Modul Praktikum
SDAA

1

PENGENALAN BAHASA PEMROGRAMAN C++

TINJAUAN UMUM BAHASA PEMROGRAMAN C++

C++ adalah sebuah bahasa pemrograman tingkat tinggi yang dirancang untuk berbagai keperluan, mulai dari pengembangan perangkat lunak skala kecil hingga aplikasi yang kompleks dan besar. Bahasa ini dikembangkan oleh Bjarne Stroustrup di Bell Labs, dimulai pada tahun 1979 dan resmi dirilis pada tahun 1985 sebagai sebuah perluasan dari bahasa pemrograman C. C++ dirancang untuk mendukung pemrograman yang lebih fleksibel dan efisien, menggabungkan fitur-fitur dari C dengan penambahan konsep-konsep baru yang memungkinkan manipulasi memori secara langsung, serta mendukung pemrograman berorientasi objek, generik, dan fungsional.

Gambar 1. Logo Bahasa Pemrograman C++

Sejak rilis pertamanya, C++ telah mengalami perkembangan yang signifikan, menjadi salah satu bahasa pemrograman yang paling banyak digunakan di berbagai industri. Pada tahun 1997, C++ sudah dilengkapi dengan kemampuan pemrograman berorientasi objek yang kuat, yang memungkinkan pengembang untuk membuat struktur data yang lebih kompleks dan mendukung konsep pewarisan, enkapsulasi, dan polimorfisme. Selain itu, fitur-fitur generik dan fungsional juga telah ditambahkan untuk memberikan fleksibilitas dalam penulisan kode, sehingga dapat digunakan untuk berbagai macam aplikasi, mulai dari perangkat lunak sistem seperti sistem operasi, hingga perangkat lunak yang berjalan pada sistem dengan sumber daya terbatas seperti mikrokontroler.

Bahasa C++ sangat diakui karena performa dan efisiensinya, yang membuatnya menjadi pilihan utama untuk pengembangan perangkat lunak yang memerlukan pengelolaan sumber daya yang ketat, seperti dalam pengembangan sistem operasi seperti Linux atau Windows. Dengan desain yang memungkinkan pengelolaan memori secara manual, C++ memberikan kontrol penuh kepada pengembang atas bagaimana memori digunakan dan dioptimalkan, sehingga sangat cocok untuk pengembangan perangkat lunak yang memerlukan kinerja tinggi dan respon yang cepat.

Selain itu, C++ juga telah distandarisasi oleh International Organization for Standardization (ISO) untuk memastikan konsistensi dan kompatibilitas antar implementasi bahasa ini di berbagai platform. Standar terbaru yang berlaku saat ini adalah ISO/IEC 14882:2020, yang dikenal sebagai C++20. Standar ini dirilis pada Desember 2020 dan mencakup berbagai penambahan dan perbaikan fitur yang memperluas kemampuan bahasa ini. Sebelum C++20, bahasa ini telah melalui beberapa iterasi standarisasi, dimulai dengan C++98 yang diresmikan pada tahun 1998, diikuti oleh C++03, C++11, C++14, dan C++17. Setiap versi standar ini membawa peningkatan dan pembaruan yang signifikan, menjadikan C++ tetap relevan dan terus digunakan dalam pengembangan perangkat lunak modern.

Secara keseluruhan, C++ adalah bahasa yang sangat powerful dan serbaguna, mampu menangani berbagai kebutuhan pemrograman, dari pengembangan perangkat lunak embedded hingga aplikasi enterprise berskala besar. Fleksibilitasnya, digabungkan dengan fokus pada performa dan efisiensi, menjadikannya pilihan yang kuat untuk pengembangan perangkat lunak yang kompleks dan kritis.

KONSEP FUNDAMENTAL BAHASA PEMROGRAMAN C++

Dalam bahasa pemrograman C++, terdapat beberapa konsep penting yang perlu dipahami. Salah satunya adalah struktur dasar program, yang menjelaskan cara program yang ditulis akan dieksekusi oleh sistem. Selain itu, ada juga sintaks dasar yang terdiri dari berbagai elemen seperti variabel, tipe data, dan operator. Struktur kontrol juga menjadi bagian penting dalam C++, yang meliputi konsep seperti perulangan (*for*, *while*, *do-while*), pernyataan kondisional (*if*, *switch*), dan penggunaan fungsi. Terakhir, terdapat konsep input dan output yang dilakukan menggunakan operasi "cin" dan "cout", serta mekanisme File I/O untuk membaca dan menulis data ke dalam file.

