Constructor adalah spesial member function, function di dalam class yang akan dieksekusi terlebih dahulu saat pembuatan instantce (object). Constructor biasanya digunakan untuk insialisasi member-member class atau melakukan persiapan lainya seperti membuka file, database atau membuat object dari class lainya.

Dalam mendirikan Constructor mirip seperti kita mendirikan function hanya saja constructor harus menjadi member dari class, harus memiliki identitas sama dengan class tersebut dan tidak memiliki return type.

Bentuk penulisan

class nama\_class{

public :

nama\_class(parameters){ //Constructor

…. //Definisi

}

};

Karakteristik Constructor

Constructor memiliki identitas sama seperti identitas dari class.

Constructor tidak memiliki return type.

Constructor harus berada dalam label public.

Constructor tidak bisa dipanggil dari luar class.

Contoh program

#include <iostream>

using namespace std;

class siswa{

public:

siswa(); //Constructor

};

siswa::siswa(){

cout<<"Constructor Terpanggil."<<endl; //Definisi Constructor

}

int main(){

siswa budi; //Pembuatan Object

return 0;

}

Di atas adalah contoh program yang mencontohkan bagaimana cara penggunaan constructor. Class “siswa” di atas mempunyai member function yang special yaitu constructor dengan nama yang sama seperti nama classnya, constructor tersebut tidak memiliki return type atau parameter dan akan bertugas untuk mengeluarkan keluaran saat pembuatan object dengan cout yang berisi “Constructor Terpanggil”.

Constructor dapat didirikan tanpa menggunakan parameter atau menggunakan parameter, Constructor yang tidak mempunyai parameter disebut sebagai Default Constructor. Di atas adalah contoh program constructor tanpa menggunakan parameter dan di bawah adalah contoh penggunaan constructor dengan parameter.

Contoh penulisan Construtor menggunakan parameter

class siswa{

public:

int nim;

string nama;

siswa(int iNim, string iNama){

nim=iNim;

nama=iNama;

}

};

Constructor dengan menggunakan parameter biasanya, saat mendirikan object, argument yang disertakan biasanya akan digunakan untuk inisialisasi member-member dari class. Dibawah adalah cara pembuatan object disertai pemberian argument atau biasa dibilang sebagai inisialisasi object.

Contohpenulsan inisialisasi object

siswa joko; //Default Constructor

siswa test = 123; //jika constructor hanya menyediakan satu parameter.

siswa budi(123, "Budi Roach"); //Jika constructor membutuhkan banyak argument, setiap argument dipisahkan degan tanda koma ( , ).

//atau anda uga bisa melakukan inisialisasi object dengan bentuk penulisan seperti di bawah ini.

siswa adi{123, "Adi Lord"};

//atau

siswa jodi={123, "Jodiii Teot"};

Contoh program

#include <iostream>

#include <string>

using namespace std;

class siswa{

public:

int nim;

string nama;

siswa(int iNim, string iNama){

nim=iNim;

nama=iNama;

}

void cetakData(){

cout<<"Nim = "<<nim<<endl;

cout<<"Nama = "<<nama<<endl;

cout<<endl;

}

};

int main(){

siswa budi(123, "Budi Roach");

siswa adi{123, "Adi Lord"};

siswa jodi={123, "Jodiii Teot"};

budi.cetakData();

adi.cetakData();

jodi.cetakData();

return 0;

}

**Overloading Constructor**

Pada beberapa bab sebelumnya dijelaskan mengenai function overloading, sama seperti constructor kita dapat melakukan overloading pada constructor, membuat banyak constructor dangan identitas yang sama tapi dengan ciri yang berbeda, perbedaan itu seperti parameter pada constructor.

Dengan melakukan Overloading pada constructor memungkinkan kita untuk menginisialisasi object disertakan memberikan argument secara fleksible, inisialisasi akan dapat menyesuaikan constructor mana yang akan dimasuki berdasarkan argument yang diberikan saat inisialisasi object.

