



FUNGSI DAN PROSEDUR



FUNGSI

- „Adalah sekumpulan perintah operasi program yang dapat menerima argumen input dan dapat memberikan hasil output yang dapat berupa nilai ataupun sebuah hasil operasi.
- „Nama fungsi yang didefinisikan sendiri oleh pemrogram tidak boleh sama dengan nama build-in function pada compiler C++.
- „Fungsi digunakan agar pemrogram dapat menghindari penulisan bagian program (kode) berulang-ulang, dapat menyusun kode program agar terlihat lebih rapi dan kemudahan dalam debugging program.



PROSEDUR

- „ Berbeda dengan bahasa pemrograman lainnya, prosedur pada pemrograman C++ dinyatakan dalam fungsi
- „ Sintak/Penulisan Prosedur pada C++ adalah sama persis dengan penulisan Fungsi
- „ Perbedaan Prosedur dan Fungsi pada C++ adalah bahwa Prosedur tidak memiliki nilai pengembalian (*return value*)



FUNGSI, DEKLARASI DAN DEFINISINYA

- Pemrogram dapat membuat fungsi yang didefinisikan sendiri olehnya.

Contoh:

```
// Fungsi kuadrat
// tipe_return nama_fungsi (tipe_argument argumen)

float kuadrat ( float x )
{
    return x*x;
}
```



FUNGSI, DEKLARASI DAN DEFINISINYA

- Fungsi yang didefinisikan oleh pemrogram terdiri atas dua bagian, yaitu judul (header) dan isi (body). Judul dari sebuah fungsi terdiri dari tipe return (float), nama fungsi (kuadrat) dan list parameter (float x).

- Jadi, judul untuk fungsi kuadrat adalah

`float kuadrat (float x)`

- Isi dari sebuah fungsi adalah blok kode yang mengikuti judulnya. Berisi kode yang menjalankan aksi dari fungsi, termasuk pernyataan return yang memuat nilai fungsi yang akan dikembalikan ke yang memangginya, Isi dari fungsi kuadrat() adalah

```
{  
    return x*x;  
}
```



FUNGSI, DEKLARASI DAN DEFINISINYA

- Biasanya isi dari fungsi cukup besar. Meskipun demikian, judulnya tetap hanya berada dalam satu baris. Isi dari sebuah fungsi dapat memanggil fungsi itu sendiri (disebut rekursif) atau memanggil fungsi lainnya.
- Pernyataan return dari sebuah fungsi mempunyai dua manfaat, yaitu akan mengakhiri fungsi dan mengembalikan nilainya ke program pemanggil.
- Bentuk umum pernyataan return adalah:

```
return ekspresi;
```
- Dengan ekspresi adalah sebuah ekspresi yang nilainya dinyatakan untuk sebuah variable yang tipenya sama seperti tipe return. Terdapat juga fungsi yang tidak memberikan nilai return atau tipe returnnya void.

A decorative graphic on the left side of the slide consists of overlapping yellow, red, and blue squares with a black crosshair.

Contoh

```
#include <iostream.h>
void sayHello(char[]) ; // deklarasi fungsi sayHello()

void main()
{
    char n[50];
    cout<<"Masukkan nama anda : "; cin>>n;
    sayHello(n);
}

void sayHello(char nama[]) // definisi fungsi sayHello() {
    cout<<"Selamat datang "<<nama;

}
```



FUNGSI, DEKLARASI DAN DEFINISINYA

- Pengertian deklarasi fungsi berbeda dengan dengan definisi fungsi.
- Suatu deklarasi fungsi adalah judul fungsi yang sederhana yang diikuti oleh tanda semicolon (;).
- Sedangkan definisi fungsi adalah fungsi yang lengkap, terdiri dari judul dan isinya.
- Suatu deklarasi fungsi disebut juga sebagai prototype fungsi.
- Suatu deklarasi fungsi seperti layaknya suatu deklarasi variabel, yang memberitahu compiler semua informasi yang dibutuhkan untuk mengkompilasi file.
- Compiler tidak perlu mengetahui bagaimana fungsi bekerja, yang perlu diketahui adalah nama fungsi, jumlah dan tipe parameternya, dan tipe balikkannya (return).
- Hal ini merupakan informasi yang dimuat secara lengkap dalam judul fungsi.



FUNGSI, DEKLARASI DAN DEFINISINYA

- Seperti sebuah deklarasi variabel, suatu deklarasi fungsi harus muncul diatas semua nama fungsi yang digunakannya.
- Berbeda dengan definisi fungsi, yang dapat diletakkan terpisah dari deklarasinya, dan dapat muncul dimana saja diluar fungsi main() dan biasanya dituliskan setelah fungsi main() atau dalam file terpisah yang jika ingin digunakan tinggal menambahkan preprocessor `#include "nama_file"` pada file utama.
- Jika definisi fungsi diletakkan diatas fungsi main() maka deklarsi fungsi tidak diperlukan.



