A. KOMPETENSI

1. Memahami pengertian kelas dan objek
2. Mampu mendefinisikan kelas
3. Mampu mendeklarasikan objek
4. Memahami constructor dan destructor

B. ALAT DAN BAHAN

1. PC/ Laptop
2. Flowchart application
3. C++ (atau bahasa pemrograman lain yang dikuasai)
4. Labsheet

C. KAJIAN TEORI

Kelas merupakan blueprint atau prottipe/kerangka yang mendefinisikan variabel-variabel (data) dan fungsi-fungsi (perilaku) umum dari sebuah objek tertentu. Dalam bahasa pemrograman, kelas tidak jauh berbedda dengan tipe data sederhana. Perbedaannya, tipe data sederhana digunakan untuk mendeklarasikan variabel 'normal', sedangkan kelas digunakan untuk mendeklarasikan sebuah variabel yang berupa objek.

Pada saat kelas baru dibuat, berarti tipe data baru telah didefinisikan. Sekali didefinisikan mata tipe data baru dapat digunakan untuk membuat suatu objek dari tipe tersebut.

Dengan kata lain, kelas merupakan pola (*template*) untuk pembuatan objek, dan objek adalah wujud nyata dari sebuah kelas. Contoh, manusia adalah kelas, sedangkan contoh objek atau wujud nyata dari kelas manusia adalah si Udin, Tono, Dewi, dll.

Bentuk umum pembuatan kelas:

```
Class nama_kelas {
   Access_specifier1:
    Data_members;
    Member_function;
          .....
   Access_specifier2:
     Data_members;
     Member_function;
     .....
....
};
```

Untuk mendefinisikan atau membuat implementasi fungsi-fungsi yang terdapat dalam sebuah kelas, digunakan operator ::.

Bentuk umum pendefinisian fungsi:

```
Tipe_data nama_kelas::nama_fungsi(daftara-parameter) {
   Statemen_yang_akan_dilakukan;
   ....
}
```

Untuk mengakses data atau fungsi yang terdapat di dalam kelas tersebut, digunakan tanda titik.
Bentuk umum pengaksesan data atau fungsi dari sebuah kelas:

```
Nama_instance.data
Atau
Nama_instance.nama_fungsi(daftar_parameter)
```

Contoh konsep pembuatan kelas dapat diperhatikan pada kode program berikut:
Contoh 1:

```
#include <iostream>

using namespace std;

class CONTOH {
  int X;
public:
  // Mendeklarasikan fungsi set_X()
  void set_X(int XX);
  // Mendeklarasikan fungsi get_X()
  int get_X();
};

// Bagian implementasi dari fungsi set_X()
void CONTOH::set_X(int XX) {
  X = XX;
}

// Bagian implementasi dari fungsi get_X()
int CONTOH::get_X() {
  return X;
}

// Fungsi utama
int main() {

  // Membuat instance dari kelas CONTOH
  // dengan nama ob
  CONTOH ob;

  // Mengakses fungsi set_X()
  // yang terdapat pada kelas CONTOH
  ob.set_X(100);

  // Mengakses fungsi get_X()
  // yang terdapat pada kelas CONTOH
  cout<<ob.get_X();

  return 0;
}
```

Hasil:

Constuctor dan destructor

Constructor adalah sebuah fungsi yang otomatis akan dipanggil setiap kali melakukan instansiasi terhada suatu kelas. Constructor digunakan untuk melakukan inisialisasi nilai dari data-data yang terdapat di dalam kelas bersangkutan. Seperti fugsi biasa, pada constructor dapat ditambahkan parameter ataupun dilakukan overload. Namun perlu diperhatikan bahwa nama dari fungsi constructor harus sama dengan nama kelasnya dan tidak memiliki tipe kembalian (tidak juga void).

Berikut ini merupakan contoh program yang di dalamnya terdapat pembuatan kelas dengan menggunakan sebuah constructor.

Contoh 2:

```
#include <iostream>

using namespace std;

// Membuat kelas dengan nama CONTOH
class CONTOH {
  int X;
public:
  // Membuat constructor
  CONTOH() {
     // Melakukan inisialisasi nilai X
     // dengan nilai 10
     X = 10;
  }
  // Membuat member function
  void ShowX() {
     cout<<"Nilai X: "<<X<<endl;
  }
  //...
};

int main() {

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

  // Memanggil fungsi ShowX()
  O.ShowX();

  return 0;
}
```

Hasil:

```
Nilai X: 10
```

Keterangan:

Pada program di atas, constructor dibuat dari kelas CONTOH yang namanya sama dengan kelas itu sendiri. Pada contoh ini, ketika proses instansiasi kelas CONTOH, nilai X akan diisi dengan nilai 10.

Contoh 3:

```
#include <iostream>

using namespace std;

// Membuat kelas dengan nama CONTOH
class CONTOH {
  int X;
public:
  // Membuat constructor tanpa parameter
  CONTOH() {
     // Melakukan inisialisasi nilai X
     // dengan nilai 10
     X = 10;
  }

  // Membuat constructor
  // dengan menggunakan satu parameter
  CONTOH(int XX) {
     X = XX;
  }

  // Membuat member function
  void ShowX() {
    cout<<"Nilai X : "<<X<<endl;
  }
  //...
};

int main() {

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

  // Melakukan instansiasi terhadap kelas CONTOH
  // dengan nama instance P
  CONTOH P(200);

  // Memanggil fungsi ShowX()
  O.ShowX();
  P.ShowX();

  return 0;
}
```

Hasil:

```
Nilai X : 10
Nilai X : 200
```

Contoh 3 di atas merupakan pengembangan dari program 2, dimana di dalamnya terdapat 2 buah constructor.

