**MODUL 8**

**PEMROGRAMAN BERORIENTASI OBJEK**

**Pengantar**

Pemrograman Berorientasi Objek sebenarnya bukanlah bahasa pemrograman baru, tetapi jalan baru untuk berpikir dan merancang aplikasi yang dapat membantu memecahkan persoalan mengenai pengembangan perangkat lunak. Pemrograman berorientasi objek disusun dan dipahami oleh ilmuwan yang memandang dunia sebagai populasi objek yang berinteraksi dengan yang lain. Prosedur yang digunakan dalam objek dipandang sebagai kepentingan kedua karena tergantung pada objek itu sendiri. Tentunya hal tersebut berbeda dengan pemrograman terstruktur. Pemrograman terstruktur mempunyai sifat pemisahan data dengan kode yang mengolahnya.

Pemrograman Berorientasi Objek (PBO) adalah metode pemrograman yang meniru cara kita memperlakukan sesuatu(benda). Ada tiga karakteristik bahasa Pemrograman Berorientasi Objek, yaitu:

1. Pengkapsulan (*Encapsulation*) : mengkombinasikan suatu struktur dengan fungsi yang memanipulasinya untuk membentuk tipe data baru yaitu kelas (class).
2. Pewarisan (*Inheritance*) : mendefinisikan suatu kelas dan kemudian menggunakannya untuk membangun hirarki kelas turunan, yang mana masing-masing turunan mewarisi semua akses kode maupun data kelas dasarnya.
3. Polimorphisme (*Polymorphism*) : memberikan satu aksi untuk satu nama yang dipakai bersama pada satu hirarki kelas, yang mana masing-masing kelas hirarki menerapkan cara yang sesuai dengan dirinya.

**Struktur dan kelas**

Dalam C++, tipe data struktur yang dideklarasikan dengan kata kunci struct, dapat mempunyai komponen dengan sembarang tipe data, baik tipe data dasar maupun tipe data turunan, termasuk fungsi. Dengan kemampuan ini, tipe data struktur menjadi sangat berdaya guna. Misalnya, kita ingin membentuk tipe data struktur yang namanya kotak. Maka dapat dideklarasikan sebagai berikut:

struct tkotak

{

double panjang;

double lebar;

}; tkotak kotak;

Untuk memberi nilai ukuran kotak tersebut, kita dapat menggunakan perintahperintah

ini:

kotak.panjang = 10;

kotak.lebar = 7;

Untuk memberi nilai panjang dan lebar kotak, salah satu caranya adalah seperti di atas. Cara lain untuk memberi nilai panjang dan lebar adalah dengan membentuk suatu fungsi. Karena fungsi ini hanya digunakan untuk memberi nilai data panjang dan lebar suatu kotak, tentunya fungsi ini khusus milik objek kotak, sehingga harus dianggap sebagai anggota struktur kotak. C++ sebagai bahasa pemrograman dapat mendefinisikan anggota tipe struktur yang berupa fungsi. Dengan menambah fungsi tersebut, maka struktur kotak menjadi lebih jelas bentuknya.

struct tkotak {

double panjang;

double lebar;

void SetUkuran(double pj, double lb)

{

panjang = pj;

lebar = lb; }; };

tkotak kotak;

dengan tipe struktur kotak seperti itu, untuk memberi nilai panjang dan lebar hanya dengan memanggil fungsi SetUkuran()

kotak.SetUkuran(10,7);

Selain punya ukuran panjang dan lebar, kotak juga mempunyai keliling dan luas. Dengan demikian, kita dapat memasukkan fungsi untuk menghitung keliling dan luas ke dalam struktur kotak. Sebagai catatan, bahwa definisi fungsi yang menjadi anggota struktur dapat ditempatkan di luar tubuh struktur. Dengan cara ini maka deklarasi struktur kotak menjadi seperti berikut:

struct tkotak {

double panjang;

double lebar;

void SetUkuran(double pj, double lb);

double Keliling();

double Luas(); };

tkotak kotak;

contoh penerapan struktur kotak dapat dilihat dalam program berikut:

#include<iostream.h>

#include<conio.h>

struct tkotak {

double panjang;

double lebar;

void SetUkuran(double pj, double lb);

double Keliling();

double Luas();

