

Materi Pertemuan 11 - 12

FUNGSI & PROSEDUR

Disusun oleh :
Danang Junaedi

OBJEKTIF

Dalam bab ini mahasiswa mempelajari tentang :

1. Pengenalan Fungsi dan Prosedur
2. Penggunaan Fungsi dan Prosedur

TUJUAN & SASARAN

Setelah mempelajari materi dalam bab ini mahasiswa diharapkan dapat :


1. Menjelaskan mengenai Fungsi/ Prosedur
2. Menjelaskan dan menggunakan variabel lokal dan global
3. Menjelaskan & menggunakan Fungsi/prosedur tanpa dan dengan nilai balik
4. Menjelaskan & menggunakan Fungsi/prosedur dengan parameter
5. Menjelaskan & menggunakan function *overloading* serta *Inline Function*

WAKTU & TEMPAT

1. 2 x (2 x 50) menit pertemuan di kelas
 - a. 50 menit materi
 - b. 50 menit tutorial (latihan soal)
2. 2 x (4 x 50) menit belajar di rumah

11-12.1 Pendahuluan

Fungsi/prosedur adalah suatu bagian dari program yang digunakan untuk menjalankan suatu tugas tertentu dan letaknya terpisah dari bagian program yang menggunakannya. Suatu fungsi/prosedur dipanggil/digunakan dengan tujuan khusus, yaitu untuk mengerjakan suatu tugas tertentu, dimana tugas-tugas tersebut dapat berupa tugas *input* (menyimpan hasil ke dalam suatu array atau file) dan/atau *output* (menampilkan hasil di layar monitor) ataupun melakukan penyeleksian dan perhitungan. Suatu Fungsi/prosedur dapat memberikan suatu hasil balik ke program yang memanggilnya atau tidak memberikan hasil balik sama sekali. Hasil balik ini biasanya berupa suatu nilai yang dibutuhkan oleh bagian program yang memanggilnya.

	<p>Perbedaan utama fungsi dan prosedur adalah :</p> <p>Fungsi hanya akan mengembalikan sebuah nilai ke bagian yang memanggilnya atau dengan kata lain sebuah fungsi hanya mengerjakan satu tugas saja, sedangkan prosedur akan mengembalikan satu atau lebih nilai atau bahkan tidak mengembalikan nilai sama sekali ke bagian yang memanggilnya, atau dengan kata lain sebuah prosedur dapat mengerjakan lebih dari satu tugas</p>
---	---

Kelebihan program yang menggunakan fungsi tambahan selain fungsi utamanya diantaranya [1] :




1. Program menjadi lebih mudah dimengerti
Hal ini disebabkan karena proses penyelesaian untuk suatu masalah dipecah menjadi beberapa sub masalah/ pemecahan masalah tersebut ke dalam bentuk fungsi yang lebih sederhana akan membuat program menjadi lebih mudah dimengerti dibandingkan jika semuanya dilakukan dalam fungsi utama saja.
2. Pengaruh antar bagian menjadi lebih kecil (Program menjadi lebih independen)
Karena suatu sub masalah diselesaikan dalam fungsi/prosedur yang terpisah maka (seharusnya) hal tersebut tidak akan mempengaruhi perintah yang ada pada fungsi/prosedur yang lain. Tidak seperti jika dikerjakan dalam fungsi utama atau bagian utama, sebuah perintah tertentu pada suatu baris program memungkinkan akan memberi pengaruh yang tidak dikehendaki ke perintah pada baris program yang lain. Dalam pembuatan sebuah fungsi/prosedur yang baik, perintah-perintah yang ada dalam suatu fungsi/prosedur seharusnya tidak mempengaruhi perintah-perintah yang ada di fungsi/prosedur yang lain (independen)
3. Dapat dipakai kembali
Fungsi/prosedur yang pernah dibuat dapat dipakai kembali di program yang lain untuk menyelesaikan masalah yang berbeda, sehingga pada pembuatan program berikutnya akan menjadi semakin mudah karena kita sudah mempunyai beberapa fungsi/prosedur tambahan yang dapat dimanfaatkan (contohnya *Library header* dalam bahasa C/C++). Hal ini akan membantu seorang *programmer* untuk menyelesaikan suatu program dengan waktu yang lebih singkat.
4. Lebih efisien dan ukuran program menjadi lebih kecil
Dengan adanya fungsi/prosedur penulisan kumpulan baris program yang sama di tempat yang berbeda dapat dikurangi atau dapat ditulis satu kali saja dalam fungsi/prosedur, jika bagian program membutuhkan perintah tersebut tinggal memanggil fungsi/prosedur tersebut. Sehingga program yang dibuat menjadi lebih efisien dan ukurannyapun menjadi lebih kecil



5. Lebih konsisten


Penulisan kumpulan baris program yang sama secara berulang-ulang ditempat yang berbeda akan mengakibatkan program menjadi tidak konsisten, karena jika terdapat kesalahan dalam kumpulan baris program tersebut maka *programmer* harus mengingat kembali posisi kumpulan baris program tersebut. Sebaliknya jika kumpulan baris program yang sama dibuat dalam bentuk fungsi/prosedur maka jika ada kesalahan perintah, *programmer* tinggal memperbaikinya pada fungsi/prosedur tersebut.

