Pertemuan : 9 (Sembilan)

Pokok Bahasan : FUNGSI

Tujuan Khusus : Mahasiswa mampu menyusun algoritma asecara terstruktur dengan menggunakan pemanggilan fungsi

---

Pendahuluan

Program komputer yang dibuat untuk menjawab permasalahan umumnya berukuran besar. Seringkali dalam membuat program besar tersebut adalah memecah program menjadi potonga-potongan program yang kecil yang dinamakan modul. Teknik pemrograman seperti ini dinamakan teknik pemrograman modular. Beberapa bahasa pemrogramn menamakan modul dengan sebutan rutin(routine), prosedure atau fingsi.

Modularisasi program mempunyai dua keuntungan:

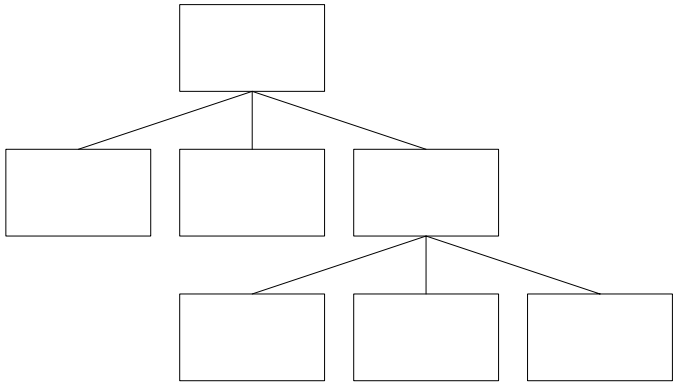
1. Untuk aktifitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama berulang kali. Modul tersebut cukup dituliskan sekali saja, lalu modul tersebut dapat diakses(diistilahkan dengan dipanggil dari bagian lain di dalam program). Hal ini bermanfaat bila ingin menghemat ukuran program.
2. Kemudahan menulis dan menemukan kesalahan (debug) program. Hal ini sangat berguna pada masalah yang besar.

Modul program di dalam C++ disebut fungsi (function)

Dasar Fungsi

Pada umumnya fungsi memerlukan masukan yang dinamakan argumen atau parameter. Hasil akhir fungsi akan berupa sebuah nilai(nilai balik fungsi).

Konsep Program dibagi menjadi sejumlah modul



Bentuk umum fungsi:

Penentu-tipe nama fungsi (daftar parameter)

Dekalarasi parameter

{

isi fungsi

}

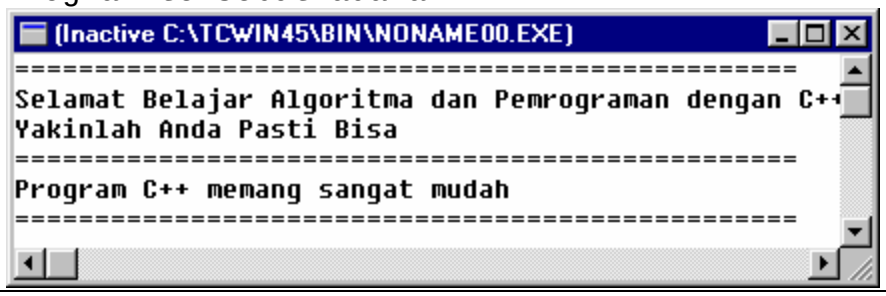
Penentu tipe fungsi untuk menentukan tipe keluaran fungsi yang dapat berupa salah satu tipe data C++ yang berlaku, misal char atau int

Sebuah contoh program yang menggunakan fungsi:

Kasus 9.1

```
#include <iostream.h>
#include <conio.h>
void garis();  /*Prototipe fungsi garis()*/
void main()
{
clrscr(); /*Hapus layar*/
garis();       /*panggil fungsi garis()*/
cout<< "Selamat Belajar Algoritma dan Pemrograman dengan
C++ \n";
cout<< "Yakinlah Anda Pasti Bisa \n";
garis();
cout<< "Program C++ memang sangat mudah \n";
garis();
}

void garis()   /*Definisi fungsi garis()*/
{
int i;
for (i=0;i<49;i++)
cout<<"=";
cout<<endl;
```

```
}

Hasil Program tersebut adalah:
```