};

int main() {

tkotak kotak;

kotak.SetUkuran(10,7);

cout<<"Panjang : "<<kotak.panjang<<endl;

cout<<"Lebar : "<<kotak.lebar<<endl;

cout<<"Keliling : "<<kotak.Keliling()<<endl;

cout<<"Luas : "<<kotak.Luas()<<endl;

getch();

return 0;

}

void tkotak::SetUkuran(double pj, double lb) {

panjang = pj;

lebar = lb;

}

double tkotak::Keliling() {

return 2\*(panjang+lebar); }

double tkotak::Luas() {

return panjang\*lebar; }

**Tampilan Output:**

Panjang : 10

Lebar : 7

Keliling : 34

Luas : 70

Bentuk program di atas, adalah contoh gaya pemrograman berorientasi prosedur (terstruktur) yang sudah mengubah pola pikirnya menjadi berorientasi objek. Dalam pemrograman berorientasi objek, jika kita telah menentukan suatu objek tertentu, maka objek tersebut kita definisikan dalam bentuk tipe baru yang namanya kelas.

Tipe data kelas didefinisikan dengan kata kunci (*keyword*) class, yang merupakan generalisasi dari pernyataan struct. Pernyataan struct secara umum digantikan dengan pernyataan class. Jika objek kotak dideklarasikan dalam bentuk kelas, maka deklarasinya mirip dengan struktur.

class tkotak {

double panjang;

double lebar;

public:

void SetUkuran(double pj, double lb);

double Keliling(); double Luas();

};

tkotak kotak;

Dalam deklarasi kelas tersebut, muncul kata public. Data atau fungsi yang dideklarasikan di bawah kata kunci public mempunyai sifat dapat diakses dari luar kelas secara langsung. Dalam deklarasi tersebut, variabel panjang dan lebar tidak bersifat public, sehingga tidak dapat diakses secara langsung dari luar kelas. Dengan demikian perintah-perintah di bawah ini tidak dapat dijalankan.

kotak.panjang = 10;

kotak.lebar = 7;

cout<<”Panjang : ”<<kotak.panjang<<endl;

cout<<”Lebar : “<<kotak.lebar<<endl;

Inilah yang membedakan struktur dan kelas. Dalam kelas, masing-masing data dan fungsi anggota diberi sifat tertentu. *Jika semua anggota kelas bersifat* *public, maka kelas sama dengan struktur*.

Untuk dapat mengakses data panjang dan lebar pada kelas tkotak harus dilakukan oleh fungsi yang menjadi anggota kelas dan bersifat public.

Pada deklarasi kelas tkotak, satu-satunya jalan untuk memberi nilai panjang dan lebar adalah dengan menggunakan fungsi SetUkuran(). Untuk mengambil nilai panjang dan lebar juga harus dilakukan oleh fungsi yang menjadi anggota kelas. Misalnya, kita definisikan fungsi GetPanjang() dan GetLebar() untuk mengambil nilai panjang dan lebar. Sebagai contoh:

//program class

#include<iostream.h>

#include<conio.h>

class tkotak

{

double panjang;

double lebar;

public:

void SetUkuran(double pj, double lb);

double Keliling();

double Luas();

double GetPanjang();

double GetLebar();

};

int main()

{

tkotak kotak;

kotak.SetUkuran(10,7);

cout<<"Panjang : "<<kotak.GetPanjang()<<endl;

cout<<"Lebar : "<<kotak.GetLebar()<<endl;

cout<<"Keliling : "<<kotak.Keliling()<<endl;

cout<<"Luas : "<<kotak.Luas()<<endl;

getch();

return 0;

}

void tkotak::SetUkuran(double pj, double lb) {

panjang = pj;

lebar = lb; }

double tkotak::Keliling() {

return 2\*(panjang+lebar); }

double tkotak::Luas()

{

return panjang\*lebar;

}

double tkotak::GetPanjang()

{

return panjang;

}

double tkotak::GetLebar() {

return lebar;

}

**Tampilan Output:**

Panjang : 10

Lebar : 7

Keliling : 34

Luas : 70

Dapat dilihat dari contoh program, bentuk pendefinisian kelas adalah sebagai berikut:

Tipe Nama\_Kelas::NamaFungsi() {

IsiFungsi }

Untuk mendefinisikan variabel kelas, digunakan deklarasi :

Nama\_Kelas Nama\_Variabel;

Contoh :

Tkotak kotak;

**Pengkapsulan(Encapsulation)**

Salah satu keistimewaan C++ adalah pengkapsulan. Pengkapsulan adalah mengkombinasikan suatu struktur dengan fungsi yang memanipulasinya untuk membentuk tipe data baru yaitu kelas(class). Kelas akan menutup rapat baik data maupun kode. Akses item di dalam kelas dikendalikan.

