Penjelasan dan macam-macam Preprocessor Directive

Written by: Fajar Laksono | Updated on: Mei 16, 2020

Penjelasan Arahan Preprosesor

Arahan Preprosesor atau *Preprocessor Directive* adalah intruksi untuk *CPU* memerintahkan untuk memproses suatu arahan atau informasi sebelum memulai melakukan kompilasi pada program yang telah kita tulis.

Arahan Preprosesor merupakan sebuah instruksi dan bukan merupakan sebuah pernyataan, Setiap intruksi Arahan Preprosesor dimulai dengan tanda pagar `#` , dan tidak di akhiri dengan sebuah tanda titik koma `;` .

Penyertaan *File* Berkas atau *Source File Inclusion*

Merupakan intruksi yang telah dan sering kita gunakan pada contoh-contoh program sebelumnya. Penyertaan *File* Sumber adalah sebuah intuksi yang memberikan perintah kepada *CPU* untuk menyertakan ke dalam program yang kita tulis sebelum melakukan kompilasi, sebuah *file* tersebut biasanya merupakan sebuah pustaka, atau kode yang pernah kita tulis pada *file* berbeda. Ada dua cara untuk menggunakan Penyertaan *File* Sumber, yaitu:

Bentuk Penulisan:

```
1  #include <header> //cara 1
2  #include "file" //cara 2
```

Cara pertama adalah cara untuk menyertakan pustaka standar yang telah disediakan oleh C/C++, seperti *iostream*, string dan lain sebagainya, untuk melakukanya dibutuhkan sepasang kurung sudut `< >` .

Cara kedua adalah cara dimana kita menggunakan tanda petik dua `"` yang digunakan untuk menyertakan sebuah *file*.

Contoh Program:
hallo.h

```
#ifndef HALLO_H  //(header guards) jika define “HALLO_H” blm ada
#define HALLO_H //maka mendirikan hallo_h

std::string cetak(){ //sebuah function di file header hallo.h
    return "Halo dunia";
}

#endif
```

main.cpp

```
#include <iostream> //menyertakan pustaka standar c++
#include "hallo.h" //menyertakan file header

using namespace std;

int main(){
    cout<<cetak()<<endl; //memanggil function pada file header hallo.h

    return 0;
}
```

Definisi Makro

Definisi Makro atau *Macro definitions* dengan sintak `#define` digunakan untuk membuat konstanta simbolik yang dinamakan sebagai makro.

Bentuk Penulisan:

```
#define IDENTITAS nilai
```

Contoh Penulisan:

```
#define PI 3.14
#define NAMA_PROGRAM "belajarcpp"
```

Ketika *CPU* bertemu dengan intruksi ini maka ia secara kesulurusan akan menggantikan setiap identifikasi di sisa kode dengan makro buatan Anda, tidak peduli bahwa identifikasi merupakan sebuah standar dari C++.

Baca : Pengertian dan Ketentuan Memberikan Indentitas pada Kode Program

Contoh Program:

```
#include <iostream>
#define PI 3.14
using namespace std;

int main(){
    cout<<(PI*7*7)<<endl;
    //PI representasi dari 3.14
    //menjadi memiliki arti 3.14 * 7 * 7

    return 0;
}
```

Macro seperti *Function* (*Function-Like Macros*)

Dalam mendirikan intruksi makro, kita dimungkinkan untuk membuatnya mirip layaknya sebuah *function* yang dapat menerima argumen.

Contoh Penulisan:

```
#define TAMBAH(a,b) (a+b)
```

Contoh Program:

```cpp
#include <iostream>
#define TAMBAH(a,b) (a+b) //Macro dapat menerima 2 argumen

using namespace std;

int main(){
    cout<<(TAMBAH(7,7))<<endl; //7+7

    return 0;
}
```

Penyertaan Berkondisi

Penyertaan berkondisi atau *Conditional Inclusions* pada umumnya mirip seperti pernyataan *IF*, dimana akan membantu kita untuk melakukan penyeleksian berdasarkan kodisi yang ada. Hal ini dapat dilakukan dengan perintah `#if` , `#elif` , `#else` , `#ifdef` atau `#ifndef` dan ditutup dengan `#endif` .

