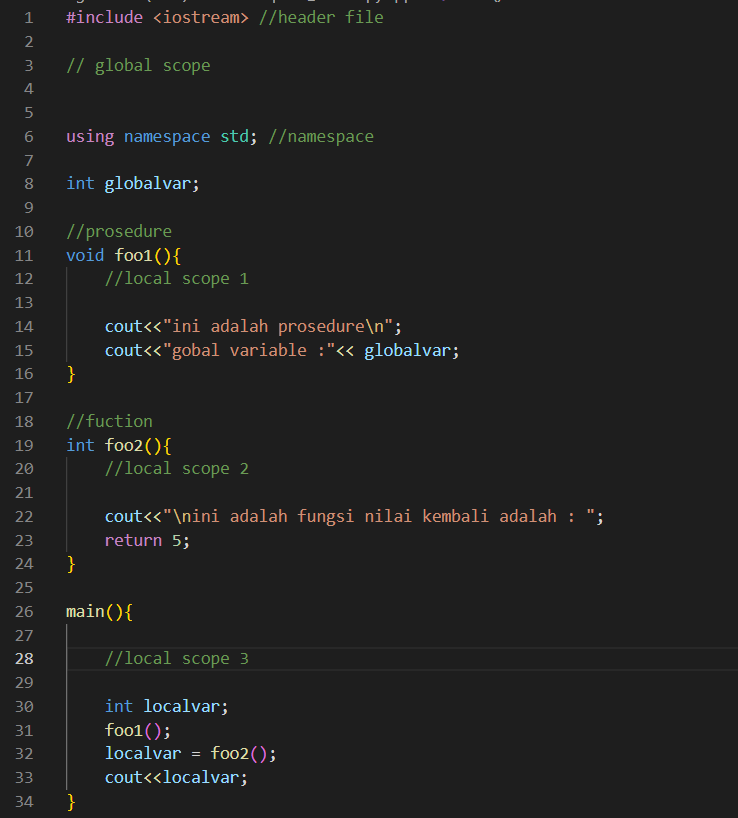
**Template Dasar Pemrograman C++**

Materi kali ini hanya memberikan template dan penjelasan dasar tentang Bahasa pemrogramman c++, untuk pemahaman lebih lanjut anda bisa membaca referensi dibawah ini atau dari luar :

* <https://www.youtube.com/watch?v=18c3MTX0PK0&list=PLlrATfBNZ98dudnM48yfGUldqGD0S4FFb>
* <https://www.w3schools.com/cpp/default.asp>
* https://app.programiz.pro/course/learn-cpp-basics/get-started?page=1
* <https://cplusplus.com/doc/tutorial/program_structure/>
* <https://www.ibm.com/docs/en/i/7.3?topic=reference-about-ile-cc-language>

1. **Syntax Dasar C++**

Berikut adalah syntax dasar c++ :



* **Header file**

#include dalah sebuah keyword untuk memasukan header file/ library yang tersedia. Header file adalah sebuah file yang berisi kumpulan fuction, procedure atau object yang dapat kita panggil ke file utama. C++ memiliki banyak header file build in dan setiap library memiliki kegunaannya masing masing (<https://en.cppreference.com/w/cpp/header>). Selain header file build in anda juga dapat membuat header file sendiri berdasarkan kebutuhan program anda. Untuk menambahkan header file build in biasanya ditulis seperti berikut : #include <NAMA\_HEADER>, sedangkan header file yang dibuat sendiri ditulis: #include “DIRECTORY/NAMA\_HEADER” tergantung dimana header tersebut disimpan.

* **Scope**

Terdapat 2 macam scope yaitu local scope dan global scope. Perbedaan yang mencolok antara keduanya adalah ketika anda mengakses sebuah variable yang terdapat di kedua scope tersebut. Variable yang berada di global dapat diakses dari scope mana saja sedangkan scope local hanya dapat diakses pada scope tersebut.