STRUKTUR DASAR

Pada umumnya, pemrograman dalam bahasa C++ memiliki struktur dasar seperti berikut:

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

int main() {
  // menggunakan std
  std::cout << "Hello World";

  // dengan menggunakan namespace std
  cout << "Hello World";

  return 0;
}
```

Gambar 2. Struktur Dasar Source Code C++

`#include <iostream>` : merupakan sebuah arahan preprocessor yang mencakup konten file header standar C++ yaitu iostream. Header ini berfungsi agar program yang dibuat dapat meminta input dan memberikan output.

`using namespace std;` : digunakan untuk memberi instruksi kepada compiler agar memperlakukan semua sintaks yang dimulai dengan std seolah-olah berada dalam ruang lingkup global. Perbedaan dapat dilihat pada Gambar 2.

`int main()` : baris ini mendeklarasi sebuah fungsi utama yang dimana tempat perintah eksekusi program dijalankan, fungsi ini dinamakan "main". Pada satu program, hanya boleh ada satu fungsi "main".

`{` dan `}` : tempat dimulainya satu blok program.

`return 0;` : perintah untuk mengembalikan nilai 0 ketika fungsi "main" selesai dan berhasil dijalankan tanpa menghasilkan error.

SINTAKS

Terdapat beberapa sintaks dasar pada bahasa pemrograman C++, sintaks ini terdiri dari tipe data, variabel, dan operator.

1. Tipe Data

   Pada bahasa pemrograman C++, terdapat beberapa tipe data yang tiap tipenya memiliki tujuan masing-masing, tiap tipe ini terdiri dari:

   a. Tipe Data Primitive

      Tipe data primitive adalah jenis tipe data fundamental yang digunakan untuk operasi dasar. Contoh dari tipe data ini meliputi int, boolean, float, double, char, dan lain sebagainya. Setiap tipe data tersebut memiliki karakteristik dan rentang nilai yang spesifik.

Tabel 1. Ukuran Tiap Tipe Data

Tipe Data	Byte	Range
unsigned char	1	0 – 255
signed char	1	-127 – 127
int	4	-2.147.483.684 – 2.147.483.647
unsigned int	4	0 – 4.294.967.295
short int	2	-32.768 – 32.767

unsigned short int	2	0 – 65.535
long int	4	-2.147.483.684 – 2.147.483.647
unsigned long int	4	0 – 4294967295
long long int	8	-(2^63) – (2^63)-1
unsigned long long int	8	0 – 18.446.744.073.709.551.615
float	4	-3.4×10^38 – 3.4×10^38
double	8	-1.7×10^308 – 1.7×10^308
long double	12	-1.1×10^4932 – 1.1×10^4932

Tipe data integer menggunakan kata kunci "int" dan digunakan untuk menyimpan bilangan bulat. Biasanya, integer memerlukan ruang memori sebesar 4 byte dan memiliki rentang nilai dari -2147483648 hingga 2147483647.

Tipe data karakter digunakan untuk menyimpan karakter, dengan kata kunci "char". Karakter biasanya memerlukan 1 byte ruang memori dan memiliki rentang nilai dari -128 hingga 127 atau 0 hingga 255.

Tipe data Boolean digunakan untuk menyimpan nilai Boolean atau logika, yaitu "true" atau "false", dengan kata kunci "bool".

Tipe data float atau floating-point digunakan untuk menyimpan nilai float dengan presisi tunggal atau nilai desimal, menggunakan kata kunci "float". Variabel "float" biasanya memerlukan 4 byte ruang memori.

Tipe data double atau double floating-point digunakan untuk menyimpan nilai floating-point dengan presisi ganda atau nilai desimal, menggunakan kata kunci "double". Variabel "double" biasanya memerlukan 8 byte ruang memori.

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

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

  return 0;
}
```

