PEMROGRAMAN BERORIENTASI OBJEK

OBJECT

Telkom University

Budhi Irawan, S.Si, M.T

OBJEK SEBAGAI PARAMETER DALAM SEBUAH FUNGSI

- Dalam C++, objek dapat juga berperan sebagai parameter dalam pendefinisian sebuah fungsi. Objek ini akan dilewatkan secara standar yaitu dengan menggunakan metoda pass by value (dikirimkan berdasar nilai)
- Hal ini sebenarnya bertujuan untuk mengakses suatu data (yang bersifat public) didalam sebuah kelas dimana pemrosesannyadilakukan didalam fungsi luar (bukan member function)

OBJEK SEBAGAI PARAMETER DALAM SEBUAH FUNGSI

```
#include <iostream>

using namespace std;

class CONTOH {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  int GetX() {
    return X;
  }
};

int Kuadrat(CONTOH A, int N) {
  A.SetX(N);
  return (A.GetX() * A.GetX());
}

int main() {

  CONTOH O;

  // Memanggil fungsi Kuadrat
  cout<<"Kuadrat dari 10 adalah: "
      <<Kuadrat(O,10);

  return 0;
}
```

OBJEK SEBAGAI NILAI KEMBALIAN DALAM SEBUAH FUNGSI

- Selain menjadi parameter, sebuah objek juga dapat digunakan sebagai nilai kembalian dari seuatu fungsi.
- Didalam fungsi akan tercipta satu objek sementara (temporary) yang kemudian akan dibebaskan kembali sesaat setelah proses pengembalian nilai.

OBJEK SEBAGAI NILAI KEMBALIAN DALAM SEBUAH FUNGSI

```
#include <iostream>

using namespace std;

class CONTOH {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  int GetX() {
    return X;
  }
};

CONTOH MyFunc() {
  CONTOH A;

  A.SetX(30);

  // Mengembalikan nilai berupa objek
  return A;
}

int main() {

  CONTOH O;

  // Memanggil fungsi MyFunc()
  // dan nilainya dimasukkan
  // ke dalam objek O
  O = MyFunc();

  cout<<"Nilai X di dalam O adalah: "
      <<O.GetX();

  return 0;
}
```

FRIEND FUNCTION

- Dalam C++, kita diizinkan untuk membuat fungsi luar (bukan member function) yang dapat mengakses bagian private suatu kelas.
- Fungsi seperti ini dinamakan dengan Friend Function.
- Cara untuk mendefinisikan fungsi tersebut sama seperti fungsi-fungsi biasa, hanya saja fungsi ini perlu dideklarasikan terlebih dahulu didalam kelas yang bersangkutan, yaitu dengan menggunakan kata kunci friend

FRIEND FUNCTION

```
#include <iostream>

using namespace std;

class CONTOH{
  int X, Y;
public:
  void SetXY(int XX, int YY) {
    X = XX;
    Y = YY;
  }
  // Deklarasi friend function
  friend int KALI(CONTOH A);
};

// Mendefinisikan fungsi KALI()
// yang bukan termasuk member function
// dari kelas CONTOH
int KALI(CONTOH A) {
  // Mengakses bagian private
  // secara langsung dari kelas CONTOH
  return (A.X * A.Y);
}

// Fungsi utama
int main() {

  // Melakukan instansiasi kelas CONTOH
  CONTOH O;

  // Mengisi nilai X=20 dan Y=3
  O.SetXY(20, 3);

  cout<<"Hasil kali: "<<KALI(O);

  return 0;
}
```

FRIEND FUNCTION

- Seperti yang kita lihat diatas bahwa fungsi KALI() bukan merupakan fungsi anggota (member function) dari kelas CONTOH, tapi fungsi tersebut dapat mengakses bagian private (dalam hal ini X dan Y) dari kelas tersebut.
- Hal ini dapat terjadi karena fungsi tersebut berlaku sebagai teman (friend) dari kelas CONTOH.

FRIEND CLASS

- Friend Class yaitu kelas yang dapat mengakses semua data (termasuk bagian private) dari kelas lain.
- Kedua kelas tersebut masing-masing adalah kedua kelas yang terpisah dan tidak ada hubungan turunan sama sekali.
- Sama halnya seperti pada friend function, friend class juga harus dilakukan deklarasi terlebih dahulu, yaitu dengan menggunakan keyword friend.

FRIEND CLASS

```
#include <iostream>

using namespace std;

class KESATU {
  int X, Y;
public:
  KESATU(int XX, int YY) {
     X = XX;
     Y = YY;
  }
  // Mendeklarasikan friend class
  friend class KEDUA;
};

// Definisi dari kelas KEDUA
class KEDUA {
  // …
public:
  int Kali(KESATU A);
  // …
};

int KEDUA::Kali(KESATU A) {
  // Mengakses data private
  // pada kelas KESATU
  return (A.X * A.Y);
};

int main() {

  // Melakukan instansiasi dari kelas KESATU
  KESATU O(40, 3);

  // Melakukan instansiasi dari kelas KEDUA
  KEDUA P;

  // Memanggil fungsi Kali()
  // yang terdapat pada kelas KEDUA
  cout<<"Hasil kali: "<<P.Kali(O);
```

FRIEND CLASS

- Seperti yang kita lihat diatas bahwa dengan menjadikan kelass KEDUA sebagai friend class dari KESATU, maka kelas KEDUA mempunyai hak akses penuh terhadap semua data yang terdapat pada kelass KESATU meskipun data tersebut bersifat private.