Contoh penulisan

class siswa{

public:

int nim;

string nama;

siswa(int iNim){

//definisi 1

}

siswa(string iNama){

//Definisi 2

}

siswa(int iNim, string iNama){

//Definisi 3

}

};

Di atas ada contoh penulisan.dari Overloading constructor, 3 constructor dengan ciri mereka yang berbeda. Deklarasi Constructor nomer satu dibuat dengan menggunakan satu parameter yaitu int “iNim”, dan constructor ke dua menggunakan satu constructor yaitu string “iNama” dan ketiga menggunakan 2 buah parameter yaitu int “iNim” dan string “iNama”.

Dengan cara di atas kita dapat mudah membuat object dari class yang fleksibel dalam menangani argument yang di berikan. Dan dibawah ini adalah contoh program dari fleksibelitas sebuah class dalam membuat object yang dapat menerima argument secara fleksibel.

Contoh program

#include <iostream>

#include <string>

using namespace std;

class siswa{

private:

int nim;

string nama;

public:

siswa(int);

siswa(string);

siswa(int iNim, string iNama);

void cetak();

};

siswa::siswa(int iNim){

nim=iNim;

nama="Tidak ada nama";

cout<<endl<<"Constructor 1 Terpanggil"<<endl;

}

siswa::siswa(string iNama){

nama=iNama;

nim=0;

cout<<endl<<"Constructor 2 Terpanggil"<<endl;

}

siswa::siswa(int iNim, string iNama){

nim=iNim;

nama=iNama;

cout<<endl<<"Constructor 3 Terpanggil"<<endl;

}

void siswa::cetak(){

cout<<endl<<"Nim ="<<nim<<endl;

cout<<"Nama ="<<nama<<endl;

}

int main(){

siswa budi(123);

budi.cetak();

siswa adi("Adi Better");

adi.cetak();

siswa joko(345, "Joko Joker");

joko.cetak();

return 0;

}

**Default Constructor**

Default Constructor adalah onstructor yang tdak mempunyai parameter, atau menggunakan parameter tapi sebagai default parameter (parameter yang sudah diinisialisasi). Default constructor biasanya otomatis diciptakan oleh C++ saat kompilasi, jika class tersebut tidak memiliki constructor. Tapi jika kita telah membuat constructor di dalam class kita wajib untuk membuat default constructor.

Default constructor digunakan untuk mengerjakan tugas seperti constructor lainya (inisialisasi dll), di saat pengguna membuat object dari class tersebut tanpa menyertakan argument saat instantiation maka saat itulah default constructor berfungsi.

Contoh Penulisan

class siswa{

public:

int nim;

string nama;

siswa(){

//Definisi Default Constructor

}

siswa(int iNim){

//definisi 1

}

siswa(string iNama){

//Definisi 2

}

siswa(int iNim, string iNama){

//Definisi 3

}

};

Di saat kita sudah mendirikan constructor atau beberapa constructor yang disertai dengan parameter, default constructor otomatis tidak akan dibuat oleh compiler tidak seperti di saat kita mempunyai class tanpa construtor. Jika kita tidak mendirikan default constructor, program akan bermasalah ketika membuat object tanpa menyertai argument. Dan biasanya pesan error dari compiler akan menyarankan untuk membuat default constructor pada class.

Contoh program

#include <iostream>

#include <string>

using namespace std;

class siswa{

private:

int nim;

string nama;

public:

siswa();

siswa(int);

siswa(string);

siswa(int iNim, string iNama);

void cetak();

};

siswa::siswa(){

nim=0;

nama="Tidak ada nama";

cout<<endl<<"Default Constructor Terpanggil"<<endl;

}

siswa::siswa(int iNim){

nim=iNim;

nama="Tidak ada nama";

cout<<endl<<"Constructor 1 Terpanggil"<<endl;

}

siswa::siswa(string iNama){

nama=iNama;

nim=0;

cout<<endl<<"Constructor 2 Terpanggil"<<endl;

}

siswa::siswa(int iNim, string iNama){

nim=iNim;

nama=iNama;

cout<<endl<<"Constructor 3 Terpanggil"<<endl;

}

void siswa::cetak(){

cout<<endl<<"Nim ="<<nim<<endl;

cout<<"Nama ="<<nama<<endl;

}

int main(){

siswa jodi;

jodi.cetak();

siswa budi(123);

budi.cetak();

siswa adi("Adi Better");

adi.cetak();

siswa joko(345, "Joko Joker");

joko.cetak();

return 0;

}