Gambar 3. Source Code Ukuran Tiap Tipe Data

b. Tipe Data Non-primitive

Tipe data non-primitif adalah jenis tipe data yang lebih kompleks dibandingkan tipe data primitif dan biasanya terdiri dari gabungan beberapa elemen. Berikut adalah beberapa contoh tipe data non-primitif:

- Tipe Data String

  Tipe data string digunakan untuk menyimpan kumpulan karakter yang membentuk sebuah teks, seperti "Andi", "Duniailkom", atau "Belajar C++". String bukan hanya satu karakter, tetapi serangkaian karakter yang disimpan sebagai satu kesatuan, sehingga memungkinkan pengolahan teks yang lebih kompleks dalam pemrograman.

- Tipe Data Array

  Tipe data array adalah sebuah struktur data yang menyimpan sejumlah elemen yang sejenis dalam satu variabel. Semua elemen dalam array memiliki tipe data yang sama, dan mereka disimpan dalam urutan yang berurutan di memori. Array memungkinkan pengelolaan dan manipulasi sejumlah besar data secara lebih efisien dalam program.

- Tipe Data Structure (Struct)

  Tipe data structure atau struct adalah kumpulan dari berbagai tipe data dasar yang dikelompokkan bersama dalam satu unit. Struktur ini memungkinkan penyimpanan beberapa jenis data yang berbeda dalam satu entitas, seperti menyimpan data seseorang yang terdiri dari nama (string), umur (integer), dan jenis kelamin (karakter).

- Tipe Data Enum (Enumerasi)

  Tipe data enum adalah tipe data yang didefinisikan oleh pengguna untuk membuat daftar nilai yang terstruktur. Dengan enum, kita dapat membuat sekumpulan nama

konstanta yang merepresentasikan nilai-nilai tertentu, yang memudahkan pembacaan dan pengelolaan kode.

- Tipe Data Pointer
  Tipe data pointer digunakan untuk menyimpan alamat memori dari variabel lain. Dengan pointer, kita dapat mengakses dan memanipulasi data yang disimpan di lokasi memori tertentu secara langsung, yang memberikan fleksibilitas lebih dalam pengelolaan memori dalam program. Pointer merupakan konsep penting dalam C++ karena memungkinkan manipulasi data pada tingkat yang lebih rendah dan efisien.

2. Variabel
   Variabel adalah placeholder atau wadah dalam memori (RAM) untuk menyimpan nilai. Format deklarasi variabel pada bahasa C++ adalah sebagai berikut.

```
<tipe_data>  <nama_variabel>;
```

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

int main() {
  int a; // Deklarasi variabel
  int b = 1; // Inisialisasi variabel
  char c = 'c';

  cout << a << endl;
  cout << b << endl;
  cout << c << endl;

