Variable digunakan untuk menyimpan data dan didirikan dengan menggunakan identitas supaya kita dapat memanggilnya untuk menggunakan variabel tersebut. pada artikel sebelumnya (Scope), secara tidak langsung menjelaskan bahwa deklarasi seperti variabel akan memiliki sifat akses tertentu tergantung pada penempatan dan apa yang digunakan.

Seperti ketika kita mendirikan variabel di dalam scope maka variabel itu akan bersifat sebagai Local Variable (Variabel Lokal), dan jika didirkan di luar scope maka akan memiliki sifat sebagai Global Variabel. Di artikel ini penulis akan jelaskan mengenai sifat-sifat akses variabel berdasarkan scope.

Local Variable

Local Variable adalah variabel yang biasanya didirikan di dalam suatu block oleh scope (seperti: function). Variabel tersebut hanya akan terlihat dan hanya dapat digunakan di dalam scope itu saja. Local Variable adalah Automatic duration yang mempunyai arti bahwa variabel tersebut akan dibuat disaat jalur eksekusi dari kompilasi masuk ke dalam definisi dari scope tersebut dan dihancurkan setelah keluar dari scope tersebut.

Contoh Program :

#include <iostream>

using namespace std;

int main ()

{

int id = 1; //[Local Variable] Variabel didirikan dan terlihat setelah pernyataan ini.

cout<<id<<endl;

return 0;

} //variabel "id" dihancurkan dan berakhir di sini.

Local variable didirikan didalam scope dan hanya telihat dan dapat digunakan di scope tersebut, dan jika scope berakhir maka local variable akan dihancurkan. Dalan scope kita juga dimungkinkan untuk melakukan nested scope (Scope di dalam scope). Hal ini akan berlaku sama untuk variabel lokal dari masing-masing scope, variabel lokal akan dihancurkan ketika bertemu dengan blok akhir dari scope. Untuk variabel lokal dari induk scope juga akan terlihat dalam anak scope.

Contoh Program :

#include <iostream>

using namespace std;

int main ()

{

int id = 1; //[Local Variable] Variabel didirikan dan terlihat setelah pernyataan ini.

{

int umur = 2; //[Local Variable] Variabel didirikan dan terlihat setelah pernyataan ini

cout<<id<<endl;

cout<<umur<<endl;

}//Variabel "umur" dihancurkan dan berakhir di sini

return 0;

} //variabel "id" dihancurkan dan berakhir di sini.

Local varibel “id” didirikan di scope pertama (Induk scope) dan juga akan terlihat di dalam nested scope(anak scope). Itu karena nested scope termasuk dalam induk scope.

Di dalam peraturan pemberian identifier, Deklarasi tidak boleh memiliki identitas yang sama dalam satu ruang lingkup. dengan arti kita dapat mendirikan nama yang sama pada ruang lingkup yang berbeda, itu juga berlaku pada anak ruang lingkup, kita dapat mendirikan deklarasi dengan identitas yang sama meskipun identitas tersebut sudah digunakan di induk ruang lingkup yang merupakan lokal deklarasi dari luar ruang lingkup (induk ruang lingkup). Jika kita mendirikan deklarasi memiliki identitas yang sama dengan deklarasi luar ruang lingkup. Identitas dari deklarasi di dalam ruang lingkup akan menggantikan deklarasi luar ruang lingkup tersebut, hal ini disebut sebagai Shadowing atau Name Hiding.

#include <iostream>

using namespace std;

int main ()

{

int id = 1; //[id1 - Local Variable] Variabel didirikan dan terlihat setelah pernyataan ini.

{

int id = 2; //[ id2 - Local Variable] Variabel didirikan dan terlihat setelah pernyataan ini

cout<<id<<endl;

}//Variabel "id2" dihancurkan dan berakhir di sini

cout<<id<<endl;

return 0;

} //variabel "id1" dihancurkan dan berakhir di sini.

Variabel Eksternal

Variabel Eksternal adalah variabel yang dideklarasikan diluar fungsi yang bersifat global yang artinya dapat digunakan bersama-sama tanpa harus dideklarasikan berulang-ulang. Untuk pendeklarasian variabel ekternal ini, diluar dari fungsi main() atau di dalam fungsi lainya.

Contoh Variabel Eksternal atau Global

#include<conio.h>

#include<iostream>

using namespace std;

int a = 6; //--> deklarasi variabel eksternal

void lokal( ); //Protoripe Fungsi

int main( )

{

cout<<"Penggunaan Variabel Eksternal"<<endl;

cout<<"\nNilai didalam funsi main() = : "<<a;

lokal ( ); //--> pemanggilan fungsi local

cout<<"\nNilai Setelah penggilan fungsi local() = ";

cout<<a;

return 0;

}

void lokal()

{

a+=10;

}

Variabel Statis

Variabel Statis dapat berupa variabel local atau variabel eksternal Sifat variable statis ini mempunyai sifat antar lain.

Jika variabel statis bersifat local, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.

Jika variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama ditempat variabel statis dideklarasikan atau di file cpp lainya.

Jika tidak ada inisialisasi oleh pemrograman secara otomatis akan diberikan nilai awal nol.

Inisialisasi pada variable akan dibaca sekali dan tidak akan di selanjutnya jika compiler melewati baris tersebut. Dengan arti nilai pada variable itu tidak akan diulang/berubah sesuai inisialisasi pada variable tersebut.

Suatu variabel statis diperoleh dengan menambahkan kata-kunci static didepan penentu tipe data variabel.

Contoh :

static int namaIdentifier

static string namaIdentifier

Contoh Penggunaan Variabel Statis

#include<conio.h>

#include<iostream>

using namespace std;

void contohStatic(); //prototype fungsi

void variabelBiasa();

int main( )

{

//int a = 15;

cout<<"Perbedaan Static Variabel dan Variabel biasa"<<endl;

//cout<<"\nNilai didalam funsi main() = : "<<a;

contohStatic();

variabelBiasa();

contohStatic();

variabelBiasa();

contohStatic();

contohStatic();

variabelBiasa();

contohStatic();

variabelBiasa();

contohStatic();

variabelBiasa();

return 0;

}

void contohStatic(){

static int a = 10;

cout<<"Nilai a didalam Variabel Static( ) = "<<++a<<endl;

}

void variabelBiasa(){

int a = 5;

cout<<"Nilai a didalam Variabel Biasa( ) = "<<++a<<endl;

}

Cukup sekian dari belajarcpp.com semoga artikel ini bisa bermanfaat untuk kita semua. Mohon maaf atas kekuranganya dan Terima Kasih atas dukungan dan kunjungan anda ke BelajarCPP. Have a nice day.

Last update : 6 Februari 2016