PEMROGRAMAN BERORIENTASI OBJEK

STRUCTURE & CLASS

Telkom University

Budhi Irawan, S.Si, M.T

PENDAHULUAN

- Dalam C++, ***struktur*** dan ***kelas*** adalah dua hal yang sebenarnya saling memiliki hubungan, artinya dapat dibuat ***kelas*** dengan menggunakan ***keyword*** ***`struct`*** seperti halnya pada saat membuat ***struktur***.
- Dalam praktiknya, hampir semua programmer C++ menggunakan ***keyword*** ***`class`*** untuk membuat sebuah ***kelas***.

MEMBUAT KELAS MENGGUNAKAN KEYWORD CLASS

```
#include <iostream>

using namespace std;

// Membuat kelas DASAR yang didalamnya terdapat data bersifat protected

class DASAR {
protected:
  int X;
public:
  // Inisialisasi nilai X dengan nilai 10
  DASAR() { X = 10; }
  //...
};

// Membuat kelas TURUNAN yang merupakan turunan dari kelas DASAR
class TURUNAN: public DASAR {
  int Y, hasil;
public:
  void SetY(int YY) {
    Y = YY;
  }
  void KaliXY() {
    // Menggunakan nilai X dari kelas DASAR
    hasil = X * Y;
  }
  int GetHasil() {
    return hasil;
  }
};
```

MEMBUAT KELAS MENGGUNAKAN KEYWORD CLASS

```
// Fungsi utama
int main() {

  DASAR A;
  TURUNAN B;

  B.SetY(5);
  B.KaliXY();
  cout<<"\nHasil X kali Y: "<<B.GetHasil();

  return 0;
}
```

MEMBUAT KELAS MENGGUNAKAN KEYWORD STRUCT

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

using namespace std;

enum TIPETRANSMISI {automatic, manual};

// Membuat kelas MOBIL
struct MOBIL {
  char merk[25];
  TIPETRANSMISI transmisi;
  int tahun;
  int silinder;
  char nopolisi[11];
  char warna[15];
public:
  void SetMobil(char *pmerk,
                TIPETRANSMISI ptransmisi,
                int ptahun,
                int psilinder,
                char *pnopolisi,
                char *pwarna) {
    strcpy(merk, pmerk);
    transmisi = ptransmisi;
    tahun = ptahun;
    silinder = psilinder;
    strcpy(nopolisi, pnopolisi);
    strcpy(warna, pwarna);
  }
```

MEMBUAT KELAS MENGGUNAKAN KEYWORD STRUCT

```
 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;
```

MEMBUAT KELAS MENGGUNAKAN KEYWORD STRUCT

```
// Menentukan data ke dalam objek M
  M.SetMobil(
     (char*) "Ferrari 599XX ",
     automatic,
     2016,
     9000,
     (char*) "D 1212 SH",
     (char*) "Merah"
  );

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

  return 0;
}
```

STRUCT DAN CLASS

- Didalam C++, pada saat didefinisikan sebuah ***struktur,*** sebenarnya kita mendefinisikan sebuah ***kelas*** sehingga kita bebas untuk memasukan ***fungsi-fungsi*** kedalamnya
- Walaupun demikian para programmer C++ pada umumnya tetap menggunakan keyword ***`class`*** untuk membuat ***kelas*** dan menggunakan kata kunci ***`struct`*** untuk membuat ***struktur*** seperti yang terdapat pada bahasa C atau pascal

STRUCT DAN CLASS

- Perbedaan antara ***keyword*** ***`struct`*** dan ***`class`*** adalah dalam hal ***tingkat akses*** default dari ***data*** maupun ***fungsi*** yang terdapat didalamnya.
- Apabila tidak menuliskan ***tingkat akses*** secara eksplisit pada saat mendeklarasikan ***data*** maupun ***fungsi*** dalam ***struktur,*** maka ***data*** dan ***fungsi*** dalam ***struktur*** tersebut akan dianggap sebagai ***data public***.
- Hal ini tentu berbeda dengan ***kelas***, yang mana dalam ***kelas*** jika tidak menuliskan ***tingkat akses***, maka ***data*** dan ***fungsi*** akan dianggap sebagai ***data private***.

STRUCT

```
struct contoh {
    int x;      // x bersifat public
};
```

- Pada kode diatas secara default data x akan ***bersifat public***.
- Apabila kita ingin membuatnya menjadi ***private*** maka, kita harus secara eksplisit menuliskan ***tingkat akses private*** kedalamnya sehingga kodenya akan tampak seperti dibawah ini :

```
struct contoh {
private:
    int x;      // sekarang x bersifat private
};
```

CLASS

- Sekarang perhatikan pendefinisian ***kelas*** yang menggunakan ***keyword class*** berikut ini :

```
class contoh {
    int x;      // x bersifat private
};
```

- Pada kode kali ini, secara default data x akan bersifat ***private***, dan bila akan menjadikannya sebagai ***data public*** maka perlu ditambahkan ***keyword public*** pada kode tersebut.