FUNGSI, DEKLARASI DAN DEFINISINYA

- Variabel-variabel yang di list di dalam parameter fungsi disebut parameter-parameter formal atau argumen-argumen formal.
- Variabel lokal seperti ini hanya ada selama eksekusi fungsi yang bersangkutan. Dalam contoh dibawah, parameter-parameter formalnya adalah x dan y.
- Variabel yang dilist dalam pemanggilan fungsi disebut parameter-parameter actual atau argumen-argumen aktual.
- Sama seperti variabel lainnya dalam program utama, variabel-variabel tersebut harus dideklarasikan sebelum digunakan dalam pemanggilan.
- Dalam contoh dibawah, parameter-parameter aktualnya adalah m dan n.



FUNGSI, DEKLARASI DAN DEFINISINYA

- Function adalah satu blok instruksi yang dieksekusi ketika dipanggil dari bagian lain dalam suatu program.

Format dari function :

type name (argument1, argument2, ...) statement



FUNGSI, DEKLARASI DAN DEFINISINYA

- Dimana :
 - type, adalah tipe dari data yang akan dikembalikan/dihasilkan oleh function.
 - name, adalah nama yang memungkinkan kita memanggil function.
 - arguments (dispesifikasikan sesuai kebutuhan). Setiap argumen terdiri dari tipe data diikuti identifier, seperti deklarasi variable (contoh, int x) dan berfungsi dalam function seperti variable lainnya. Juga dapat melakukan passing parameters ke function itu ketika dipanggil. Parameter yang berbeda dipisahkan dengan koma.
 - statement, merupakan bagian badan suatu function. Dapat berupa instruksi tunggal maupun satu blok instruksi yang dituliskan diantara kurung kurawal {}.



FUNGSI, DEKLARASI DAN DEFINISINYA

Contoh function 1 :

- // function example
- #include <iostream.h>
- int addition (int a, int b)
- {
- int r;
- r=a+b;
- return (r);
- }
- int main ()
- {
- int z;
- z = addition (5,3);
- cout << "The result is " << z;
- return 0;
- }

Output :

The result is 8



FUNGSI, DEKLARASI DAN DEFINISINYA

- Program diatas, ketika dieksekusi akan mulai dari fungsi main. main function memulai dengan
- deklarasi variabel z dengan tipe int. Setelah itu instruksi pemanggilan fungsi addition. Jika
- diperhatikan, ada kesamaan antara sruktur pemanggilan dengan deklarasi fungsi itu sendiri,
- perhatikan contoh dibawah ini :

```
int addition (int a, int b)
              ↑       ↑
z = addition ( 5 , 3 ) ;
```



FUNGSI, DEKLARASI DAN DEFINISINYA

- Instruksi pemanggilan dalam fungsi main untuk fungsi addition, memberikan 2 nilai : 5 dan 3
- mengacu ke parameter int a dan int b yang dideklarasikan untuk fungsi addition.
- Saat fungsi dipanggil dari main, kontrol program beralih dari fungsi main ke fungsi addition.
- Nilai dari kedua parameter yang diberikan (5 dan 3) di-copy ke variable local ; int a dan int b

FUNGSI, DEKLARASI DAN DEFINISINYA

Fungsi addition mendeklarasikan variable baru (`int r;`), kemudian ekspresi `r=a+b;`, yang berarti `r` merupakan hasil penjumlahan dari `a` dan `b`, dimana `a` dan `b` bernilai 5 dan 3 sehingga hasil akhirnya 8. perintah selanjutnya adalah :

`return (r);`

Merupakan akhir dari fungsi addition, dan mengembalikan kontrol pada fungsi main.

Statement return diikuti dengan variabel `r` (`return (r);`), sehingga nilai dari `r` yaitu 8 akan dikembalikan :

```
int addition (int a, int b)
    ↓ 8
z = addition ( 5 , 3 );
```

Dengan kata lain pemanggilan fungsi (`addition (5,3)`) adalah menggantikan dengan nilai yang akan dikembalikan (8).

A decorative graphic on the left side of the slide consists of overlapping yellow, red, and blue squares with a black crosshair.

Function tanpa Tipe (Kegunaan Void)

Function tanpa tipe (Kegunaan void) Deklarasi fungsi akan selalu diawali dengan tipe dari fungsi, yang menyatakan tipe data apa yang akan dihasilkan dari fungsi tersebut. Jika tidak ada nilai yang akan dikembalikan, maka dapat digunakan tipe void, contoh :

```
// void function example
#include <iostream.h>
void dummyfunction (void)
{
    cout << "I'm a function!";
}
int main ()
{
    dummyfunction ();
    return 0;
}
```

Output :
I'm a function!