  return 0;
}
```

Gambar 4. Variabel

3. Operator
   C++ mendukung berbagai macam operator yang secara luas dapat dibagi kedalam kategori berikut:
   a. Operator Aritmatika
      Operasi yang digunakan untuk operasi aritmatika.

Penjumlahan	+
Pengurangan	-
Perkalian	*
Pembagian	/
Modulus (sisa bagi)	%

Increment	++
Decrement	--

b. Operator Penugasan

Operator penugasan (Assignment Operator) merupakan operator untuk memberikan tugas pada variabel. Biasanya untuk mengisi nilai.

Penugasan	=
Penjumlahan	+=
Pengurangan	-=
Perkalian	*=
Pembagian	/=
Modulus (sisa bagi)	%=

c. Operator Perbandingan

Nilai yang dikembalikan dari operator ini berbentuk boolean.

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

d. Operator Logika

Operator logika digunakan untuk memeriksa kesamaan nilai dari dua data atau lebih dan nilai yang dikembalikan dari operator ini berbentuk boolean.

AND	&&
OR	||
NOT	!

e. Operator Bitwise

Operator Bitwise adalah operasi matematika yang mengoperasikan pada bilangan biner.

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

STRUKTUR KONTROL

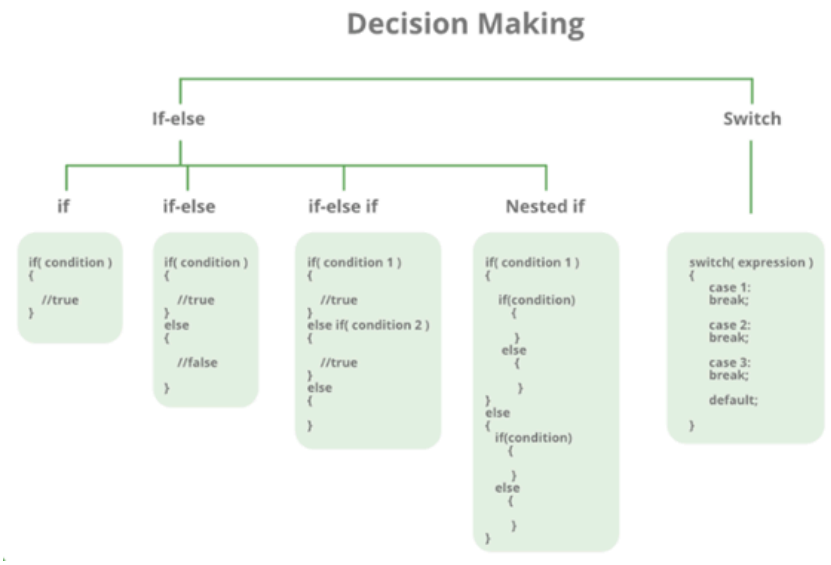
Struktur kontrol adalah mekanisme yang digunakan untuk mengatur alur eksekusi dalam suatu program. Algoritma atau program apa pun dapat menjadi lebih jelas dan mudah dipahami jika disusun menggunakan modul-modul yang berdiri sendiri, yang dikenal sebagai struktur logika atau kontrol. Struktur ini berfungsi untuk menganalisis dan menentukan arah alur program berdasarkan kondisi atau parameter tertentu. Terdapat tiga jenis logika dasar atau alur kontrol yang dikenal, yaitu:

1. Logika sekuensial, atau alur urutan

   Logika sekuensial, sesuai dengan namanya, mengikuti alur yang bersifat serial atau berurutan, di mana alur eksekusi bergantung pada serangkaian instruksi yang diberikan kepada komputer. Modul-modul akan dijalankan secara berurutan sesuai urutan instruksi yang ada, kecuali jika ada instruksi baru yang diberikan. Urutan ini dapat ditentukan secara eksplisit melalui langkah-langkah yang diberi nomor, atau secara implisit mengikuti urutan di mana modul-modul tersebut ditulis dalam kode. Sebagian besar pemrosesan, bahkan untuk beberapa masalah yang kompleks, umumnya akan mengikuti pola alur dasar ini.

2. Logika kondisional, atau alur pemilihan

   Logika kondisional melibatkan sejumlah kondisi atau parameter yang menentukan satu dari beberapa modul yang telah ditulis. Struktur yang menggunakan jenis logika ini dikenal sebagai Struktur Kondisional. Dalam bahasa pemrograman C++, logika kondisional atau percabangan ini terbagi menjadi beberapa bagian seperti yang dapat dilihat pada Gambar 5.



Gambar 5. Jenis Percabangan pada C++

- If

```
if (kondisi) {
    // Menjalankan isi dari blok ini apabila kondisi benar
}
```

- If-else

```
if (kondisi) {
    // Menjalankan isi dari blok ini apabila kondisi benar
} else {
    // Menjalankan isi dari blok ini apabila kondisi salah
}
```

- Else-if