11-12.2 Pendeklarasian & Pemanggilan

Dalam bahasa C/C++, fungsi/prosedur merupakan elemen utama dari program, dimana bahasa C/C++ dibentuk dari sekumpulan fungsi/prosedur. Terdapat minimal sebuah fungsi/prosedur utama dalam bahasa C/C++ yaitu fungsi **main()**, selain itu juga bahasa C/C++ didukung oleh beberapa fungsi-fungsi yang lain diantaranya fungsi pustaka (*library header*) dan fungsi/prosedur yang dibuat oleh pemrogram itu sendiri.

	<p>Fungsi utama dalam bahasa C/C++ biasanya ditulis dengan struktur :</p> <pre>int main() { statement-statement; return 0; }</pre> <p>Jika fungsi utama tersebut telah berhasil melaksanakan tugasnya maka fungsi utama tersebut akan mengembalikan nilai 0 dan sebaliknya jika tidak berhasil maka fungsi utama tersebut akan mengembalikan nilai -1</p>
	<p>Supaya Fungsi/prosedur dapat digunakan harus dapat dipanggil dari bagian program yang membutuhkannya, untuk itu suatu Fungsi/prosedur harus diberi nama. Aturan penamaan Fungsi/prosedur sama dengan aturan penamaan variabel dan/atau konstanta.</p>
	<p>Pada C/C++, untuk mendeklarasikan fungsi tambahan kita dapat menuliskannya sebagai berikut :</p> <pre>tipe_fungsi nama_Fungsi(argumen1, argumen2,...) → definisi Fungsi { xxxx xxxx return argumen; } tubuh Fungsi</pre> <p>contoh : double Absolut(double X)</p> <pre>{ if (X,0) X=-X; return(X); }</pre> <p>Keterangan :</p> <ul style="list-style-type: none"> • Definisi Fungsi berisi tipe_Fungsi (tergantung dari tipe data hasil balik yang akan diberikan oleh Fungsi dapat berupa double, float tergantung dari ketepatan yang diinginkan. Khusus untuk Fungsi yang tidak ditulis tipenya maka akan dianggap bertipe int atau char), nama_Fungsi (merupakan nama yang dibentuk sendiri oleh <i>programmer</i>) dan argumen1, argumen 2,... atau disebut juga sebagai parameter input (sebagai alat komunikasi untuk data yang dikirimkan dari bagian program yang memanggil Fungsi. Untuk Fungsi yang tidak menggunakan argumen cara penulisannya menjadi tipe_fungsi nama_Fungsi(void)) • Tubuh Fungsi, berisi statemen-statemen program yang akan melaksanakan tugas yang diberikan, untuk bagian ini harus diawali dengan tanda { dan diakhiri dengan tanda }


	<p>Dalam bahasa C/C++ tidak mengenal istilah prosedur, yang dikenal hanya fungsi saja. Prosedur dalam bahasa C/C++ dikenal juga dengan sebutan fungsi tanpa nilai balik, untuk mendeklarasikan prosedur/fungsi tanpa nilai balik tambahan kita dapat menuliskannya sebagai berikut :</p> <p>void nama_Prosedur(argumen1, argumen2,...) → definisi Prosedur</p> <pre> { xxxx xxxx } tubuh Prosedur xxxx } </pre> <p>contoh : void Tampil(char Nama[15], int Kali)</p> <pre> { int I; for(I=0;I<Kali;I++) printf(Nama); } </pre> <p>Keterangan :</p> <ul style="list-style-type: none"> • Definisi Prosedur berisi tipe dari Prosedur yaitu void karena prosedur tidak memberikan nilai hasil balik, nama_Prosedur (merupakan nama yang dibentuk sendiri oleh pemrogram) dan argumen1, argumen 2,... atau disebut juga sebagai parameter input/output tergantung fungsi argumen tersebut (sebagai alat komunikasi untuk data yang dikirimkan dari bagian program yang memanggil atau dikirimkan dari prosedur tersebut). • Tubuh Prosedur, berisi statemen-statemen program yang akan melaksanakan tugas yang diberikan, untuk bagian ini harus diawali dengan tanda { dan diakhiri dengan tanda }
	<p>Pada C/C++, secara lengkap penulisan fungsi/prosedur tambahan dan fungsi utama serta pemaanggilannya dapat ditulis sebagai berikut :</p> <pre> #include <library header> tipe_fungsi nama_Fungsi(argumen1, argumen2,...) } prototipe fungsi void nama_Prosedur(argumen1, argumen2,...) } /Prosedur int main() { printf("%...",nama_fungsi(argumen)): atau variabel = nama_fungsi(argumen); printf("%...",variabel): } </pre> <p style="text-align: right;">} Pemanggilan fungsi</p> <pre> nama_prosedur(argumen1, argumen2,...); proses argumen ex : printf("%...",argumen1); dst return 0; } </pre> <p style="text-align: right;">} Pemanggilan Prosedur</p> <p>tipe_fungsi nama_Fungsi(argumen1, argumen2,...) → definisi Fungsi</p> <pre> { xxxx xxxx } tubuh Fungsi return argumen; } </pre> <p>void nama_Prosedur(argumen1, argumen2,...) → definisi Prosedur</p> <pre> { xxxx xxxx } tubuh Prosedur xxxx } </pre>