Prototipe Fungsi

Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan, deklarasi fungsi dikenal dengna sebutan prototipe fungsi. Prototipe fungsi berupa:

- Nama fungsi
- Tipe nilai balik fungsi
- Jumlah dan tipe agumen

Dan diakhiri dengan titik koma, sebagaimanapada pendeklarasian variabel. Sebagai contoh:

a. long kuadatrat(long l);

   Pada contoh pertama, fungsi kuadrat() mempunyai argumen bertipe long dan nilai balik bertipe long.

b. int maks(int a, int b, int c);

   Pada contoh kedua, fungsi maks() memiliki tiga buah argumen, masing-masing bertipe int dan nilai balik juga bertipe int.

c. double maks(double x, double y);

   Pada contoh ketiga, fungsi maks() mempunyai dua buah argumen, dengan masing-masing argumen bertipe double.

d. void garis();

   Pada contoh keempat, fungsi garis() tidak memiliki argumen dan nilai baliknya tidak ada (void)

Manfaat dari prototipe fungsi adalah untuk menjamin tipe argumen yang dilewatkan paada pemanggilan fungsi benar-benar sesuai. Tanpa adanya prototipe

fungsi, amatlah mudah bagi pemrogram untuk melakukan kesalahan tanpa sengaja dalam melewatkan argumen.

Pada prototipe fungsi, nama argumen boleh ditiadakan.
Sebagai contoh:

```
Long kuadrat (long);
```

Merupakan alternatif dari

```
Long kuadrat (long l);
```

Namun keberadaan nama argumen terkadang membantu bagi pembaca, dalam hal lebih memberikan kejelasan.

Definisi fungsi

Setiap fungsi yang dipanggil di dalam program harus didefinisikan. Letaknya bisa dimana saja. Khusus untuk fungsii yang disediakan di sistem, definisinya sebenarnya ada dalam pustaka, yang akan digaungkan dengan program sewaktu proses linking.

Kasus 9.2

Contoh lain misalkan Anda diminta membuat algoritma dan program untuk menentukan bilangan terbesar dari 2 buah bilangan yang diketahui dengan menggunakan fungsi. Anda terlebih dahuu harus mendeklarasikan fungsi untuk menampung dan membandingkan 2 buah bilangan tersebut.

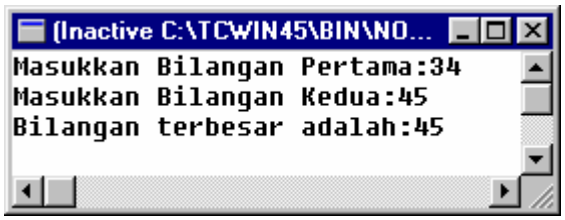
Algoritma untuk permasalahan di atas:

1. Deklarasikan fungsi untuk menampung 2 buah bilangan(bil pertama dan bil kedua)
2. Deklarasikan fungsi untuk mencari bilangan terbesar
3. Tentukan dua buah bilangan tersebut
4. Tentukan variabel untuk menampung fungsi dari bilangan terbesar
5. Sesuai dengan langkah satu, lakukan pengujian untuk 2 buah bilangna tersebut
6. Jika bilangan pertama lebih besar dari bilangan kedua, dan jawabannya ya maka bilangan pertama yang menjadi ilangan terbesar
7. Dari langkah 6 bila jawabannya tidak, maka bilangan kedua yang menjadi bilangan terbesar.

