loop. Jika hasil evaluasi ini adalah true, maka loop lanjut. Jika false, maka loop udahan.

Contoh:

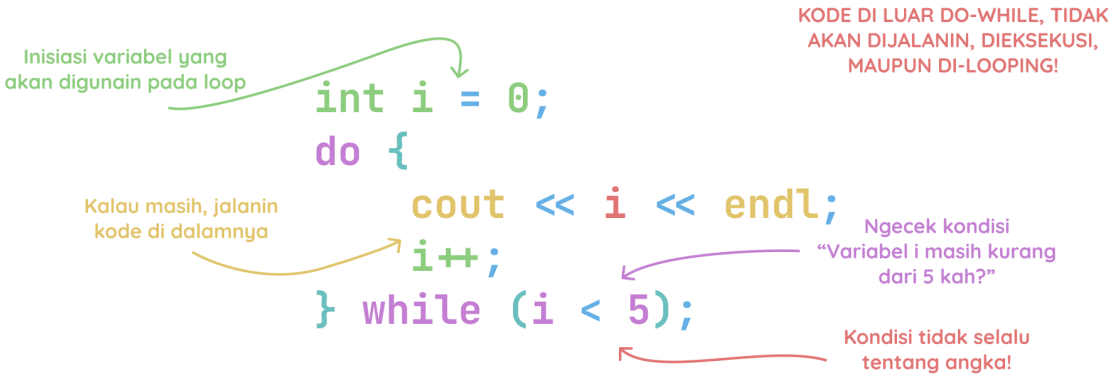
```
#include <iostream>
using namespace std;

int main() {
    int i = 0;
    do {
        cout << i << endl;
        i++;
    } while (i < 5);
    return 0;
}
```

Output

```
0
1
2
3
4
```

Ilustrasi do-while loop:

.



D. Perulangan Foreach

Foreach-loop ini adalah loop yang digunakan untuk mengakses elemen-elemen dari array atau collection.

Struktur dari Foreach loop adalah sebagai berikut:

```
for (tipe_data variabel : array) {
    // kode yang akan diulang
}
```

- Tipe Data adalah tipe data dari elemen array.
- Variabel adalah variabel yang akan nyimpen nilai bijian dari elemen array.
- Array adalah array yang akan di-looping.

Contoh :

```
#include <iostream>
using namespace std;
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    for (int x : arr) {
        cout << x << endl;
    }
    return 0;
}
```

Output :

```
1
2
3
4
5
```

FLOWCHART

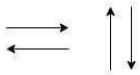
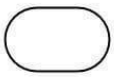
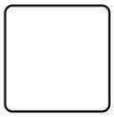
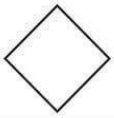
A. Flowchart

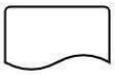
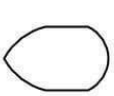
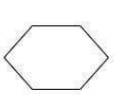
Flowchart atau bagan alur adalah diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah digambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah.

Fungsi utama dari *flowchart* adalah memberi gambaran jalannya sebuah program dari satu proses ke proses lainnya. Sehingga, alur program menjadi mudah dipahami oleh semua orang. Selain itu, fungsi lain dari *flowchart* adalah untuk menyederhanakan rangkaian prosedur agar memudahkan pemahaman terhadap informasi tersebut.

Yang perlu diperhatikan dalam membuat *flowchart*:

- *Flowchart* dibuat dari atas ke bawah dimulai dari bagian kiri suatu halaman.
- Kegiatan dalam *flowchart* harus ditunjukkan dengan jelas.
- Kegiatan dalam *flowchart* harus jelas di mana akan dimulai dan di mana akan berakhir.
- Kegiatan yang ada dalam *flowchart* digunakan kata yang mewakili pekerjaan.
- Kegiatan dalam *flowchart* harus sesuai dengan urutannya.
- Kegiatan yang terpotong dihubungkan dengan simbol penghubung.

	**Flow** Simbol yang digunakan untuk menggabungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga dengan Connecting Line.
	**On-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang sama.
	**Off-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang berbeda.
	**Terminator** Simbol yang menyatakan awal atau akhir suatu program.
	**Process** Simbol yang menyatakan suatu proses yang dilakukan komputer.
	**Decision** Simbol yang menunjukan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, yaitu ya dan tidak.

	**Input/output** Simbol yang menyatakan proses input atau output tanpa tergantung peralatan.
	**Manual Operation** Simbol yang menyatakan suatu proses yang tidak dilakukan oleh komputer.
	**Document** Simbol yang menyatakan bahwa input berasal dari dokumen dalam bentuk fisik, atau output yang perlu dicetak.
	**Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program) atau prosedure.
	**Display** Simbol yang menyatakan peralatan output yang digunakan.
	**Preparation** Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal.

B. Contoh Flowchart

STUDI KASUS

1. Buatlah flowchart, pseudocode dan program yang meminta input waktu dalam detik, lalu mengonversinya ke dalam format jam:menit:detik.!
2. Buatlah flowchart, pseudocode dan program Buatlah program yang meminta berat badan (kg) dan tinggi badan (m) dari pengguna, lalu menghitung dan menentukan kategori BMI mereka.