	<p>atau dapat dituliskan juga sebagai berikut</p> <pre>#include <library header> type_fungsi nama_Fungsi(argumen1, argumen2,...) } definisi Fungsi { xxxx xxxx return argumen; } void nama_Prosedur(argumen1, argumen2,...) → definisi Prosedur { xxxx xxxx xxxx } int main() { printf("%...",nama_fungsi(argumen)): atau variabel = nama_fungsi(argumen); printf("%...",variabel): } nama_prosedur(argumen1, argumen2,...); proses argumen ex : printf("%...",argumen1); dst } Pemanggilan Prosedur return 0; }</pre>
---	--

11-12.3 Parameter Fungsi

11-12.3.1 Pendahuluan

Parameter adalah suatu variabel yang berfungsi menampung nilai yang akan dikirimkan ke dalam fungsi atau sebaliknya menampung suatu nilai yang akan dikirimkan oleh fungsi ke bagian yang memanggilya. Dengan adanya parameter suatu fungsi akan bersifat dinamis.

	<p>Terdapat dua jenis parameter yaitu</p> <ol style="list-style-type: none"> 1. Parameter Formal, adalah parameter yang terdapat dalam pendefinisian fungsi 2. Parameter acuan, adalah parameter yang terdapat pada saat pemanggilan fungsi tersebut <p>Contoh</p> <pre>type_fungsi nama_Fungsi(<u>argumen1, argumen2,...</u>) void nama_Prosedur(<u>argumen1, argumen2,...</u>) int main() { printf("%...",nama_fungsi(<u>argumen1, argumen2,...</u>)): atau variabel = nama_fungsi(<u>argumen1, argumen2,...</u>); printf("%...",variabel): nama_prosedur(<u>argumen1, argumen2,...</u>); proses argumen ex : printf("%...",argumen1); dst return 0; }</pre> <p>Diagram labels:</p> <ul style="list-style-type: none"> Parameter formal (points to the parameter lists in the function definitions) Parameter acuan (points to the parameter lists in the function calls)
---	---

Dalam pemrograman dikenal tiga jenis parameter yaitu :

1. Parameter masukan, adalah parameter yang digunakan untuk menampung nilai yang akan dijadikan masukan (*input*) ke dalam suatu fungsi, artinya, sebuah fungsi dapat menghasilkan nilai yang berbeda tergantung dari nilai dalam parameter yang dimasukkan pada saat pemanggilan fungsi tersebut.

2. Parameter keluaran, adalah parameter yang digunakan untuk menampung nilai yang akan dijadikan keluaran (*output*) yang akan dikirimkan ke bagian yang memanggil fungsi tersebut, umumnya parameter ini digunakan untuk fungsi yang tidak memiliki nilai balik (prosedur)
3. Parameter masukan/keluaran, adalah parameter yang digunakan untuk menampung nilai yang akan dijadikan masukan (*input*) ke dalam suatu fungsi selain itu juga menampung nilai yang akan dijadikan keluaran (*output*) yang akan dikirimkan ke bagian yang memanggil fungsi tersebut, artinya, sebuah parameter sebelum fungsi dijalankan bertindak sebagai parameter masukan dan setelah fungsi dijalankan parameter tersebut bertindak sebagai parameter keluaran.

11-12.3.2 Pengiriman Parameter

Terdapat tiga jenis data yang mungkin dikirim dengan menggunakan parameter yaitu :

1. Isi atau nilai dari suatu variabel/data atau dikenal juga sebagai ***passing by value***

Karakteristik pengiriman parameter ini diantaranya :

- Yang dikirim ke fungsi/prosedur adalah nilai dari data,
- Fungsi/prosedur yang menerima kiriman nilai ini akan menyimpannya di alamat yang terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi/prosedur tersebut,
- Perubahan nilai fungsi/prosedur tidak akan mengubah nilai asli di bagian program yang memanggil fungsi/prosedur walaupun keduanya menggunakan nama variabel yang sama,
- Merupakan pengiriman searah, yaitu dari bagian program yang memanggil fungsi/prosedur ke fungsi/prosedur yang dipanggil,
- Dapat digunakan untuk suatu ungkapan, sebuah variabel atau elemen larik atau konstanta.