8. Sesuai langkah 2, Cetak bilangan terbesar.

Program untuk kasus diatas adalah:

```
#include <iostream.h>
#include <conio.h>
int   findmax(int   n1,   int   n2);     /*Prototipe   fungsi
garis()*/
void printmax(int m);

void main()
{
int x, y, k;
clrscr(); /*Hapus layar*/
cout<<"Masukkan Bilangan Pertama:";
cin>>x;
cout<<"Masukkan Bilangan Kedua:";
cin>>y;
k = findmax(x,y);
printmax(k);
}

int findmax(int n1, int n2) //definisi fungsi
{
if (n1>n2)
{
return (n1);
}
else
{
return (n2);
}
}

void printmax(int m)
{
cout<<"Bilangan terbesar adalah:"<<m;
}

Hasil program diatas (misalkan bil 1 dimasukkan angka 34
dan bil 2 dimasukkan angka 45)
```



Fungsi Tanpa Nilai Balik

Adakalanya suatu fungsi tidak perlu memiliki nilai balik. Misal fungsi yang hanya dimasukkan untuk menampilkan suatu keterangan saja. Pada fungsi seperti ini, tipe nilai balik fungsi yang diperlukan adalah void.

Contoh

```
Void tampilkan_judul()
{
cout<<”Universitas INDONUSA Esa Ungul”<<endl;
cout<<”Jalan Terusan Arjuna Tol Tomang”<<endl;
cout<<”Jakarta Barat”<<endl;
}
```

Pada contoh di atas, tidak ada pernyataan return, mengingat fungsi tidak memiliki nilai balik. Namun penggunaan pernyataan return secara eksplisit juga bisa diperkenankan.

```
Void tampilkan_judul()
{
cout<<”Universitas INDONUSA Esa Ungul”<<endl;
cout<<”Jalan Terusan Arjuna Tol Tomang”<<endl;
cout<<”Jakarta Barat”<<endl;
return;
}
```

Lingkup Variabel

Lingkup variabel menentukan keberadaan suatu variabel tertentu dalam suatu fungsi. Jenis variabel berdasarkan kelas penyimpanan:

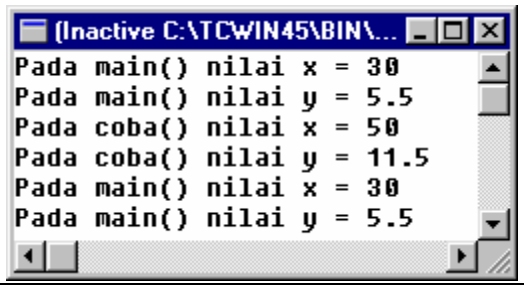
a. variabel otomatis
b. variabel eksernal
c. variabel statis

Variabel Otomatis

Variabel yang didefinisikan di dalam suatu fungsi berlaku sebagai variabel lokal bagi fungsi. Artinya variabel tersebut hanya dikenal di dalam fungsi tempat variabel didefinisikan.

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

```
#include <conio.h>

void coba(); //Prototipe Fungsi

void main()
{
     int x= 30;    //variabel lokal pada main()
     double y= 5.5;//variabel lokal pada main()
     clrscr();  //Hapus Layar
     cout<<"Pada main() nilai x = " <<x<<"\n";
     cout<<"Pada main() nilai y = " <<y <<"\n";
     coba();
     cout<<"Pada main() nilai x = " <<x<<"\n";
     cout<<"Pada main() nilai y = " <<y<<"\n";
}
//Definisi fungsi coba()
void coba()
{
     int x = 50;
     double y = 11.5;
     cout<<"Pada coba() nilai x = " <<x<<"\n";
     cout<<"Pada coba() nilai y = " <<y<<"\n";
}

Hasil Program adalah:
```



Tampak bahwa perubahan x dan y pada coba() tidak mempengaruhi variabel dengan nama yang sama pada main(). Ini membuktikan bahwa variabel-variabel tersebut bersifat lokal bagi masing-masing fungsi yang mendefinisikannya.