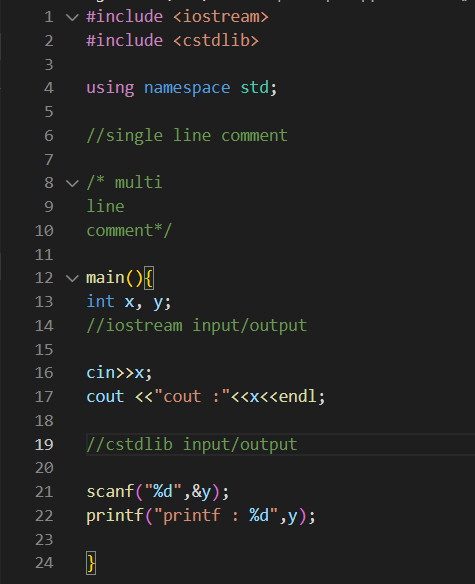
* **Namespace**

adalah sebuah keyword yang berguna untuk membuat space sendiri untuk variabel, function atau object tertentu. dalam C++ sudah terdapat namespace build in yaitu namespace std. using namespace std simplenya anda menggunakan space std setiap kali memanggil, declare sebuah fungsi atau variable. jika using namespace tidak disertakan maka setiap kali anda memanggil function seperti cout maka namespace harus ditulis bersamaan dengan fuction cout jadi seperti ini: "std::cout<<"hello";".

* **Main()**

Adalah function utama dalam code anda. Function ini berisikan algoritma/ workflow code /program yang anda buat.

1. **Output, Input & Comments**



* **INPUT/OUTPUT**

Untuk input output anda dapat menggunakan salah satu dari 2 library diatas.

**Iostream**

Untuk input dapat menggunakan fungsi **cin>>VARIABLE;** sedangkan output menggunakan fungsi **cout<<VARIABLE/”STRING”;**

**Cstdlib**

Untuk input dapat menggunakan **scanf(FORMAT\_INPUT,REFERENCE\_TO\_VARIABLE,…);** sedangkan output dapat menggunakan **printf(“FORMAT\_OUTPUT”,VARIABLE\_OUTPUT);**

* **Comments**

Gunakan comment pada code anda untuk memberi note. Comment tidak akan dikompile oleh compiler.

**//** untuk comment single line

**/\* … \*/** untuk multiple line comment

1. **Variable & Data Types**

Variable adalah wadah untuk menyimpan sebuah value, ketika anda mendeklarasikan variable program akan mengalokasikan beberapa byte dalam RAM/cache device anda. Besar kecilnya alokasi memory tersebut tergantung pada data type yang digunakan. Data type juga dapat merepresentasikan bentuk value baik binner, character, ataupun nilai floating.

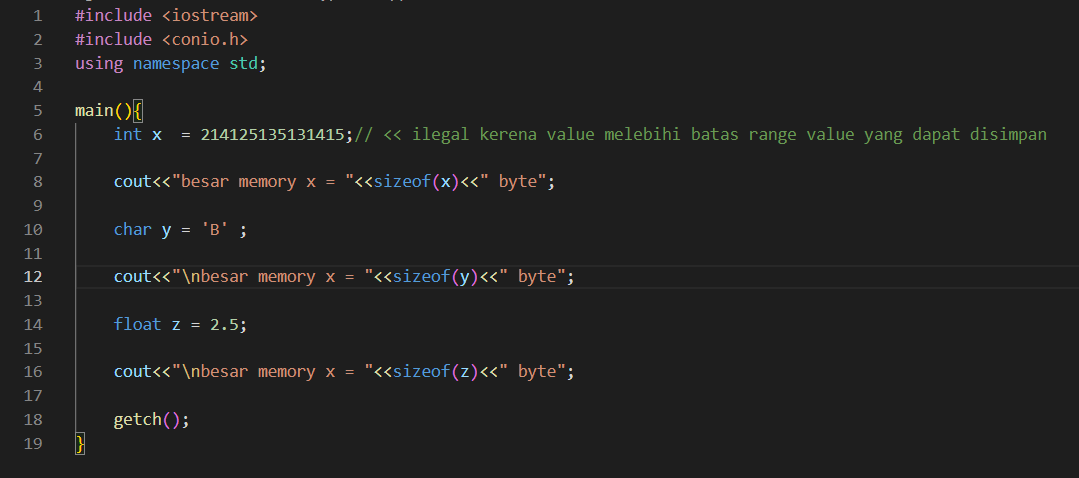


Table type data, besar memory, dan representasi valuenya :