Contoh : Program dengan menggunakan pengiriman parameter ***passing by value***

```
//By. Danang Junaedi Teknik Informatika Universitas Widyatama
//B.421 9 Desember 2007
#include<stdio.h>
int Tambah5(int X)
{
    X = X + 5;
    printf("\n Nilai di dalam fungsi A = %d",X);
    return X;
}
int main()
{
    //deklarasi data
    int A;
    //input nilai
    printf("Masukan Nilai A : ");scanf("%d",&A);
    //menampilkan nilai awal
    printf("\n sebelum fungsi dijalankan A = %d",A);
    //panggil fungsi Tambah5
    printf("\n Hasil Fungsi %d + 5 = %d\n",A,Tambah5(A));
    //menampilkan nilai akhir
    printf("\n setelah fungsi dijalankan A = %d",A);
    return 0;
}
```

Hasil *running*:

```

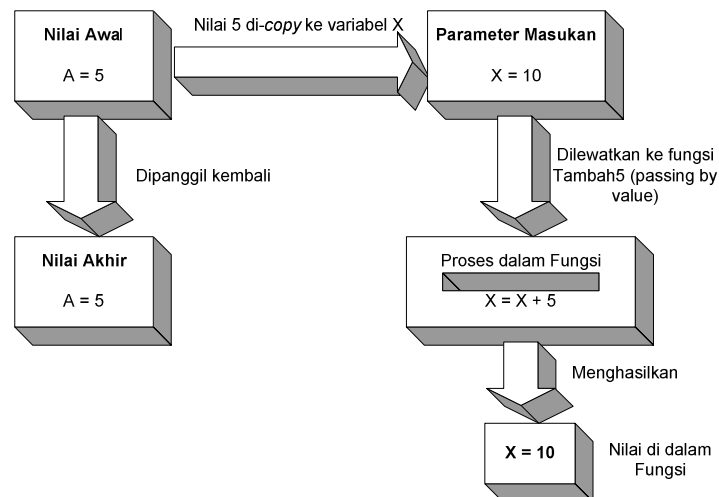
C:\WINDOWS\system32\cmd.exe
E:\ProgC++>ex11-12-01
Masukan Nilai A : 5

sebelum fungsi dijalankan A = 5
Nilai di dalam fungsi A = 10
Hasil Fungsi 5 + 5 = 10

setelah fungsi dijalankan A = 5
E:\ProgC++>

```

Ilustrasi dari program di atas adalah



2. Alamat atau Nama dari suatu variabel/data atau dikenal juga sebagai ***passing by reference***

Karakteristik pengiriman parameter ini diantaranya :

- Yang dikirim ke fungsi/prosedur adalah alamat letak dari nilai datanya,
- Fungsi/prosedur yang menerima kiriman alamat ini akan menggunakan alamat yang sama untuk mendapatkan nilai datanya,
- Perubahan nilai fungsi/prosedur akan mengubah nilai asli di bagian program yang memanggil fungsi/Prosedur,
- Merupakan pengiriman dua arah, yaitu dari bagian program yang memanggil fungsi/prosedur ke fungsi/prosedur yang dipanggil dan sebaliknya,
- Tidak dapat digunakan untuk suatu ungkapan, hanya dapat digunakan untuk sebuah variabel atau elemen larik atau konstanta saja.

Contoh : Program dengan menggunakan pengiriman parameter ***passing by reference***

```

//By. Danang Junaedi Teknik Informatika Universitas Widyatama
//B.421 9 Desember 2007
#include<stdio.h>
int Tambah5(int &A) //tanda & digunakan untuk menunjukan reference dari variabel A
{
    A = A + 5;
    printf("\n Nilai di dalam fungsi A = %d",A);
    return A;
}
int main()
{
    //deklarasi data
    int A;
    //input nilai

```

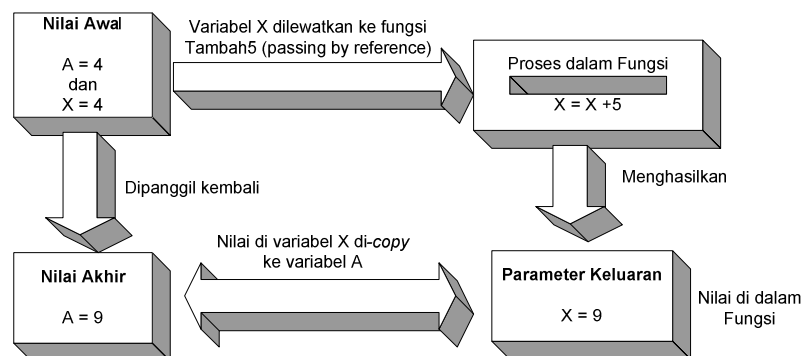
```

printf("Masukan Nilai A : ");scanf("%d",&A);
//menampilkan nilai awal
printf("\n sebelum fungsi dijalankan A = %d",A);
//panggil fungsi Tambah5
printf("\n Hasil Fungsi %d + 5 = %d\n",A,Tambah5(A));
//menampilkan nilai akhir
printf("\n setelah fungsi dijalankan A = %d",A);
return 0;
}

