MODUL
MATA KULIAH
PEMROGRAMAN KOMPUTER

JURUSAN PENDIDIKAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
2006

MODUL KULIAH
PEMROGRAMAN KOMPUTER

DAFTAR ISI

BAB 1. PENDAHULUAN BAHASA C++
1.1 Struktur Bahasa C++
1.2 Obyek Dasar Dalam C++
1.3 Memberi Nilai Ke Variabel (Penugasan)
1.4 Operator Aritmatika
1.5 Operator Relasional
1.4 Strlen
Latihan Soal

BAB 2. STRUKTUR KENDALI PROGRAM (*Control Structure*)
2.1 Seleksi Kondisional Memakai If
2.2 Seleksi Kondisional Memakai if . . else
2.3 If Bersarang (Nested If)
2.4 Struktur Selektif Menggunakan Switch
Latihan Soal

BAB 3. PERULANGAN/ITERASI
3.1 Loop While
3.2 Loop Do . . while
3.3 Loop For
3.4 Statement Jump
    3.4.1 Statement : Break
    3.4.2. Statement : Continue
    3.4.3. Statement : Goto
Latihan Soal

BAB 4. FUNGSI
4.1 Manfaat Penggunaan Fungsi
4.2 Format Penggunaan Fungsi
4.3 Pelewatan Parameter
4.4. Variabel Lokal Dan Variabel Global
4.5 Fungsi Yang Tidak Mengembalikan Nilai
4.6 Arguments Dilewatkan Melalui Nilai dan Referensi.
4.7 Fungsi Overload
4.8 Rekursif
Latihan Soal

BAB 5. ARRAY
5.1 Definisi Array
5.2 Inisialisasi Nilai Array
5.3 Mengakses Elemen Array
5.4 Array Multidimensi
5.5 Array Tipe Char
5.6 Inisialisasi Array Tipe Char
Latihan Soal

BAB 6. PEMROGRAMAN PORT PARALEL
6.1 Port Serial Dan Paralel
6.2 Pengalamatan Port Paralel
Latihan Soal

BAB 7. OBJECT-ORIENTED PROGRAMMING (OOP)
7.1 Memandang Sesuatu Sebagai Objek
7.2 Dunia Tersusun Dari Objek-Objek
7.3 Objek Abstrak Dan Instant
7.4 Inheritance (Pewarisan)
7.5 Objek Di Dunia Nyata Serta Pemrograman Berorientasi Objek
7.6 Bekerja Dengan Class
    7.6.1 Definisi Class
    7.6.2 Variabel Dan Class
7.7. Method
Latihan Soal

BAB 8. OOP 2
8.1 Pemakaian Struktur
8.2 Pemakaian Konsep OOP dalam program
Latihan Soal

BAB 1
PENDAHULUAN BAHASA C++

Bahasa C++ merupakan perkembangan dari bahasa pendahulunya, yaitu bahasa C. Bahasa C diciptakan oleh Brian Kernighan dan Dennis Ritchie, sedangkan C++ diciptakan oleh Bjarne Stroustrup. Bahasa C ini banyak digunakan untuk membangun perangkat lunak seperti Microsoft Windows, Microsoft Office, dsb. Linux yang merupakan sistem operasii bersifat *open source* juga dikembangkan oleh programmer di seluruh dunia menggunakan C.

Bahasa C++ memiliki kelebihan dibandingkan dengan Bahasa C, karena C++ memiliki kemampuan dalam hal OOP (*Object Oriented Programming*/Pemrograman Berorienrtasi Obyek) yang merupakan trend masa kini dalam bidang pemrograman. Dibandingkan dengan Java yang hanya berorientasi pada OOP, C++ mengadopsi kemampuan bahasa C (pemrograman terstruktur) plus kemampuan OOP. Jadi, C++ tidak murni OOP.

1.1 STRUKTUR BAHASA C++

Sebelum membahasa mengenai OOP, terlebih dahulu diperkenalkan struktur umum dan perintah-perintah dasar yang digunakan pada C++. Apabila anda telah menguasai bahasa C, maka sangat mudah bagi anda untuk menguasai C++. Untuk menguasai bahasa C++, terlebih dahulu harus kita pahami struktur umum kode program C++. Untuk memahami struktur umum bahasa C++, kita coba dengan membuat sebuah program sederhana sebagai berikut.

```
// program pertama dalam C++
//dibuat tanggal 25 Maret 2006

#include <iostream>
int main ()
{
  cout << "Hello World!"<<endl;
  return 0;
}
```

Program di atas akan menghasilkan keluaran : "Hello World!" pada layar. Umumnya, para programmer yang ingin menguasai C++ memulai membuat program dengan program sederhana seperti di atas, mengingat dalam program tersebut telah mengandung komponen dasar dari sebuah program C++.