```
if (kondisi) {
    // Menjalankan blok ini apabila kondisi benar
} else if (kondisi) {
    // Menjalankan blok ini bila kondisi sebelumnya salah
} else {
    // Menjalankan blok bila kedua kondisi sebelumnya salah
}
```

- Nested if

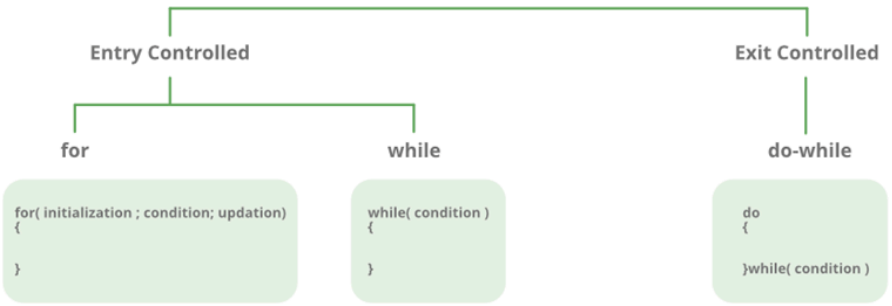
```
if (kondisi1) {
    // Menjalankan blok ini apabila kondisi1 benar
    if (kondisi2) {
            // Menjalankan blok ini apabila kondisi2 benar
    }
}
```

- Switch

```
switch (n) {
    case 1:
        // Menjalankan blok ini apabila n = 1
        break;
    case 2:
        // Menjalankan blok ini apabila n = 2
        break;
    default:
        // Menjalankan blok ini jika tidak ada yang cocok
}
```

3. Logika iterasi, atau alur perulangan

   Dalam pemrograman, terkadang diperlukan untuk menjalankan suatu operasi lebih dari sekali atau sebanyak n kali. Loop digunakan ketika kita perlu mengeksekusi ulang sebuah blok pernyataan secara berulang.



Gambar 6. Jenis Perulangan pada C++

- For

```
for (initialization expr; test expr; update expr) {
    // kode yang ingin dijalankan
}
```

Initialization Expression: Menetapkan nilai awal yang digunakan untuk memulai perulangan. Contoh: "int i = 1"

Test Expression: Menentukan kondisi perulangan yang menentukan kapan perulangan harus berhenti. Contoh: "i < 10;"

Update Expression: Setelah perulangan selesai dieksekusi, nilai variabel dapat diubah pada pernyataan ini, biasanya menggunakan Increment. Contoh: "i++"

- While

```
while (kondisi) {
    // dijalankan selama kondisi benar
}
```

- Do While

```
do {
    // menjalankan kode
} while (kondisi); // mengecek kondisi
```

I/O

Bahasa pemrograman C++ dilengkapi dengan berbagai pustaka yang menyediakan banyak cara untuk melakukan operasi input dan output. Dalam C++, input dan output dilakukan dalam bentuk rangkaian byte, yang lebih umum dikenal sebagai aliran data atau stream.

Input Stream digunakan untuk aliran byte yang bergerak dari perangkat (misalnya, keyboard) menuju memori utama, maka proses ini disebut sebagai *input*. Sebaliknya, *Output Stream* terjadi jika aliran byte bergerak dari memori utama ke perangkat (misalnya, layar tampilan), yang dikenal sebagai *output*.

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

int main() {
    int angka;
    cout << "Masukkan angka: " << endl;
    cin >> angka;
    cout << "Angka masuk: " << angka << endl;

    return 0;
}
```

Gambar 7. Input dan Output menggunakan console di C++

1. Input

   Input atau "cin" yang merupakan singkatan dari console input merupakan sebuah fungsi yang terdapat pada pustaka "iostream". Kegunaan dari fungsi ini ialah mengambil input dari pengguna melalui console atau terminal. Fungsi ini dapat digunakan dengan format sebagai berikut.

   ```
   cin >> variabel;
   ```

2. Output

   Output atau "cout" yang merupakan singkatan dari console output merupakan sebuah fungsi yang terdapat juga pada Pustaka "iostream". Fungsi ini berkebalikan dari "cin" yang mengambil input dari pengguna, yang mana fungsi ini berfungsi untuk menampilkan output kepada pengguna melalui console atau terminal. Fungsi ini dapat digunakan dengan format sebagai berikut.

   ```
   cout << hal_yang_ingin_dilihat;
   ```

3. File

   Dalam bahasa pemrograman C++, kita dapat melakukan penulisan dan pembacaan terhadap file. Pustaka yang digunakan adalah "fstream", namun dapat pula dispesifikkan jika ingin menggunakan salah satunya saja, yaitu "ofstream" untuk melakukan penulisan data ke file dan "ifstream" untuk membaca data dari file.