```

Hasil *running*:

Ilustrasi dari program di atas adalah



3. Pointer dari suatu variabel/data atau dikenal juga sebagai ***passing by pointer***

Karakteristik pengiriman parameter ini diantaranya :

- Yang dikirim ke fungsi/prosedur adalah alamat datanya,
- Fungsi/prosedur yang menerima kiriman alamat ini akan menggunakan alamat yang sama untuk mendapatkan nilai datanya,
- Perubahan nilai fungsi/prosedur akan mengubah nilai asli di bagian program yang memanggil fungsi/prosedur,
- Merupakan pengiriman dua arah, yaitu dari bagian program yang memanggil fungsi/prosedur ke fungsi/prosedur yang dipanggil dan sebaliknya,

Contoh : Program dengan menggunakan pengiriman parameter ***passing by pointer***

```

//By. Danang Junaedi Teknik Informatika Universitas Widyatama
//B.421 9 Desember 2007
#include<stdio.h>
void Tukar(int *A, int *B) //tanda * digunakan untuk menunjukan
{
    //pointer atau alamat dari variabel A dan B
    int Temp;
    Temp = *A;
    *A = *B;
    *B = Temp;
    printf("\n Nilai di dalam fungsi --> A = %d dan B = %d", *A, *B);
    printf("\n Alamat di dalam fungsi --> A = %d dan B = %d", A, B);
}

```



```

}
int main()
{
    //deklarasi data
    int A,B;

    //input nilai
    printf("Masukan Nilai A : ");scanf("%d",&A);
    printf("Masukan Nilai B : ");scanf("%d",&B);

    //menampilkan nilai awal
    printf("\n sebelum fungsi dijalankan nilai A = %d dan B = %d",A,B);
    printf("\n sebelum fungsi dijalankan alamat A = %d dan B = %d",&A,&B);

    //panggil fungsi Tukar
    Tukar(&A,&B);

    //menampilkan nilai akhir
    printf("\n setelah fungsi dijalankan nilai A = %d dan B = %d",A,B);
    printf("\n sebelum fungsi dijalankan alamat A = %d dan B = %d",&A,&B);

    return 0;
}

```

Hasil *running* :

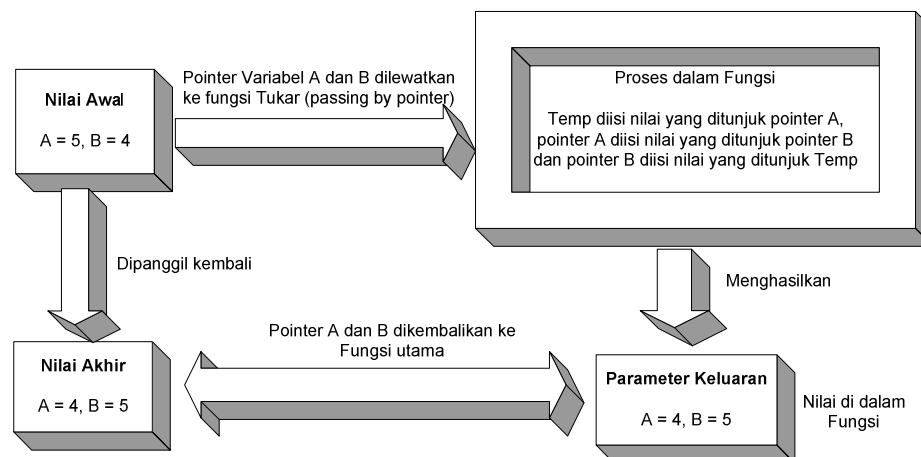
```

C:\WINDOWS\system32\cmd.exe
E:\ProgC++>ex-11-12-04
Masukan Nilai A : 5
Masukan Nilai B : 4

sebelum fungsi dijalankan nilai A = 5 dan B = 4
sebelum fungsi dijalankan alamat A = 2293620 dan B = 2293616
Nilai di dalam fungsi --> A = 4 dan B = 5
Alamat di dalam fungsi --> A = 2293620 dan B = 2293616
setelah fungsi dijalankan nilai A = 4 dan B = 5
sebelum fungsi dijalankan alamat A = 2293620 dan B = 2293616
E:\ProgC++>

```

Ilustrasi dari program di atas adalah



11-12.4 *Function Overloading* (Penimpaan Fungsi)

Function overloading (penimpaan fungsi) adalah salah satu kelebihan C++ dibandingkan bahasa C. *Function overloading* (penimpaan fungsi) didefinisikan sebagai fungsi-fungsi dengan nama yang sama tetapi memiliki parameter yang berbeda, ditinjau dari segi :