> ➢ **Catatan :**
> Keluaran program yang dihasilkan setelah anda meng*compile* program di atas umumnya tidak terlihat karena layar akan segera menutup setelah menampilkan tulisan "Hello World!". Agar anda dapat melihat keluaran, tambahkan baris : `getch();` pada akhir program.

Penjelasan program di atas adalah :

1. `// my first program in C++`

Merupakan baris komentar yang digunakan untuk memberi penjelasan kode program yang dibuat (seperti : tanggal pembuatan program, pencipta program, serta

tujuan dibuatnya program). Baris komentar selalu diawali dengan tanda (//) . baris komentar tidak akan dieksekusi oleh *compiler*/diabaikan.

2. `#include <iostream>`

Merupakan pengarah preprocessor. `#include <iostream>` berarti bahwa memberi perintah kepada preprocessor untuk melibatkan (*include*) file standar iostream. File iostream ini merupakan file yang didalamnya berisi deklarasi pustaka standar input-output pada C++, dan oleh karenanya dilibatkan karena fungsi ini selanjutnya akan dipergunakan oleh program.

3. `int main ()`

Merupakan awal dari fungsi utama pada C++. Fungsi utama ini merupakan awal dimana program C++ akan mulai dieksekusi. Di dalam fungsi utama ini, bisa jadi terdapat juga fungsi-fungsi lain (seperti yang telah anda ketahui, bahwa sebuah program yang sangat besar akan lebih mudah untuk ditulis dengan membaginya/dipecah menjadi beberapa fungsi). Ibarat anda menulis sebuah buku, anda membagi isi buku menjadi beberapa bab, demikian juga dengan kode program.

4. Kurung kurawal buka ({) dan tutup (})

Blok program pada fungsi utama diawali dengan kurung kurawal buka ({) dan berakhir hingga ditemukan kurung kurawal tutup (}). Semua statement yang berada pada blok program fungsi utama inilah yang akan dieksekusi oleh *compiler*.

5. `Cout<< "Hello World"`

Merupakan statement pada C++. statement merupakan ekspresi (sederhana atau kompleks) yang akan menghasilkan efek tertentu saat dieksekusi. Statement `cout` (baca : C-out) akan menampilkan keluaran pada layar. `cout` dideklarasikan pada file `iostream`. Oleh karena itulah file `iostream` dilibatkan melalui perintah `#include <iostream>`.

Tanda << dalam C++ berarti meyisipkan **string** "Hello World"ke dalam aliran keluaran.

6. `Endl`

Merupakan perintah untuk berganti baris setelah menuliskan "Hello World". Biasakan anda selalu menggunakan perintah ganti baris, yang menandakan bahwa anda siap memberikan instruksi pada compiler untuk mengeksekusi baris program selanjutnya.

7. `Return 0;`

statement `return` menyatakan akhir dari fungsi utama. `return` umumnya diikuti dengan kode tertentu (dalam contoh ini, kode return adalah 0). `return 0` menyatakan bahwa program bekerja sebagaimana yang diinginkan tanpa adanya error selama eksekusi. Kode ini juga melakukan terminalisasi (menutup berkas-berkas yang digunakan) serta mengembalikan kendali program ke sistem operasi. `Return 0` umumnya digunakan untuk mengakhiri program C++.

Anda bisa juga menuliskan kode program di atas sebagai berikut :

```
int main () { cout << "Hello World"<< endl;return 0; }
```

Namun penulisan dengan cara kedua tersebut sangat tidak dianjurkan, karena kode program menjadi sulit dibaca, dan sulit dipahami. Usahakan untuk selalu berpindah ke baris baru setelah anda mendeklarasikan fungsi utama, dan setelah ada tanda titik koma (;).

LATIHAN I.