Cara lain yang digunakan untuk membuat constructor adalah dengan menggunakan bentuk seperti di bawah:

```
class CONTOH {
   Int X;
   Int Y;
public:
   //inisialisasi X = 10 dan Y = 0
   CONTOH () : X (10), Y (0) { }

   //nilai inisialisasi X dan Y
   //diambil dari parameter
   CONTOH (int XX, int YY) : X (XX), Y (YY)  {  }

   //deklarasi fungsi-fungsi lainnya ...
};
```

Berikut cntoh program yang menggunakan constructor dengan bentuk di atas

Contoh 4:

```
#include <iostream>

using namespace std;

// Membuat kelas dengan nama CONTOH
class CONTOH {
  int X;
public:
  // Membuat constructor tanpa parameter
  CONTOH() : X(10) {
  }

  // Membuat constructor
  // dengan menggunakan satu parameter
  CONTOH(int XX) : X(XX) {
  }

  // Membuat member function
  void ShowX() {
    cout<<"Nilai X : "<<X<<endl;
  }
  //...
};

int main() {

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

  // Melakukan instansiasi terhadap kelas CONTOH
  // dengan nama instance P
  CONTOH P(200);

  // Memanggil fungsi ShowX()
  O.ShowX();
  P.ShowX();
```

```
  return 0;
}
```

Hasil:

Destructor

Merupakan fungsi kebalikan dari constructor yang berguna untuk menghancurkan atau membuang sebuah objek (instance) dari memori. Nama fungsi destructor sama seperti nama kelas maupun nama constructor, akan tetapi di depannya ditambah tanda tilde ('~'). Destructor juga tidak memiliki tipe kembalian (tidak juga void).

Contoh program yang menggunakan fungsi destructor:

Contoh 5:

```
#include <iostream>

using namespace std;

// Membuat kelas dengan nama CONTOH
class CONTOH {
  int *X;
public:
  // Membuat constructor
  CONTOH(int XX) {
     X = new int;  // Memesan ruang memori
     *X = XX;
  }
  // Membuat destructor
  ~CONTOH() {
    delete X;  // Menghapus pointer X
  }
  void ShowX() {
    cout<<"Nilai X : "<<*X<<endl;
  }
  // ...
};

// Fungsi utama
int main() {

  CONTOH O(10);

  O.ShowX();

  return 0;
}
```

Hasil:

Tingkat Akses

Dalam C++ penentu hak akses disebut dengan access specifier, yang berarti terdapat pembatasan akses terhadap data maupun fungsi sehingga tidak semua kelas maupun lingkungan program dapat bebas mengaksesnya.

Ada 3 macam tingkat akses, yaitu:

1. **Private**

   Tingkat akses ini hanya berguna untuk memberikan hak akses data hanya pada kelas yang bersangkutan saja. Kelas-kelas turunan ataupun lingkungan luar di dalam program tidak diijinkan mengakses data tersebut. Untuk menentukan data bersifat private maka digunakan kata kunci **private**. Tetapi apabila kata kunci untuk tingkat akses tidak dituliskan dalam sebuah kelas maka secara default data atau fungsi tersebut dianggap sebagai data private.

   Contoh 6. program dengan tingkat akses private:

```
#include <iostream>

using namespace std;

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

// Fungsi utama
int main() {

  CONTOH O;
  O.SetX(100);
  O.ShowX();

  return 0;
}
```

   Hasil:

```
Nilai X: 100
```

Keterangan:

X bersifat private maka X hanya dapat diakses oleh fungsi-fungsi yang terdapat di dalam kelas CONTOH.

Contoh kode salah/ eror saat kompilasi:

```
//fungsi utama
Int main () {

	CONTOH O;
	O.X = 100; // salah karena mengakses bagian private dari kelas CONTOH

	return 0;
}
```

2. **Public**
3. **Protected**

D. LANGKAH KERJA

1. Berdoalah sebelum memulai pekerjaan.
2. Baca dan pahami labsheet yang diberikan.
3. Kerjakan contoh-contoh hasil program di atas .
4. Apabila mengalami kesulitan atau pekerjaan telah selesai, konsultasikan pada dosen pengampu.
5. unggah hasil pekerjaan anda di bestmart pada mata kuliah algoritma dan struktur data.

E. TUGAS

1. Pelajari dan pahami kode program di atas.
2. Buatlah algoritma dengan bahasa narasi, flowchart, pseudocode, dan program di atas.
3. Buatlah 3 algoritma dan program dengan menggunakan OOP.

F. DAFTAR PUSTAKA

Kajian pustaka diambil dari buku:

Budi Raharjo. 2015. Pemrograman C++ Mudah dan Cepat Menjadi Master C++. Bandung: Penerbit Informatika.