Suatu variabel otomatis mempunyai sifat:

- variabel hanya akan diciptakan pada saat fungsi dipanggil
- Pada saat fungsi berakhir (selesai dieksekusi), variabel otomatis akan sirna
- Tidak ada inisialisasi secara otomatis (saat variabel diciptakan). Inisialisasi program akan dikerjakan setiap kali fungsi dipanggil.
- Hanya dapat diakses di dalam fungsi yang mendefinisikannya

Variabel Eksternal

Variabel eksternal merupakan kebalikan variabel otomatis. Variabel eksternal adalah ariabel yang didefinisikan di luar fungsi manapun. Variabel ini dikenal juga sebagai variabel global sebab variabel ini dikenal di semua fungsi.

Kasus 9.3 Contoh penggunaan variabel eksternal

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

int a = 10;   //variabel eksternal
void kali(); //Prototipe Fungsi

void main()
{
     clrscr();
     cout<<a<<endl;
     kali();
     cout<<a<<endl;
     kali();
     cout<<a<<endl;
}
//Definisi Fungsi
void kali()
{
     a = a*2;
}
```

Hasil program di atas adalah:

Tampak bahwa meskipun di dalam fungsi main() dan kali() tidak mendefinisikan variabel a, ternyata variabel ini dikenal di dalam kedua fungsi tersebut. Nilai dari variabel a dapat diubah dari dalam fungsi kali().

Penggunaan variabel eksternal diusahakan sesedikit mungkin atau sedapat mungkin tidak usah digunakan. Tidka lain adalah karena ini mudah sekali berubah oleh pernyataanpenugasan yang letaknya bisa di mana saja. Ini bisa menimbulkan efek samping yang suli untuk melacaknya, terutama untuk program besar.

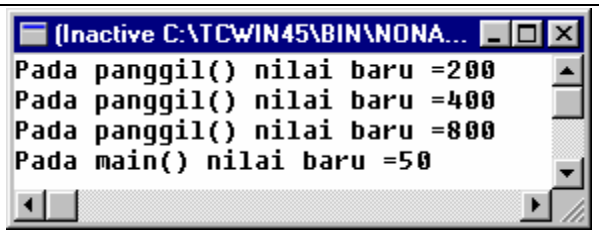
Variabel Statis

Baik variabel eksternal maupun otomatis dapat berkedudukan sebagai variabel statis. Suatu variabel statis mempunyai sifat:

1. Jika variabel lokal berdiri sebagai variabel statis, maka:
    - variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
    - Variabel tidak hilang saat diekskusi fungsi berakhir. Nilainya akan tetap dipertahankan, sehingga akan dikenali pada pemanggilan fungsi untuk tahap berikutnya.
    - Inisialisasi oleh pemrogram akan dilakukan sekali saja selama program dijalankan. Jika tidak ada inisialisasi secara eksplisit, variabel diisi dengan nol.
2. Jika variabel eksternaldijadikan sebagai variabel statis variabel ini dapat diakses oleh semua file yang didefinisikan pada file yang sama dengan variabel eksternal tersebut.

Kasus 9.4 Contoh penggunaan variabel static

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

void panggil(); //Prototipe Fungsi

void main()
{
     int baru = 50;
     clrscr();
     panggil();
     panggil();
     panggil();
     cout<<"Pada main() nilai baru ="<<baru<<endl;
}
//Pada fungsi berikut variabel baru didefinisikan sebagai
variabel statis

//Definisi Fungsi
void panggil()
{
     static int baru= 100;
     baru = baru *2;
     cout<<"Pada panggil() nilai baru ="<<baru<<endl;
}

Hasil untuk program di atas adalah:
```



Berdasarkan hasil di atas, terlihat bahwa variabel statis baru pada fungsi panggil hanya diinisialisasikan sekali saja kemudian seap fungsi panggil() diakses nilai variabel tersebut diproses.