1. Tuliskan kalimat berikut pada layar komputer :

```
Nama : (isi nama anda)
NIM  : (isi NIM anda)
Jurusan : (isi sesuai jurusan anda)
```

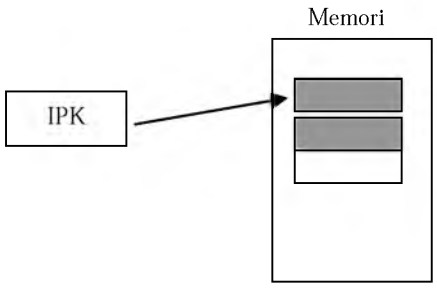
Program yang telah anda buat hanya melibatkan perintah keluaran. Tentu saja kita tidak mempelajari bahasa pemrograman hanya untuk menampilkan keluaran teks seperti program di atas. Program yang baik umumnya melibatkan interaksi dengan *user*. Jadi, ada masukan yang dberikan oleh *user*. Program di bawah ini melibatkan perintah masukan, yaitu `cin`.

```
// program kedua dalam C++
//dibuat tanggal 25 Maret 2006

 #include <iostream>
int main ()
{
  float ipk;
  cout <<"masukkan ipk anda = "<<flush;
  cin >> ipk;
  cout <<"nilai IPK anda adalah = "<<ipk <<"(apik tenan)";
  return 0;
}
```

Catatan :

- `Flush` merupakan perintah yang mirip dengan `endl`, namun tidak mengakibatkan ganti baris baru. Program di atas akan meminta user memasukkan nilai IPK. Saat user mengetik nilai IPK, maka nilai yang diketik tersebut (merupakan masukan) akan disimpan pada *buffer* (memori) komputer hingga penuh. Dalam hal ini, `flush` digunakan untuk memerintahkan komputer mengosongkan isi buffer dan menuliskannya ke layar monitor.
- Digunakan `cin >>` sebagai perintah meminta masukan dari user, perhatikan bahwa `cin` diikuti dengan tanda >>, sedangkan `cout` selalu diikuti dengan <<. **Jangan sampai terbalik !!**
- `Float IPK`
  Merupakan pernyataan deklarasi variabel. Saat anda mengetik nilai IPK, nilai tersebut disimpan pada memori komputer, dengan nama IPK. Ilustrasinya sebagai berikut.



Gambar 1.1 Penyimpanan variabel ke dalam memori

BAB 8
OOP II

8.1 STRUKTUR

Untuk dapat memahami pemrograman OOP pada C++, diharapkan anda sudah memahami konsep penggunaan tipe data struktur. Misalkan sebuah tipe struktur dalam program C++ sebagai berikut:

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

struct atribut
{  char nama[20];
     char alamat[30];
   float ipk;
} ;
int main ()
{
atribut mhs;
cout<<"nama anda = ";gets(mhs.nama);
cout<<"alamat = ";gets(mhs.alamat);
cout<<"IPK = ";
cin>>(mhs.ipk);
clrscr();
cout<<" SELAMAT DATANG "<<(mhs.nama)<<endl;
cout<<" IPK ANDA = "<<(mhs.ipk);
cout<<" alamat anda di : "<<(mhs.alamat);
getch();
}
```

1. Bagaimanakah hasil tampilan program ?
2. Apa beda fungsi `gets` dan `cin` ?

OOP melakukan pemrograman menggunakan class. Apabila program di atas dijadikan OOP, maka deklarasi tipe data struktur diubah sebagai berikut :

```
class atribut
{  public:
     char nama[20];
     char alamat[30];
   float ipk;
} ;
```

Kata public menyatakan bahwa anggota kelas dapat diakses secara umum. Apabila anda menginginkan anggota class tidak dapat diakses di luar class secara langsung, maka anda jadikan sebagai **private**. Dalam OOP, setiap objek dalam anggota kelas dapat memiliki method (behavior) tertentu. Untuk objek class atrribut di atas, terdapat beberapa method (behavior) yang dapat kita lakukan, antara lain :

1. Masukkan data mhs.
2. Tampilkan data mhs.

Maka program OOP selengkapnya adalah sebagai berikut :

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

class atribut
{  private:
      char nama[20];
      char alamat[30];
        float ipk;

   public:
   void isi_data()
   {
   cout<<"nama = "; gets(nama);
   cout<<"alamat = "; gets(alamat);
   cout<<"ipk = "; cin>>ipk;
   }

   void tampil()
   {
   cout<<"nama anda adalah = "<<nama<<endl;
   cout<<" beralamat di = "<<alamat<<endl;
   cout<<"IPK = anda ternyata "<<ipk<<endl;
   }
};

int main ()
{
clrscr();
atribut mhs;
mhs.isi_data();
clrscr();
mhs.tampil();
getch();
}
```