| **Type Name** | **Bytes** | **Other Names** | **Range of Values** |
| --- | --- | --- | --- |
| **int** | 4 | **signed** | -2,147,483,648 to 2,147,483,647 |
| **unsigned int** | 4 | **unsigned** | 0 to 4,294,967,295 |
| **\_\_int8** | 1 | **char** | -128 to 127 |
| **unsigned \_\_int8** | 1 | **unsigned char** | 0 to 255 |
| **\_\_int16** | 2 | **short**, **short int**, **signed short int** | -32,768 to 32,767 |
| **unsigned \_\_int16** | 2 | **unsigned short**, **unsigned short int** | 0 to 65,535 |
| **\_\_int32** | 4 | **signed**, **signed int**, **int** | -2,147,483,648 to 2,147,483,647 |
| **unsigned \_\_int32** | 4 | **unsigned**, **unsigned int** | 0 to 4,294,967,295 |
| **\_\_int64** | 8 | **long long**, **signed long long** | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| **unsigned \_\_int64** | 8 | **unsigned long long** | 0 to 18,446,744,073,709,551,615 |
| **bool** | 1 | none | **false** or **true** |
| **char** | 1 | none | -128 to 127 by default  0 to 255 when compiled by using [/J](https://learn.microsoft.com/en-us/cpp/build/reference/j-default-char-type-is-unsigned?view=msvc-170) |
| **signed char** | 1 | none | -128 to 127 |
| **unsigned char** | 1 | none | 0 to 255 |
| **short** | 2 | **short int**, **signed short int** | -32,768 to 32,767 |
| **unsigned short** | 2 | **unsigned short int** | 0 to 65,535 |
| **long** | 4 | **long int**, **signed long int** | -2,147,483,648 to 2,147,483,647 |
| **unsigned long** | 4 | **unsigned long int** | 0 to 4,294,967,295 |
| **long long** | 8 | none (but equivalent to **\_\_int64**) | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| **unsigned long long** | 8 | none (but equivalent to **unsigned \_\_int64**) | 0 to 18,446,744,073,709,551,615 |
| **enum** | varies | none |  |
| **float** | 4 | none | 3.4E +/- 38 (7 digits) |
| **double** | 8 | none | 1.7E +/- 308 (15 digits) |
| **long double** | same as **double** | none | Same as **double** |
| **wchar\_t** | 2 | **\_\_wchar\_t** | 0 to 65,535 |

1. **Operators**

operator adalah sebuah symbol yang berfungsi untuk menjalankan sebuah operasi terhadap variabel dan valuenya.

Ada macam macam kategori operator yaitu :

* **Arithmetic Operators**

|  |  |
| --- | --- |
| Operator | Operation |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulo Operation (Remainder after division) |

* **Assignment Operators**

|  |  |  |
| --- | --- | --- |
| Operator | Example | Equivalent to |
| = | a = b; | a = b; |
| += | a += b; | a = a + b; |
| -= | a -= b; | a = a - b; |
| \*= | a \*= b; | a = a \* b; |
| /= | a /= b; | a = a / b; |
| %= | a %= b; | a = a % b; |
| ++ | A++ | A = A + 1 |
| -- | B++ | B = B + 1 |

* **Relational Operators**

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| == | Is Equal To | 3 == 5 gives us **false** |
| != | Not Equal To | 3 != 5 gives us **true** |
| > | Greater Than | 3 > 5 gives us **false** |
| < | Less Than | 3 < 5 gives us **true** |
| >= | Greater Than or Equal To | 3 >= 5 give us **false** |
| <= | Less Than or Equal To | 3 <= 5 gives us **true** |

* **Logical Operators**

|  |  |  |
| --- | --- | --- |
| Operator | Example | Meaning |
| && | expression1 **&&** expression2 | Logical AND. True only if all the operands are true. |
| || | expression1 **||** expression2 | Logical OR. True if at least one of the operands is true. |
| ! | **!**expression | Logical NOT. True only if the operand is false. |

* **Bitwise Operators**

|  |  |
| --- | --- |
| Operator | Description |
| & | Binary AND |
| | | Binary OR |
| ^ | Binary XOR |
| ~ | Binary One's Complement |
| << | Binary Shift Left |
| >> | Binary Shift Right |

Terdapat operator yang lainnya yang biasa digunakan yaitu :

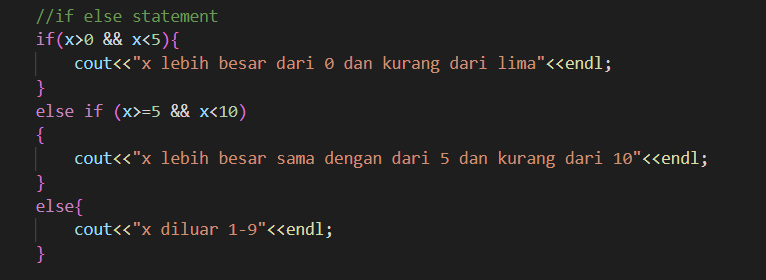
|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| sizeof | returns the size of data type | sizeof(int); // 4 |
| ?: | returns value based on the condition | string result = (5 > 0) ? "even" : "odd"; // "even" |
| & | represents memory address of the operand | &num; // address of num |
| . | accesses members of struct variables or class objects | s1.marks = 92; |
| -> | used with pointers to access the class or struct variables | ptr->marks = 92; |
| << | prints the output value | cout << 5; |
| >> | gets the input value | cin >> num; |

1. **Conditions**

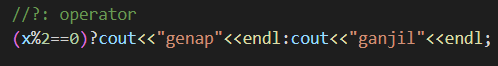
Condition pada c++ merujuk pada keword atau operator yang berfungsi sebagai pembanding, apakah blok code tertentu akan dieksekusi atau tidak.