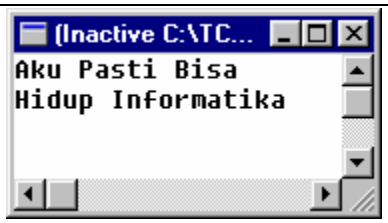
Tampak pula bahwa variabel bernama sama yang didefinisikan di fungsi main() tidak ada kaitannya dengan variabel yang ada di fungsi panggil().

Nilai Bawaan untuk Argumen Fungsi

Salah satu keistimewaan C++ yang sangat bermanfaat dalam pemrogramn adalah adanya kemampuan untk menyetel nilai bawaan (default) argumen fungsi. Argumen-argumen yang mempunyai nilai bawaan nantinya dapat tidak disertakan di dalam pemanggilan fungsi dan dengan sendirinya C++ akan menggunakan nilai bawaan dari argumen yang tidak disertakan.

Kasus 9.5 Lihat contoh berikut ini:

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

void cetak(int jum); //Prototipe Fungsi

void main()
{
     clrscr();
     cetak(1);

}
//Definisi Fungsi
void cetak(int jum)
{
     for (int i=0;i<jum;i++)
          cout<<"Aku Pasti Bisa"<<endl;

     cout<<"Hidup Informatika"<<endl;
}

Hasil program di atas adalah
```



Kasus 9.5 Sebuah contoh lain mengenai pemberian bilai awal terhadap argumen fungsi:

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

void cetak(char x='*', int jum= 10); //Prototipe Fungsi

void main()
{
     clrscr();
     cetak('A', 5); //Tampilkan A sebanyak 5 kali
     cetak('B');   //Tampilkan B sebanyak 10 kali
     cetak();        //Tampilan * sebanyak 10 kali

}
//Definisi Fungsi
void cetak(char x, int jum)
{
     for (int i=0;i<jum;i++)
          cout<<x;

     cout<<endl;
}
```

Hasil Program di atas adalah:

Referensi

Pada C++ referensi digunakan untuk memberikan nama alias dari variabel. Bentuk peneklarasiannya:

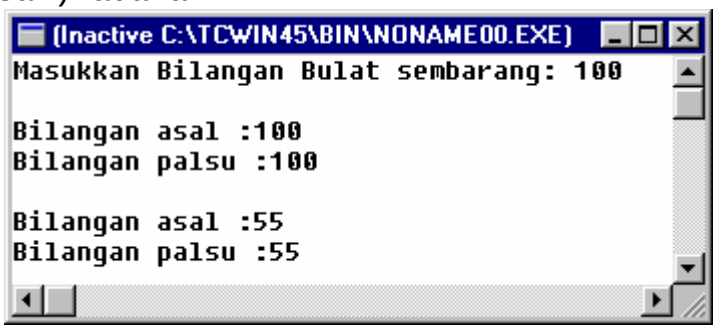
Int & ref = nama-variabel;

Tanda & mengawali nama referensi.

Setelah pendeklarasian seperti di atas, ref menjadi nama alias dari nama_variabel. Pengubahan nilai terhadap nama-variabel dapat dilakukan melalui nama_variabel itu sendiri ataupun nelalui referensi ref, sebagaimana contoh berikut ini:

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

void main()
{
     int asal;
     int &palsu = asal; //Deklarasi referensi
     clrscr();

     cout<<"Masukkan Bilangan Bulat sembarang: ";
     cin>>asal;
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;

     palsu = 55;
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;