POINTER KE OBJEK

- Seperti halnya pada tipe data dasar maupun ***tipe data bentukan***, ***pointer*** juga dapat menunjuk ke ***tipe kelas***.
- Sama seperti pada ***pointer*** yang menunjuk ke ***tipe struktur***, ***pointer ke objek*** juga akan mengakses ***data*** atau ***fungsi*** kedalam ***kelas*** dengan menggunakan ***operator ->*** dan bukanlah menggunakan ***operator titik (.)*** seperti pada ***instance*** yang bukan merupakan ***pointer***.

POINTER KE OBJEK

```
#include <iostream>

using namespace std;

// Membuat kelas
class CONTOH {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  void ShowX() {
    cout<<"Nilai X: "<<X<<endl;
  }
};

// Fungsi utama
int main() {

  // Mendeklarasikan pointer yang menunjuk ke kelas CONTOH
  CONTOH *P;

  // Mengalokasikan memori untuk objek  dari kelas CONTOH
  P = new CONTOH; // P menunjuk ke alamat yang baru dialokasikan

  // Memanggil fungsi-fungsi milik kelas CONTOH, yaitu dengan operator ->
  P->SetX(100);     // Ingat, bukan menggunakan titik, karena P adalah pointer
  P->ShowX();

  // Mendealokasikan memori
  delete P;

  return 0;
}
```

POINTER THIS

- ***Pointer This*** adalah ***pointer*** secara otomatis yang dilewatkan setiap kali ***objek*** dibuat dalam memori dan akan mewakili ***nama kelas*** yang bersangkutan.
- ***Pointer This*** pada umumnya digunakan ketika kita mendefinisikan ***fungsi*** yang nama ***parameternya*** sama dengan ***nama data anggota***. Contoh :

```
class CONTOH {
    int x; // X berperan sebagai nama data
public:
    void SetX(int X) { // X berperan sebagai parameter
        this->x = x;
    }
}
```

POINTER THIS

- Pada kode diatas, ***nama data (X)*** sama dengan ***nama parameter*** yang dilewatkan kedalam ***fungsi SetX().***
- Dalam situasi semacam ini, kita perlu menggunakan ***pointer This*** untuk menunjukan bahwa ***X*** yang kita gunakan adalah ***X*** yang berperan sebagai ***data anggota kelas***, bukan ***X*** yang berperan sebagai parameter, dengan demikian kode :

```
this->X = X;
```

sama artinya dengan

```
X milik kelas CONTOH = X parameter;
```

POINTER THIS -I

```
#include <iostream>
#include <cstring>

using namespace std;

enum TIPETRANSMISI {automatic, manual};

// Membuat kelas MOBIL
class MOBIL {
  char merk[25];
  TIPETRANSMISI transmisi;
  int tahun;
  int silinder;
  char nopolisi[11];
  char warna[15];
public:
  void SetMobil(char *merk,
                TIPETRANSMISI transmisi,
                int tahun,
                int silinder,
                char *nopolisi,
                char *warna) {
    strcpy(this->merk, merk);
    this->transmisi = transmisi;
    this->tahun = tahun;
    this->silinder = silinder;
    strcpy(this->nopolisi, nopolisi);
    strcpy(this->warna, warna);
  }
```

POINTER THIS -I

```
void ShowInfoMobil() {
    cout<<"Merk  \t: "<<merk<<endl;
    cout<<"Transmisi \t: ";
    switch (transmisi) {
      case 0: cout<<"Automatic\n"; break;
      case 1: cout<<"Manual\n"; break;
    }
    cout<<"Tahun  \t: "<<tahun<<endl;
    cout<<"Silinder \t: "<<silinder<<endl;
    cout<<"No Polisi\t: "<<nopolisi<<endl;
    cout<<"Warna  \t: "<<warna<<endl;
  }
};

// Fungsi utama
int main() {

  // Melakukan instansiasi terhadap kelas MOBIL
  MOBIL M;
```

POINTER THIS -I

```
// Menentukan data ke dalam objek M
  M.SetMobil(
     (char*) "Ferrari 430 Spider",
     automatic,
     2012,
     4500,
     (char*) "D 1212 SK",
     (char*) "Kuning"
  );

  // Menampilkan data pada objek M
  M.ShowInfoMobil();

  return 0;
}
```

POINTER THIS -2

```
#include <iostream>

using namespace std;

// Membuat kelas BALOK
class BALOK {
  double p;
  double l;
  double t;
public:
  void SetData(double p, double l, double t) {
    this->p = p;
    this->l = l;
    this->t = t;
  }

  double GetVolume() {
    return (p * l * t);
  }

  void ShowVolume() {
    cout<<"Volume balok: "
        <<GetVolume();
  }
};

int main() {
  // Mendeklarasikan objek b
  // bertipe BALOK
  BALOK b;

  // Memanggil fungsi SetData()
  b.SetData(5.0, 3.0, 2.0);

  // Memanggil fungsi ShowVolume()
  b.ShowVolume();

  return 0;
```