Terdapat 3 cara menggunakan condition dalam C++ yaitu:

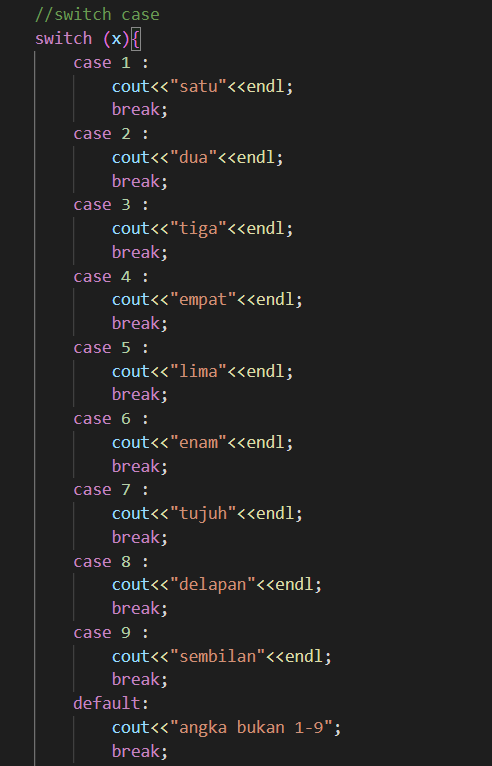
* **IF/ELSE**



* **OPERATOR ?:**



* **SWITCH CASE**

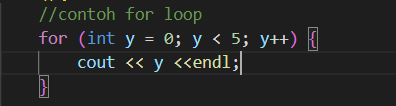


1. **Loops**

Dalam pemrograman C++, "looping" (atau perulangan) adalah mekanisme untuk mengeksekusi blok kode berulang kali selama suatu kondisi tetap benar (true). C++ menyediakan beberapa jenis struktur perulangan: for, while, dan do...while.

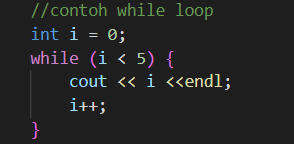
* **For loop**

Loop for digunakan ketika Anda tahu berapa kali Anda ingin mengeksekusi suatu pernyataan atau blok pernyataan.



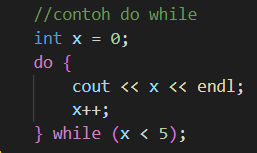
* **While loop**

Loop while digunakan ketika Anda ingin mengeksekusi blok kode selama kondisi tertentu adalah true.



* **Do while loop**

Mirip dengan loop while, tetapi perbedaannya adalah loop do...while mengeksekusi blok kode setidaknya sekali sebelum memeriksa kondisi, karena pengecekan kondisi dilakukan setelah eksekusi blok kode.



Dalam semua jenis loop ini, sangat penting untuk memastikan bahwa kondisi loop pada akhirnya akan menjadi false sehingga program bisa keluar dari loop. Jika tidak, program Anda bisa masuk ke dalam loop tak terbatas, yang bisa menyebabkan program menjadi tidak responsif atau mengalami crash.

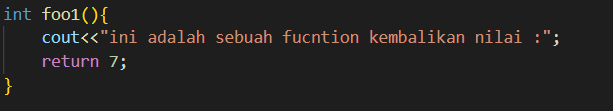
Selain itu, Anda juga dapat mengendalikan loop dengan pernyataan seperti break (untuk keluar dari loop) dan continue (untuk melompat ke iterasi selanjutnya dari loop).

1. **Functions & Pocedure**

Dalam bahasa pemrograman C++, terdapat dua konsep yang sering digunakan untuk mengorganisasikan kode, yaitu "function" (fungsi) dan "procedure". Meskipun dalam C++, istilah "procedure" tidak secara eksplisit digunakan seperti dalam beberapa bahasa lain (seperti Pascal), biasanya istilah ini merujuk pada fungsi yang tidak mengembalikan nilai. Sebaliknya, dalam C++, kita biasanya menyebutnya sebagai "fungsi dengan tipe pengembalian void".