```
class contoh {
public:
    int x;      // sekarang x bersifat public
};
```

ANGGOTA KELAS YANG BERSIFAT STATIS

- Didalam C++, anggota-anggota dari sebuah ***kelas (data dan fungsi)*** dapat dijadikan sebagai ***statis*** yaitu dengan menambahkan ***keyword*** `static` didepan deklarasinya.
- Bagian ini akan menjelaskan bagaimana cara membuat data maupun ***fungsi statis*** dan alasan-alasan yang mendasarinya.

DATA STATIS

- Normalnya setiap ***data*** yang terdapat dalam ***kelas*** akan dialokasikan untuk setiap ***objek (instance)*** yang ada.
- Namun jika dideklarasikan sebagai ***statis*** maka dalam memori hanya akan terdapat satu ***data*** yang dipakai untuk berapapun ***instance*** yang ada.
- Ini berarti bahwa semua ***instance*** akan menggunakan atau berbagi satu data yang sama.
- Data statis ini akan diinisialisasi dengan ***0*** sebelum ***instance pertama*** dibuat.

DATA STATIS

- Perlu diperhatikan bahwa ketika kita mendeklarasikan ***data statis*** dalam ***kelas***, sebenarnya kita tidak mendefinisikan ***data*** tersebut. Ini berarti bahwa sebelumnya ***data*** tersebut tidak tersimpan dalam ***memori***.
- Untuk itu kita harus menyediakan ***definisi global*** untuk ***data*** tersebut dimanapun letaknya asalkan berada diluar ***kelas*** yang bersangkutan.
- Hal ini dilakukan dengan cara mendeklarasikan ulang ***data statis*** dengan menggunakan ***operator scope resolution*** (::) untuk mengidentifikasi ***kelas*** sipemilik data.

DATA STATIS

```
#include <iostream>

using namespace std;

class CONTOH {
  static int X;    // variabel statis
  int Y;           // variabel non-statis
public:
  // Constructor kelas CONTOH
  CONTOH(int XX, int YY) {
     X = XX;
     Y = YY;
  }
  // Fungsi untuk menampilkan nilai X dan Y
  void ShowXY() {
     cout<<"Nilai X: "<<X<<endl;
     cout<<"Nilai Y: "<<Y<<endl;
     cout<<endl;
  }
};
// Mendeklarasikan ulang variabel X di luar kelas
int CONTOH::X;
```

DATA STATIS

```
// Fungsi utama
int main() {

  // Membuat objek A dengan X=10 dan Y=10
  CONTOH A(10, 10);

  // Menampilkan nilai X dan Y dari objek A
  cout<<"Di dalam objek A"<<endl;

  A.ShowXY();

  // Membuat objek B dengan X=50 dan Y=50
  CONTOH B(50, 50);

  // Menampilkan nilai X dan Y dari objek B
  cout<<"Di dalam objek B"<<endl;

  B.ShowXY();

  // Menampilkan kembali nilai X dan Y  dari objek A
  cout<<"Di dalam objek A"<<endl;

  A.ShowXY();

  return 0;
}
```

DATA STATIS

- Seperti kita lihat pada kode diatas, ***variabel X*** yang dideklarasikan sebagai ***data statis*** maka dari itu ketika ***nilai X*** pada suatu ***objek (instance)*** berubah, maka itu berarti mengubah semua ***nilai X*** dari ***objek-objek*** lainnya.
- Pada kasus ini, ketika kita mengubah ***nilai X*** untuk ***objek B*** dengan ***nilai 50***, ***nilai X*** yang terdapat pada ***objek A*** juga akan berubah menjadi ***50.***

DATA STATIS

- ***Variabel Y*** yang merupakan ***variabel normal (non statis)***, nilainya tidak akan berubah, hal ini karena sebenarnya didalam program tersebut terdapat ***dua buah variabel Y***, yaitu ***Y*** milik ***objek A*** dan ***Y*** milik ***objek B***.
- Ini tentu berbeda dengan ***variabel X***, yang sebenarnya hanya ada satu, akan tetapi digunakan oleh ***dua buah objek,*** yaitu ***A*** dan ***B***.

FUNGSI STATIS

- Sama seperti halnya ***data*** atau ***variabel*** dari sebuah ***kelas***, ***fungsi-fungsi*** yang terdapat didalam suatu ***kelas*** juga dapat dijadikan sebagai ***fungsi statis***.
- Caranya sama, yaitu dengan menambahkan ***keyword*** `static` didepannya, namun dalam melakukan hal ini perlu sekali untuk memperhatikan hal-hal berikut :
  - ***Fungsi statis*** yang terdapat didalam ***kelas*** tersebut tidak memiliki ***pointer*** `this`
  - ***Fungsi statis*** tidak boleh bersifat ***virtual***
  - Tidak boleh terdapat dua (atau lebih) ***fungsi*** yang sama walaupun satu bersifat ***statis*** dan yang lainnya ***non statis***.

FUNGSI STATIS

```
#include <iostream>

using namespace std;

class CONTOH {
  static int X;
public:
  static void Inisialisasi(int XX) {
    X = XX;
  }
  void ShowX() {
    cout<<"Nilai X: "<<X<<endl;
  }
};

// Mendefinisikan X
int CONTOH::X;

// Fungsi utama
int main() {

  // Memanggil fungsi Inisialisasi() sebelum sebuah objek/instance dibuat
  CONTOH::Inisialisasi(25);

  // Melakukan instansiasi terhadap kelas CONTOH dengan nama instance A
  CONTOH A;

  // Memanggil fungsi ShowX() milik kelas CONTOH
  A.ShowX();

  return 0;
}
```