Analisis :

1. Class atribut memiliki 3 anggota yang bersifat private , yaitu : `nama`, `alamat` dan `ipk`. Class ini memiliki 2 method/fungsi anggota yang bersifat public, yaitu : `isi_data()` dan `tampil()`.
2. Objek bernama mhs memiliki class atribut, ditunjukkan pada deklarasi : `atribut mhs`
3. Selanjutnya, fungsi anggota dapat dipanggil di dalam fungsi `main()`

Pada program di atas, kita memiliki 1 objek bernama mhs dengan class atribut. Kita dapat membentuk beberapa objek dengan class yang sama (fungsi inheritance?). tentu saja data anggota masing-masing objek akan berbeda. Misal, kita buat objek menjadi 2, yaitu : mhs_S1 dan mhsD3.

Dalam hal ini, deklarasi anggota class dan method/behavior/fungsi anggotanya tetap. Potongan fungsi di dalam `main` sebagai berikut.

```
int main ()
{
clrscr();
atribut mhs_S1, mhsD3;
mhs_S1.isi_data();
mhsD3.isi_data();
clrscr();
mhs_S1.tampil();
mhsD3.tampil();
getch();
}
```

Pertanyaan :

1. Bagaimana keluaran program ?
2. Apa yang terjadi jika di dalam main terdapat kode berikut :

```
int main ()
{
clrscr();
atribut mhs_S1, mhsD3;
mhs_S1.isi_data();
mhsD3=mhs_S1;   //apa maksudnya ?
clrscr();
mhsD3.tampil();
getch();
}
```

Pada OOP, terdapat 2 cara mendefinisikan method/behavior/fungsi anggota. Cara pertama adalah seperti yang sudah kita gunakan, dimana fungsi anggota class kita definiskan secara langsung di dalam class. Keuntungan bentuk semacam ini adalah : semua kode terkumpul dalam class, sehingga mempercepat pengeksekusian, namun memiliki kelemahan karena kode program yang dibangkitkan menjadi besar, terutama apabila banyak fungsi anggota yang didefinisikan dan banyak objek yang terlibat. Bentuk yang kedua, kita definisikan fungsi class di luar class seperti di bawah ini. Umumnya, bentuk ke dua ini paling banyak digunakan, terutama apabila fungsi class sangat panjang dan kompleks. Dengan cara pertama ataupun kedua, akan didapatkan keluaran yang sama.

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

class atribut
{  private:
     char nama[20];
     char alamat[30];
   float ipk;

   public:
```

```
    void isi_data();
    void tampil();
};

int main ()
{
clrscr();
atribut mhs;
mhs.isi_data();
clrscr();
mhs.tampil();
getch();
}

void atribut::isi_data()
{
      cout<<"nama  = "; gets(nama);
    cout<<"alamat  = "; gets(alamat);
    cout<<"ipk = "; cin>>ipk;
}
void atribut::tampil()
{
      cout<<"nama anda adalah= "<<nama<<endl;
    cout<<"anda beralamat di= "<<alamat<<endl;
    cout<<"IPK anda ternyata= "<<ipk<<endl;
}
```

Catatan : kata void pada fungsi anggota berarti tidak ada nilai baliknya (void). Pelajari kembali : **function.**

BAB 8
OOP II

8.1 STRUKTUR

Untuk dapat memahami pemrograman OOP pada C++, diharapkan anda sudah memahami konsep penggunaan tipe data struktur. Misalkan sebuah tipe struktur dalam program C++ sebagai berikut:

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

struct atribut
{  char nama[20];
     char alamat[30];
   float ipk;
} ;
int main ()
{
atribut mhs;
cout<<"nama anda = ";gets(mhs.nama);
cout<<"alamat = ";gets(mhs.alamat);
cout<<"IPK = ";
cin>>(mhs.ipk);
clrscr();
cout<<" SELAMAT DATANG "<<(mhs.nama)<<endl;
cout<<" IPK ANDA = "<<(mhs.ipk);
cout<<" alamat anda di : "<<(mhs.alamat);
getch();
}
```

1. Bagaimanakah hasil tampilan program ?
2. Apa beda fungsi `gets` dan `cin` ?

OOP melakukan pemrograman menggunakan class. Apabila program di atas dijadikan OOP, maka deklarasi tipe data struktur diubah sebagai berikut :

```
class atribut
{  public:
     char nama[20];
     char alamat[30];
   float ipk;
} ;
```