   Pada Gambar 8, terdapat sintaks yang mungkin terlihat asing, yaitu "getline". Fungsi ini sebenarnya mirip dengan "cin" yang digunakan untuk mengambil input dari pengguna. Namun, perbedaannya adalah "getline" mengambil input sebagai string, sedangkan "cin" mengambil input dalam bentuk karakter. Selain itu, sangat disarankan untuk selalu menutup file setelah operasi terhadap file tersebut selesai. Hal ini penting untuk memastikan bahwa tidak ada memori yang tetap terpakai untuk operasi file tersebut.

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

int main() {
    // Memberi input melalui console
    int angka;
    cout << "Masukkan angka: ";
    cin >> angka;

    // Menulis data ke file
    ofstream outfile("data.txt");
    if (outfile.is_open()) {
        outfile << "Angka masuk: " << angka << endl;
        outfile.close();
    }

    // Membaca data dari file dan menampilkan ke console
    ifstream infile("data.txt");
    string isi_file;
    if (infile.is_open()) {
        while (getline(infile, isi_file)) {
            cout << isi_file << endl;
        }
        infile.close();
    }

    return 0;
}
```

Gambar 8. File I/O

FUNGSI DAN PROSEDUR

Ketika program yang kita buat semakin kompleks, akan semakin sulit untuk dipahami, terlebih lagi disaat kita ingin mengubah atau menambahkan program pada bagian tertentu. Dengan adanya fungsi kita dapat memecah program menjadi sub-sub program yang lebih sederhana sehingga program tidak menumpuk di satu fungsi utama. Penggunaan fungsi juga sangat efisien karena dapat digunakan atau dipanggil kembali, baik di dalam maupun di luar fungsi itu sendiri.

Fungsi sama seperti halnya dengan prosedur, namun tetap ada perbedaannya. Perbedaan pada fungsi dan prosedur dapat dilihat dari tabel berikut.

Tabel 2. Perbedaan Fungsi dan Prosedur

Fungsi	Prosedur
Menggunakan tipe data dalam deklarasinya seperti int, float, string, dll.	Tidak terdapat tipe data dalam deklarasi. Dideklarasikan dengan keyword Void
Mengembalikan nilai (return value).	Tidak mengembalikan nilai.

Keyword void pada prosedur berarti fungsi tersebut tidak mengembalikan nilai. Untuk nilai yang dikembalikan pada fungsi memiliki tipe data yang sama dengan tipe data fungsi.

TIPE FUNGSI

Terdapat dua tipe fungsi yaitu *user-defined function* dan *standard template library*. Masing-masing dari tipe tersebut memiliki cara penggunaan yang berbeda.

1. User-defined Function
   User-defined function merupakan sebuah fungsi yang harus dibuat secara manual oleh pengguna/programmer terlebih dahulu agar bisa digunakan sesuai dengan kebutuhan.

2. Standard Template Library (STL)
   Standard Template Library atau STL adalah sebuah fungsi bawaan (Built-in function) yang sudah tersedia pada C++, sehingga kita hanya perlu memanggil nama library-nya untuk dapat menggunakan fungsi-fungsi yang ada di dalamnya. Contoh dari STL ini adalah "iostream".

METODE DEKLARASI

Untuk mendeklarasikan sebuah fungsi atau prosedur, terdapat sebuah metode sebagai berikut.

Deklarasi Fungsi

```
Tipe_data nama_fungsi(tipe_data_parameter nama_parameter){
    // isi fungsi
}
```

Deklarasi Prosedur