1. Jumlah Parameter

Contoh : Program dengan menggunakan *Function Overloading* ditinjau dari segi jumlah parameter

```
// Referensi nomor 3, halaman 178
```

```
//B.423, 13 Desember 2007
#include <iostream>
//Mendefinisikan fungsi Tulis dengan satu parameter
void Tulis(char *S)
{
    cout<<S<<endl;
}
//Mendefinisikan fungsi Tulis dengan dua parameter
void Tulis(char *S1, char *S2)
{
    cout<<S1<<" "<<S2<<endl;
}
//Mendefinisikan fungsi Tulis dengan tiga parameter
void Tulis(char *S1, char *S2, char *S3)
{
    cout<<S1<<" "<<S2<<" "<<S3<<endl;
}
//Fungsi utama
int main()
{
    //Pemanggilan fungsi
    Tulis("Teknik Informatika");
    Tulis("Teknik Informatika","- Fakultas Teknik");
    Tulis("Teknik Informatika","- Fakultas Teknik","- Universitas Widyatama");
    return 0;
}
```

Hasil *running* :



```
C:\WINDOWS\system32\cmd.exe
D:\ProgC++>ex11-12-04
Teknik Informatika
Teknik Informatika - Fakultas Teknik
Teknik Informatika - Fakultas Teknik - Universitas Widyatama
D:\ProgC++>
```

2. Tipe Data Parameter

Contoh : Program dengan menggunakan *Function Overloading* ditinjau dari segi tipe parameter

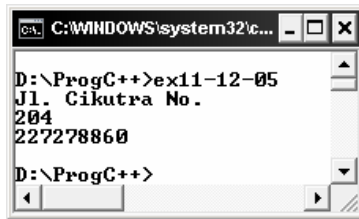
```
// Referensi nomor 3, halaman 178 - 180
//B.423, 13 Desember 2007
#include <iostream>
//Mendefinisikan fungsi Tulis dengan parameter bertipe string (char *)
void Tulis(char *S)
{
    cout<<S<<endl;
}
//Mendefinisikan fungsi Tulis dengan parameter bertipe integer (int)
void Tulis(int S)
{
    cout<<S<<endl;
}
//Mendefinisikan fungsi Tulis dengan parameter bertipe real (float)
void Tulis(float S)
{
    cout<<S<<endl;
}
//Fungsi utama
int main()
{
    //Pemanggilan fungsi
```

```

    Tulis("Jl. Cikutra No.");
    Tulis(204);
    Tulis(227278860);
    return 0;
}

```

Hasil *running* :



3. Jumlah Parameter dan Tipe Data Parameter

Contoh : Program dengan menggunakan *Function Overloading* ditinjau dari segi jumlah parameter dan tipe data parameter

```

// Referensi nomor 3, halaman 180 - 181
//B.423, 13 Desember 2007
#include <iostream>

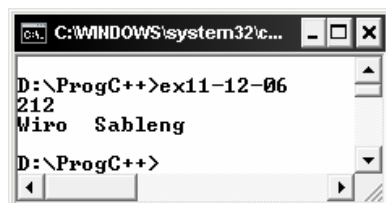
//Mendefinisikan fungsi Tulis dengan satu parameter dan bertipe integer (int)
void Tulis(int S)
{
    cout<<S<<endl;
}

//Mendefinisikan fungsi Tulis dengan dua parameter dan bertipe String (char *)
void Tulis(char *S1, char *S2)
{
    cout<<S1<<" "<<S2<<endl;
}

//Fungsi utama
int main()
{
    //Pemanggilan fungsi
    Tulis(212);
    Tulis("Wiro","Sableng");
    return 0;
}

```

Hasil *running* :



Dalam *Function overloading* (penimpaan fungsi), pada saat kita melakukan pemanggilan fungsi, kompiler akan memilih fungsi yang parameter aktualnya sesuai dengan parameter formalnya (apa perbedaan parameter formal dan aktual ? lihat lagi halaman M-XI/XII-11 di atas)

11-12.5 *Inline Function* (Fungsi *Inline*)

Pada saat kita mendefinisikan sebuah fungsi, kompiler akan membuat satu set statemen di dalam memori. Ketika pemanggilan fungsi tersebut, eksekusi program akan loncat ke statemen-statemen tersebut. Kemudian setelah fungsi mengembalikan nilai (atau proses dalam fungsi selesai

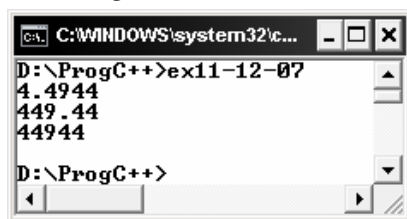
dilaksanakan), maka eksekusi program akan kembali loncat ke baris selanjutnya (di fungsi/program yang memanggilnya). Jika kita melakukan pemanggilan fungsi tersebut sebanyak lima kali, maka kompiler juga akan melakukan proses tersebut sebanyak lima kali. Apabila fungsi yang kita definisikan hanya terdiri dari sedikit statemen, hal ini tentu akan mengurangi efisiensi program karena kompiler harus keluar masuk ke fungsi tersebut. Kata “efisien” bagi seorang *programmer* tentu akan dihubungkan dengan masalah “kecepatan” eksekusi sebuah program. Untuk menghindari hal ini, C++ menyediakan fitur yang disebut fungsi *inline* (*inline function*), dengan menggunakan *keyword inline*.