Pengendalian ini tidak hanya berupa data tetapi juga kode. Saat kelas akan digunakan, kelas harus sudah dideklarasikan. Yang penting, pemakai kelas mengetahui deskripsi kelas, tetapi bukan implementasinya. Bagi pemakai, detail internal kelas tidak penting. Konsep ini disebut penyembunyian informasi (*information hiding*). Untuk menggunakan kelas, kita perlu mengetahui sesuatu tentangnya. Kita perlu mengetahui fungsi apa yang bisa digunakan dan data apa saja yang dapat diakses. Fungsi yang dapat digunakan dan data yang dapat diakses disebut antarmuka pemakai (user interface). Antarmuka pemakai menceritakan bagaimana kelas berperilaku, bukan bagaimana kelas dibuat. Kita tidak perlu mengetahui implementasi kelas. Sekali kelas dibuat, kita bisa memakainya berulang-ulang. Bagi pandangan pemakai, kelas adalah kotak hitam dengan perilaku tertentu.

**Kendali Akses terhadap Kelas**

Tugas kelas adalah untuk menyembunyikan informasi yang tidak diperlukan oleh pemakai. Ada tiga macam pemakai kelas:

1. kelas itu sendiri
2. pemakai umum
3. kelas turunan Setiap macam pemakai mempunyai hak aksesnya masingmasing.

Hak akses ini ditandai dengan kenampakan anggota kelas. Kelas pada C++ menawarkan tiga

aras kenampakan anggota kelas (baik anggota data maupun fungsi anggota):

1. **private**

Anggota kelas private mempunyai kendali akses yang paling ketat. Dalam bagian private, hanya fungsi anggota dari kelas itu yang dapat mengakses anggota private atau kelas yang dideklarasikan sebagai teman (friend).

1. **Public**

Dalam bagian public, anggotanya dapat diakses oleh fungsi anggota kelas itu sendiri, instance kelas, fungsi anggota kelas turunan. Suatu kelas agar bisa diakses dari luar kelas, misalnya dalam fungsi main(), perlu mempunyai hak akses publik. Hak akses ini yang biasanya digunakan sebagai perantara antara kelas dengan dunia luar.

1. **Protected**

Suatu kelas dapat dibuat berdasarkan kelas lain. Kelas baru ini mewarisi sifat-sifat dari kelas dasarnya. Dengan cara ini bisa dibentuk kelas turunan dari beberapa tingkat kelas dasar. Bila pada kelas dasar mempunyai anggota dengan hak akses terproteksi, maka anggota kelas ini akan dapat juga diakses oleh kelas turunannya. Anggota kelas terproteksi dibentuk dengan didahului kata kunci protected. Pada bagian protected, hanya fungsi anggota dari kelas dan kelas-kelas turunannya yang dapat mengakses anggota.

Contoh program :

//File persegiPJ.Cpp

#include <iostream.h>

class PersegiPanjang {

protected :

int panjang;

int lebar;

public :

PersegiPanjang(int pj, int lb)

{

panjang = pj; lebar

= lb; }

int Panjang() {

return panjang;

}

int Lebar() {

return lebar; }

int Keliling() {

return 2\*(panjang+lebar); }

int Luas()

{

return panjang\*lebar;

}

void Skala(float sk) {

panjang \*= sk;

lebar \*= sk; }

void TulisInfo() {

cout<<"Panjang : "<<panjang<<endl <<"Lebar

:"<<lebar<<endl <<"Keliling :"<<Keliling()<<endl

<<"Luas :"<<Luas()<<endl; } };

Simpan file di atas dengan nama **PersegiPJ.Cpp**

Kemudian lanjutkan pembuatan program berikut pada halaman baru:

// penggunaan kelas persegi panjang

#include "PersegiPJ.Cpp"

void main()

{

PersegiPanjang pp1(10,5);

pp1.Skala(2);

pp1.TulisInfo();

getch();

}