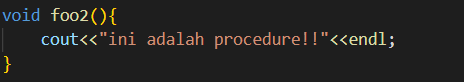
* **Fungsi**

Adalah sebuah blok kode yang dapat menerima beberapa parameter, menjalankan instruksi berdasarkan parameter tersebut, dan mengembalikan sebuah nilai.



* **Prosedure**

Seperti yang disebutkan sebelumnya, dalam C++, prosedure dapat dilihat sebagai fungsi dengan tipe pengembalian void. Ini berarti prosedure tidak mengembalikan nilai apa pun.



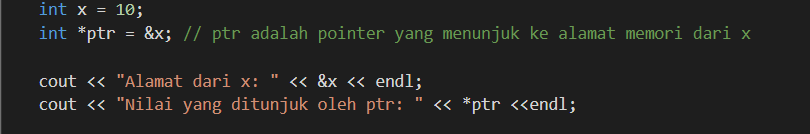
Meskipun terdapat perbedaan antara fungsi dan prosedure berdasarkan apakah mereka mengembalikan nilai atau tidak, secara struktural, mereka keduanya adalah blok kode yang dapat dipanggil oleh bagian lain dari program. Fungsi dan prosedure membantu dalam membuat kode yang terorganisir, mudah dibaca, dan dapat digunakan kembali.

1. **Reference & Pointer**

Dalam C++, reference dan pointer adalah dua mekanisme yang memungkinkan Anda untuk merujuk atau berinteraksi dengan lokasi memori dari variabel lain. Meskipun keduanya memiliki fungsi serupa, cara kerja dan penggunaannya berbeda. Mari kita pelajari perbedaannya:

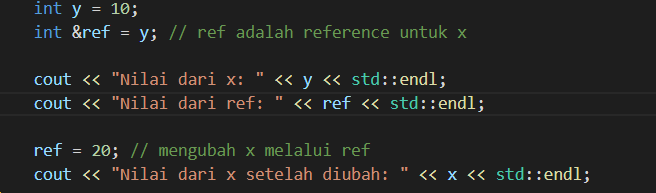
* **Pointer:**

Sebuah variabel yang menyimpan alamat memori dari tipe data lain. Menggunakan operator \* untuk mendeklarasikannya dan & untuk mendapatkan alamat memori dari variabel. Dapat diubah untuk menunjuk ke alamat memori yang berbeda. Dapat memiliki nilai nullptr yang menandakan pointer tidak menunjuk ke mana pun.



* **Reference:**

Alias atau nama lain untuk variabel yang sudah ada. Seolah-olah Anda memiliki dua nama untuk objek yang sama. Harus diinisialisasi saat dideklarasikan dan tidak bisa berubah untuk merujuk ke variabel lain setelah diinisialisasi. Tidak dapat memiliki "nilai nol"; selalu harus merujuk ke objek yang ada.



1. **OPP**

Object-Oriented Programming (OOP) adalah paradigma pemrograman yang menggunakan "objek" dan kelas untuk menyusun kode. C++ adalah salah satu bahasa pemrograman yang mendukung OOP. Konsep-konsep utama dari OOP dalam C++ antara lain:

1. **Kelas (Class) dan Objek (Object)**

* **Kelas**: Definisi blueprint atau cetakan untuk membuat objek. Kelas mendefinisikan variabel (atribut) dan metode (fungsi) yang akan dimiliki oleh objek.
* **Objek**: Instansiasi dari kelas. Sebuah representasi konkret dari kelas.

1. **Enkapsulasi**

Menggabungkan data dan metode yang beroperasi pada data tersebut ke dalam satu unit (kelas) dan menyembunyikan detail-detail implementasi dari pengguna.

1. **Pewarisan (Inheritance):**

Kemampuan sebuah kelas untuk mewarisi atribut dan metode dari kelas lain.

1. **Polimorfisme**

Kemampuan objek untuk diambil dalam banyak bentuk. Dalam C++, hal ini sering dicapai melalui penggunaan metode virtual dan pewarisan.

1. **Abstraksi**

Menyembunyikan detail kompleks dan menampilkan fitur-fitur esensial dari objek.

Dengan menggunakan OOP, kode dapat menjadi lebih modular, lebih mudah untuk dikelola, diperbaharui, dan didebug. Konsep-konsep OOP dalam C++ membantu dalam pengembangan perangkat lunak skala besar dan mempermudah kolaborasi antar developer.