Konsep dasar dari *inline function* adalah proses penyalinan (*copy*) baris yang terdapat pada definisi fungsi ke baris pada saat kita melakukan pemanggilan fungsi tersebut. Artinya jika kita membuat fungsi *inline*, kompiler tidak akan menyimpannya ke dalam memori melainkan hanya akan membuat salinan kode dari fungsi tersebut. Hal ini tentu tidak membutuhkan proses loncatan statemen seperti pada saat kita membuat fungsi biasa, sehingga proses eksekusinya akan lebih cepat.

Contoh : Program dengan menggunakan *Inline Function*

```
// Referensi nomor 3, halaman 184
//B.423, 13 Desember 2007
#include <iostream>
//Mendefinisikan fungsi kuadrat (pangkat 2) dengan menggunakan inline function
inline float Kuadrat(float Bilangan)
{
    return Bilangan * Bilangan; //Bilangan2
}
//Fungsi utama
int main()
{
    //Pemanggilan fungsi pertama kali
    cout<<Kuadrat(2.12)<<endl;
    //Pemanggilan fungsi kedua kali
    cout<<Kuadrat(21.2)<<endl;
    //Pemanggilan fungsi ketiga kali
    cout<<Kuadrat(212)<<endl;
    return 0;
}
```

Hasil *running*:



11-12.6 Referensi

1. Hartanto, Budi, “Pembuatan Program C Secara Mudah”, Andi, Yogyakarta., 2004 [Bab 8]
2. Joni, I Made; Raharjo, Budi, “Pemrograman C dan Implementasinya”, Informatika, Bandung, 2006 [Bab 5]
3. Raharjo, Budi, “Pemrograman C++ Mudah dan Cepat Menjadi Master C++ dengan Mengungkap Rahasia-Rahasia Pemrograman dalam C++”, Informatika, 2006 [Bab 8]

11-12.7 Bahan Renungan (Semoga bisa menjadi tambahan motivasi)

Aku bermimpi suatu hari aku pergi ke surga dan seorang malaikat menemaniku dan menunjukkan keadaan di surga. Kami berjalan memasuki suatu ruang kerja penuh dengan para malaikat.

Malaikat yang mengantarku berhenti di depan ruang kerja pertama dan berkata, " Ini adalah Seksi Penerimaan. Disini, semua permintaan yang ditujukan pada Allah diterima". Aku melihat-lihat sekeliling tempat ini dan aku dapati tempat ini begitu sibuk dengan begitu banyak malaikat yang memilah-milah seluruh permohonan dari manusia di seluruh dunia.

Kemudian aku dan malaikat-ku berjalan lagi melalui koridor yang panjang lalu sampailah kami pada ruang kerja kedua. Malaikat-ku berkata, "Ini adalah Seksi Pengemasan dan Pengiriman. Disini kemuliaan dan rahmat yang diminta manusia diproses dan dikirim ke manusia-manusia yang masih hidup yang memintanya". Aku perhatikan lagi betapa sibuknya ruang kerja itu. Ada banyak malaikat yang bekerja begitu keras karena ada begitu banyaknya permohonan yang dimintakan dan sedang dipaketkan untuk dikirim ke bumi.

Kami melanjutkan perjalanan lagi hingga sampai pada ujung terjauh koridor panjang tersebut dan berhenti pada sebuah pintu ruang kerja yang sangat kecil. Yang sangat mengejutkan aku, hanya ada satu malaikat yang duduk disana, hampir tidak melakukan apapun. Ini adalah Seksi Pernyataan Terima Kasih", kata Malaikatku pelan. Dia tampak malu. "Bagaimana ini? Mengapa hampir tidak ada pekerjaan disini?", tanyaku. "Menyedihkan", Malaikat-ku menghela napas. " Setelah manusia menerima rahmat yang mereka minta, sangat sedikit manusia yang mengirimkan pernyataan terima kasih". "Bagaimana manusia menyatakan terima kasih atas rahmat Tuhan?", tanyaku. "Sederhana sekali", jawab Malaikat. "Cukup berkata, 'ALHAMDULILLAH RABBIL AALAMIIN, Terima kasih, Tuhan' ", lanjutnya.

"Lalu, rahmat apa saja yang perlu kita syukuri", tanyaku. Malaikat menjawab,

"Jika engkau mempunyai makanan di lemari es, pakaian yang menutup tubuhmu, atap di atas kepalamu dan tempat untuk tidur, maka engkau lebih kaya dari 75% penduduk dunia ini.

"Jika engkau memiliki uang di bank, di dompetmu, dan uang-uang receh, maka engkau berada diantara 8% kesejahteraan dunia.