```
void nama_fungsi(tipe_data_parameter nama_parameter){
    // isi fungsi
}
```

Fungsi akan mengembalikan sebuah nilai (return value) yang merupakan hasil dari prosesnya. Oleh karena itu, kita perlu menentukan terlebih dahulu tipe data dari nilai yang akan dikembalikan oleh fungsi tersebut. Penggunaan parameter pada fungsi dan prosedur bersifat opsional, artinya bisa ada atau tidak.

JENIS PENULISAN FUNGSI

Pada program, fungsi dapat dibuat sebelum atau sesudah penulisan fungsi main. Bila dibuat sesudah penulisan fungsi main, maka harus mendefinisikan (menuliskan prototype fungsi) di bagian atas program (sebelum fungsi main).

Contoh fungsi **setelah** fungsi main() :

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

/* Prototype atau deklarasi Fungsi*/
int tambah(int a1, int a2);
int main() {
    int angka1, angka2;
    cout << "Angka Pertama : "; cin >> angka1;
    cout << "Angka Kedua : "; cin >> angka2;
    cout << "Hasil = " << tambah(angka1, angka2);
}

int tambah(int a1, int a2) {
    int hasil;
    hasil = a1 + a2;
    return hasil;
}
```

Contoh fungsi **sebelum** fungsi main() :

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

int tambah(int a1, int a2) {
    int hasil;
    hasil = a1 + a2;
    return hasil;
}

int main() {
    int angka1, angka2;
    cout << "Angka Pertama : "; cin >> angka1;
    cout << "Angka Kedua : "; cin >> angka2;
    cout << "Hasil = " << tambah(angka1, angka2);
}
```

PENGGOLONGAN VARIABEL

Variabel lokal adalah sebuah variabel sederhana yang dideklarasikan di dalam sebuah fungsi. Variabel ini hanya bisa diakses dalam fungsi itu sendiri. Sedangkan variabel global memiliki ruang lingkup sepanjang semua program dijalankan, sehingga variabel global akan tetap dikenali dimanapun bagian atau blok program dijalankan.

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

int a = 7; // variabel global
int ambilA() {
    int a = 5; // variabel lokal
    return a;
}
int main() {
    int b = ambilA();
    cout << "Nilai Dari Variabel Global a = " << a <<endl;
    cout << "Nilai Dari Variabel Lokal a = " << b <<endl;
    return 0;
}
```

FUNGSI OVERLOADING

Fungsi Overloading dalam C++ memungkinkan adanya banyak fungsi dengan nama yang sama namun memiliki parameter yang berbeda. Ini merupakan salah satu fitur dari pemrograman berorientasi objek yang memberikan fleksibilitas dan kemudahan dalam penggunaan fungsi.

Fungsi overloading adalah kemampuan untuk mendefinisikan beberapa fungsi dengan nama yang sama dalam satu program, namun dengan parameter yang berbeda—baik dalam jumlah, jenis, atau urutan parameter. Compiler akan secara otomatis menentukan fungsi mana yang akan dipanggil berdasarkan argumen yang diberikan saat pemanggilan fungsi.

Fungsi overloading berguna ketika kita ingin melakukan operasi yang serupa, tetapi dengan tipe data atau jumlah argumen yang berbeda. Ini membuat kode lebih intuitif dan mudah dipahami karena Anda tidak perlu membuat nama fungsi yang berbeda-beda untuk setiap variasi.

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

void display(int);
void display(float);
void display(int, float);
```

```
int main() {
    int a = 5;
    float b = 5.5;
    display(a);
    display(b);
    display(a, b);
    return 0;
}

void display(int var) {
    cout << "Integer: " << var << endl;
}

void display(float var) {
    cout << "Float: " << var << endl;
}

void display(int var1, float var2) {
    cout << "Integer: " << var1;
    cout << " and float:" << var2;
}
```

FUNGSI REKURSIF

Fungsi rekursif dalam pemrograman adalah fungsi yang memanggil dirinya sendiri. Fungsi ini sering dibandingkan dengan perulangan karena memiliki pola yang berulang pada setiap pemanggilannya. Fungsi rekursif berguna dalam menyelesaikan masalah tertentu, seperti perhitungan bilangan Fibonacci dan faktorial.

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

int hitungFactorial(int input) {
    if(input > 1) {
        return input * hitungFactorial(input - 1);
    } else {
        return 1;
    }
}

int main() {
    cout << "## Program C++ Hitung Faktorial ##" << endl;
    cout << "==================================" << endl;
    cout << endl;
    int angka;
    cout << "Input angka: ";
    cin >> angka;
```

```
    cout << angka << "! = " << hitungFactorial(angka);
    cout << endl;
    return 0;
}
```