Kata public menyatakan bahwa anggota kelas dapat diakses secara umum. Apabila anda menginginkan anggota class tidak dapat diakses di luar class secara langsung, maka anda jadikan sebagai **private**. Dalam OOP, setiap objek dalam anggota kelas dapat memiliki method (behavior) tertentu. Untuk objek class atrribut di atas, terdapat beberapa method (behavior) yang dapat kita lakukan, antara lain :

1. Masukkan data mhs.
2. Tampilkan data mhs.

Maka program OOP selengkapnya adalah sebagai berikut :

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

class atribut
{  private:
      char nama[20];
      char alamat[30];
        float ipk;

   public:
   void isi_data()
   {
   cout<<"nama = "; gets(nama);
   cout<<"alamat = "; gets(alamat);
   cout<<"ipk = "; cin>>ipk;
   }

   void tampil()
   {
   cout<<"nama anda adalah = "<<nama<<endl;
   cout<<" beralamat di = "<<alamat<<endl;
   cout<<"IPK = anda ternyata "<<ipk<<endl;
   }
};

int main ()
{
clrscr();
atribut mhs;
mhs.isi_data();
clrscr();
mhs.tampil();
getch();
}
```

Analisis :

1. Class atribut memiliki 3 anggota yang bersifat private , yaitu : `nama`, `alamat` dan `ipk`. Class ini memiliki 2 method/fungsi anggota yang bersifat public, yaitu : `isi_data()` dan `tampil()`.
2. Objek bernama mhs memiliki class atribut, ditunjukkan pada deklarasi : `atribut mhs`
3. Selanjutnya, fungsi anggota dapat dipanggil di dalam fungsi `main()`

Pada program di atas, kita memiliki 1 objek bernama mhs dengan class atribut. Kita dapat membentuk beberapa objek dengan class yang sama (fungsi inheritance?). tentu saja data anggota masing-masing objek akan berbeda. Misal, kita buat objek menjadi 2, yaitu : mhs_S1 dan mhsD3.

Dalam hal ini, deklarasi anggota class dan method/behavior/fungsi anggotanya tetap. Potongan fungsi di dalam `main` sebagai berikut.

```
int main ()
{
clrscr();
atribut mhs_S1, mhsD3;
mhs_S1.isi_data();
mhsD3.isi_data();
clrscr();
mhs_S1.tampil();
mhsD3.tampil();
getch();
}
```

Pertanyaan :

1. Bagaimana keluaran program ?
2. Apa yang terjadi jika di dalam main terdapat kode berikut :

```
int main ()
{
clrscr();
atribut mhs_S1, mhsD3;
mhs_S1.isi_data();
mhsD3=mhs_S1;   //apa maksudnya ?
clrscr();
mhsD3.tampil();
getch();
}
```

Pada OOP, terdapat 2 cara mendefinisikan method/behavior/fungsi anggota. Cara pertama adalah seperti yang sudah kita gunakan, dimana fungsi anggota class kita definiskan secara langsung di dalam class. Keuntungan bentuk semacam ini adalah : semua kode terkumpul dalam class, sehingga mempercepat pengeksekusian, namun memiliki kelemahan karena kode program yang dibangkitkan menjadi besar, terutama apabila banyak fungsi anggota yang didefinisikan dan banyak objek yang terlibat. Bentuk yang kedua, kita definisikan fungsi class di luar class seperti di bawah ini. Umumnya, bentuk ke dua ini paling banyak digunakan, terutama apabila fungsi class sangat panjang dan kompleks. Dengan cara pertama ataupun kedua, akan didapatkan keluaran yang sama.

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>

class atribut
{  private:
     char nama[20];
     char alamat[30];
   float ipk;

   public:
```

```
    void isi_data();
    void tampil();
};

int main ()
{
clrscr();
atribut mhs;
mhs.isi_data();
clrscr();
mhs.tampil();
getch();
}

void atribut::isi_data()
{
      cout<<"nama  = "; gets(nama);
    cout<<"alamat  = "; gets(alamat);
    cout<<"ipk = "; cin>>ipk;
}
void atribut::tampil()
{
      cout<<"nama anda adalah= "<<nama<<endl;
    cout<<"anda beralamat di= "<<alamat<<endl;
    cout<<"IPK anda ternyata= "<<ipk<<endl;
}
```

Catatan : kata void pada fungsi anggota berarti tidak ada nilai baliknya (void). Pelajari kembali : **function.**