Bentuk Penulisan:

```cpp
//if elif dan else
#if kondisi
//aksi
#elif kondisi
//aksi
#else
//aksi
#endif
```

Contoh Program:

```cpp
#define UMUR 22 //mendirikan makro

#undef TIPE_PRODUK //menghapus makro sebelumnya jika ada
#if UMUR<5 //if
#define TIPE_PRODUK "Balita"
//mendirikan macro TIPE_PRODUK dengan nilai "Balita"
#elif UMUR<17 //else if
#define TIPE_PRODUK "Remaja"
//mendirikan macro TIPE_PRODUK dengan nilai "Remaja"
#elif UMUR<30 //else if
#define TIPE_PRODUK "Dewasa"
//mendirikan macro TIPE_PRODUK dengan nilai "Dewasa"
#elif UMUR<70 //else if
#define TIPE_PRODUK "Orang Tua"
//mendirikan macro TIPE_PRODUK dengan nilai "Orang Tua"
#else //else
#define TIPE_PRODUK "Tidak dikenal"
//mendirikan macro TIPE_PRODUK dengan "Tidak dikenal"
#endif //endif

#include <iostream>
using namespace std;

int main(){
    //pemanggilan macro TIPE_PRODUK
    cout<<"Produk atau program ini telah di atur ulang untuk: "<<(TIPE_PRODUK)<<endl;

    return 0;
}
```

Untuk `#ifdef` dan `#ifndef` menggunakan keberadaan deklarasi makro define sebagai kondisi penentu apakah isi dari pemyeleksian akan di eksekusi atau dilewati.

Contoh Program:

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

#ifndef DEVELOPMENT_MODE
//Jika macro DEVELOPMENT_MODE tidak ada
#define DEVELOPMENT_MODE 0
//maka akan mendirikan macro DEVELOPMENT_MODE
//dengan nilai 0
#endif // DEVELOPMENT_MODE

#ifdef DEVELOPMENT_MODE //Jika DEVELOPMENT_MODE ada
//maka akan mendirikan seuah function di bawah
string statusProyek(){
    return DEVELOPMENT_MODE?"Mode pengembangan":"Mode Produksi";
}
#endif //DEVELOPMENT_MODE

int main(){
    cout<<statusProyek()<<endl;

    return 0;
}
```

Biasanya `#ifndef` dan `ifdef` sering digunakan untuk melakukan *Header guards*, yang bertujuan untuk menghindari duplikasi *header*.

Baca : Penjelasan dan Tipe-tipe Pewarisan (Inheritance)

Nama Makro yang Sudah Ada (*Predefined Macro Names*)

Berikut adalah beberapa makro yang telah disediakan oleh C/C++, makro tersebut biasanya di awali dengan 2 tanda garis bawah `__` .

Makro	Nilai yang dimuat
`__TIME__`	Memuat informasi mengenai waktu saat itu juga dengan bentuk “Jam:Menit:Detik”
`__DATE__`	Memuat informasi mengenai tanggal saat itu juga, dengan bentuk “Bulan Tanggal Tahun”
`__LINE__`	Memuat informasi mengenai nomer baris dari pemanggilan makro tersebut
`__FILE__`	Memuat informasi mengenai alamat direktori dan nama file saat pemanggilan makro tersebut

Contoh Program:

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

int main(){
    cout<<"Waktu : "<<__TIME__<<endl;
    cout<<"Tanggal : "<<__DATE__<<endl;
    cout<<"Nomer Baris : "<<__LINE__<<endl;
    cout<<"Alamat dan nama file : "<<__FILE__<<endl;

    return 0;
}
```

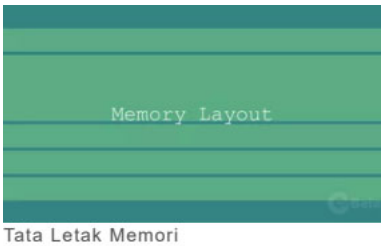

Related

Penjelasan dan Cara Input Output Berbentuk File

September 29, 2019
dalam "C++"

Bagian-Bagian dan Struktur Kode Program C++

Januari 22, 2018
dalam "C++"



Tata Letak Memori

Februari 18, 2018
dalam "C++"