}
```

Hasil Program tersebut (Bila dimasukkan angka 100 pada variabel asal) adalah:

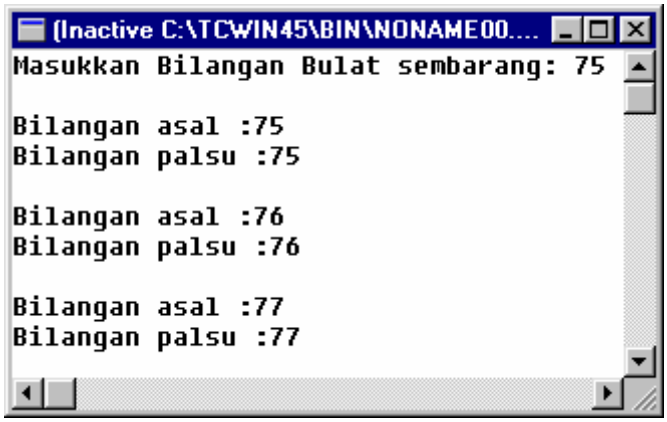


Kasus 9.6 Contoh program yang menggunakan operator:

```
#include <iostream.h>
#include <conio.h>
void main()
{

     int asal;
     int &palsu = asal; //Deklarasi referensi
     clrscr();

     cout<<"Masukkan Bilangan Bulat sembarang: ";
     cin>>asal;
```

```
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;
     asal++;
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;
     palsu++;
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;

}
```

Hasil dari program di atas bila dimasukkan angka 75 adalah;



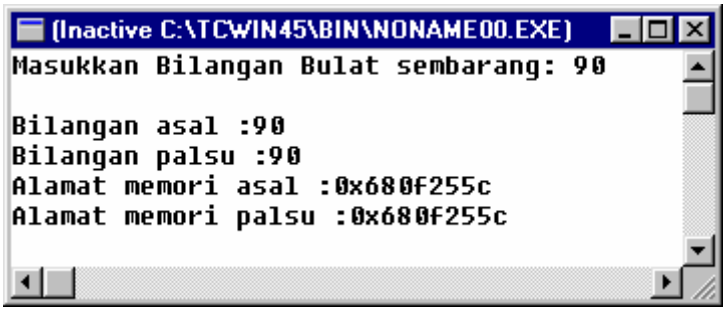
Tampak penggunaan operator ++ pada asal dan palsu akan mengubah nilai keduanya, hal ini terjadi karena variabel asal dan palsu menunjuk pada alamat memori yang sama.

Kasus 9.7 Berikut ini program untuk menampilkan alamat memori suatu variabel:

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

void main()
{

     int asal;
     int &palsu = asal; //Deklarasi referensi
     clrscr();

     cout<<"Masukkan Bilangan Bulat sembarang: ";
     cin>>asal;
     cout<<"\nBilangan asal :"<<asal<<endl;
     cout<<"Bilangan palsu :"<<palsu<<endl;
     cout<<"Alamat memori asal :"<<&asal<<endl;
     cout<<"Alamat memori palsu :"<<&palsu<<endl;
```

```
}
```

Hasil program di atas bila dimasukkan angka 90 adalah:



Latihan:

1. Buatlah algoritma dan program untuk menentukan jumlah gaji bersih dari gaji pokok yang diinputkan ditambah dengan jumlah bonus juga diinputkan. Untuk mencari bonus adalah jumlah bonus yang dimasukkan dikalikan dengan gaji pokok. Gaji bersih = gaji pokok + jumlah bonus.

   Fungsi yang diinginkan adalah:

   - Fungsi Input Data
   - Fungsi Mencari Bonus
   - Fungsi Gaji Bersih
   - Fungsi Gaji Total

2. Buatlah program dalam bentuk MENU UTAMA yang berisi operasi aritmatika penjumlahan, pengurangan, perkalian dan pembagian. Menu utama tersebut berisi 5 pilihan yang tampil seperti berikut:

   MENU UTAMA
   1. OPERASI PENJUMLAHAN
   2. OPERASI PENGURANGAN
   3. OPERASI PERKALIAN
   4. OPERASI PEMBAGIAN
   5. KELUAR

   MASUKKAN KODE [1, 2, 3, 4,5]

   Menu di atas akan di ulang secara terus menerus dengan meminta pengguna supaya memasukkan kode. Jika kode yang dimasukkan angka 5 maka akan muncul pesan “Terima Kasih”