Juga....

"Jika engkau bangun pagi ini dengan lebih banyak kesehatan daripada kesakitan ... engkau lebih dirahmati daripada begitu banyak orang di dunia ini yang tidak dapat bertahan hidup hingga hari ini.

"Jika engkau tidak pernah mengalami ketakutan dalam perang, kesepian dalam penjara, kesengsaraan penyiksaan, atau kelaparan yang amat sangatmaka engkau lebih beruntung dari 700 juta orang di dunia".

"Jika engkau dapat menghadiri Masjid atau pertemuan religius tanpa ada ketakutan akan penyerangan, penangkapan, penyiksaan, atau kematian ... maka engkau lebih dirahmati daripada 3 milyar orang di dunia.

"Jika orangtuamu masih hidup dan masih berada dalam ikatan pernikahanmaka engkau termasuk orang yang sangat jarang.

"Jika engkau dapat menegakkan kepala dan tersenyum, maka engkau bukanlah seperti orang kebanyakan, engkau unik dibandingkan semua mereka yang berada dalam keraguan dan keputusasaan.

"Jika engkau dapat membaca pesan ini, maka engkau menerima rahmat ganda, yaitu bahwa seseorang yang mengirimkan ini padamu berpikir bahwa engkau orang yang sangat istimewa baginya, dan bahwa, engkau lebih dirahmati daripada lebih dari 2 juta orang di dunia yang bahkan tidak dapat membaca sama sekali".

Nikmatilah hari-harimu, hitunglah rahmat yang telah Allah anugerahkan kepadamu. "Dan ingatlah tatkala Tuhanmu menyatakan bahwa,'Sesungguhnya jika kamu bersyukur, pasti Aku akan menambahkan lebih banyak nikmat kepadamu' ".(QS:Ibrahim (14) :7)

Surat dari Tuhan

Saat kau bangun dipagi hari, Aku memandangmu dan berharap engkau akan berbicara kepadaKu, walaupun hanya sepatah kata, meminta pendapatKu atau bersyukur kepadaKu atas sesuatu hal indah yang terjadi di dalam hidupmu kemarin, tetapi aku melihat engkau begitu sibuk mempersiapkan diri untuk pergi bekerja.

Aku kembali menanti.

Saat engkau sedang bersiap, Aku tahu akan ada sedikit waktu bagimu untuk berhenti dan menyapaKu, tetapi engkau terlalu sibuk. Di satu tempat, engkau duduk di sebuah kursi selama lima belas menit tanpa melakukan apapun.

Kemudian Aku melihat engkau menggerakkan kakimu. Aku berpikir engkau ingin berbicara kepadaKu, tetapi engkau berlari ke telepon dan menelepon seorang teman untuk mendengarkan gosip terbaru. Aku melihatmu ketika engkau pergi bekerja dan Aku menanti dengan sabar sepanjang hari. Dengan semua kegiatanmu, Aku berpikir engkau terlalu sibuk untuk mengucapkan sesuatu kepadaKu. Sebelum makan siang Aku melihatmu memandang kesekeliling, mungkin engkau merasa malu untuk berbicara kepadaKu, itulah sebabnya mengapa engkau tidak menundukkan kepalamu. Engkau memandang tiga atau empat meja sekitarmu dan melihat beberapa temanmu berbicara kepadaku dengan lembut sebelum mereka makan, tetapi engkau tidak melakukannya.

Tidak apa-apa.

Masih ada waktu yang tersisa, dan Aku berharap engkau akan berbicara kepadaKu, meskipun saat engkau pulang ke rumah kelihatannya seakan-akan banyak hal yang harus kau kerjakan. Setelah beberapa hal tersebut selesai engkau kerjakan, engkau menyalakan televisi, Aku tidak tahu apakah kau suka menonton televisi atau tidak, hanya saja engkau selalu kesana dan menghabiskan banyak waktu setiap hari didepannya, tanpa memikirkan apapun hanya menikmati acara yang ditampilkan.

Kembali Aku menanti dengan sabar saat engkau menonton TV dan menikmati makananmu tetapi kembali kau tidak berbicara kepadaKu. Saat tidur Kupikir kau merasa terlalu lelah. Setelah mengucapkan selamat malam kepada keluargamu, kau melompat ke tempat tidur dan tertidur tak lama kemudian. Tidak apa-apa karena mungkin engkau tidak menyadari bahwa Aku selalu hadir untukmu. Aku telah bersabar lebih lama dari yang kau sadari. Aku bahkan ingin mengajarkanmu bagaimana bersabar terhadap orang lain. Aku sangat mengasihimu, setiap hari Aku menantikan sepatah kata, doa atau pikiran atau syukur dari hatimu.

Baiklah... engkau bangun kembali dan kembali.

Aku akan menanti dengan penuh kasih bahwa hari ini kau akan memberiKu sedikit waktu. Semoga harimu menyenangkan.