A decorative graphic on the left side of the slide consists of overlapping yellow, red, and blue squares with a black crosshair.

Function tanpa Tipe (Kegunaan Void)

Function tanpa tipe (Kegunaan void) Deklarasi fungsi akan selalu diawali dengan tipe dari fungsi, yang menyatakan tipe data apa yang akan dihasilkan dari fungsi tersebut. Jika tidak ada nilai yang akan dikembalikan, maka dapat digunakan tipe void, contoh :

```
// void function example
#include <iostream.h>
void dummyfunction (void)
{
    cout << "I'm a function!";
}
int main ()
{
    dummyfunction ();
    return 0;
}
```

Output :
I'm a function!



Prototyping function.

Format :

`type name (argument_type1, argument_type2, ...);`

Hampir sama dengan deklarasi fungsi pada umumnya, kecuali :

- Tidak ada statement fungsi yang biasanya dituliskan dalam kurung kurawal { }.
- Diakhiri dengan tanda semicolon (;).
- Dalam argumen dituliskan tipe argumen, bersifat optional.

Prototyping function.

Contoh:

```
// prototyping
#include <iostream.h>
void odd (int a);
void even (int a);
int main ()
{
    int i;
    do {
        cout << "Type a number: (0 to exit)";
        cin >> i;
        odd (i);
    } while (i!=0);
    return 0;
}
```

```
void odd (int a)
{
    if ((a%2)!=0) cout << "Number is odd.\n";
    else even (a);
}
void even (int a)
{
    if ((a%2)==0) cout << "Number is even.\n";
    else odd (a);
}
```

Output :

```
Type a number (0 to exit): 9
Number is odd.
Type a number (0 to exit): 6
Number is even.
Type a number (0 to exit): 1030
Number is even.
Type a number (0 to exit): 0
Number is even.
```

A decorative graphic on the left side of the slide consists of overlapping yellow, red, and blue squares with a black crosshair.

Prototyping function.

Contoh diatas tidak menjelaskan tentang efektivitas program tetapi bagaimana prototyping dilaksanakan.

Perhatikan prototype dari fungsi odd dan even:

```
void odd (int a);  
void even (int a);
```

Memungkinkan fungsi ini dipergunakan sebelum didefinisikan. Hal lainnya mengapa program diatas harus memiliki sedikitnya 1 fungsi prototype, karena fungsi dalam odd terdapat pemanggilan fungsi even dan dalam even terdapat pemanggilan fungsi odd. Jika tidak satupun dari fungsi tersebut dideklarasikan sebelumnya, maka akan terjadi error.




Contoh

```
// Penggunaan Fungsi Rekursif :  
// Program mengecek sebuah bilangan integer atau bukan  
#include <iostream.h>  
#include <conio.h>  
#include <math.h>  
void cekInt(double);  
void main()  
{  
    double angka;  
    cout<<"Masukan sebuah angka :";cin>>angka;  
    cekInt(angka);  
}  
void cekInt(double n)  
{  
    if(n>1)cekInt(n-1);  
    else if(n<1)cekInt(-n-1);  
    else  
    {  
        if(n>0&& n<1)cout<<n<<"\t Bukan bilangan bulat\n";  
        else cout<<n<<"\t Bilangan bulat\n";  
    }  
}
```



NILAI BAWAAN UNTUK ARGUMEN FUNGSI

- „ Salah satu keistimewaan C++ yang sangat bermanfaat dalam pemrograman adalah adanya kemampuan untuk menyetel nilai defaultArgumen 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.

A decorative graphic is located on the left side of the slide. It consists of a black crosshair centered over a yellow square, which is partially overlaid by a red square and a blue square.

```
#include <iostream.h>
#include <conio.h>
void sayHello(int);
void main()
{
    sayHello();
}
void sayHello(int N=1)
{
    for(int N=0;N<N;M++) cout<<"Halloo \n";
}
```




FUNGSI-FUNGSI BAWAAN C++

- Fungsi-fungsi bawaan C++, misalkan fungsi-fungsi matematika, pengolah kata dan banyak lagi. Sebenarnya (mungkin tidak terasa bagi anda) main juga adalah fungsi, jadi tanpa anda sadari sebenarnya anda telah menggunakan fungsi.
- Untuk dapat menggunakan fungsi-fungsi tersebut anda harus meng-include file dimana fungsi tersebut didefinisikan
- Misalkan :
 - Fungsi - fungsi matematika, anda harus meng-include file math.h
 - Fungsi - fungsi pengolah string dan karakter, anda harus meng-include file string.h
 - Fungsi clrscr(), getch(), getche() dalam file conio.h