

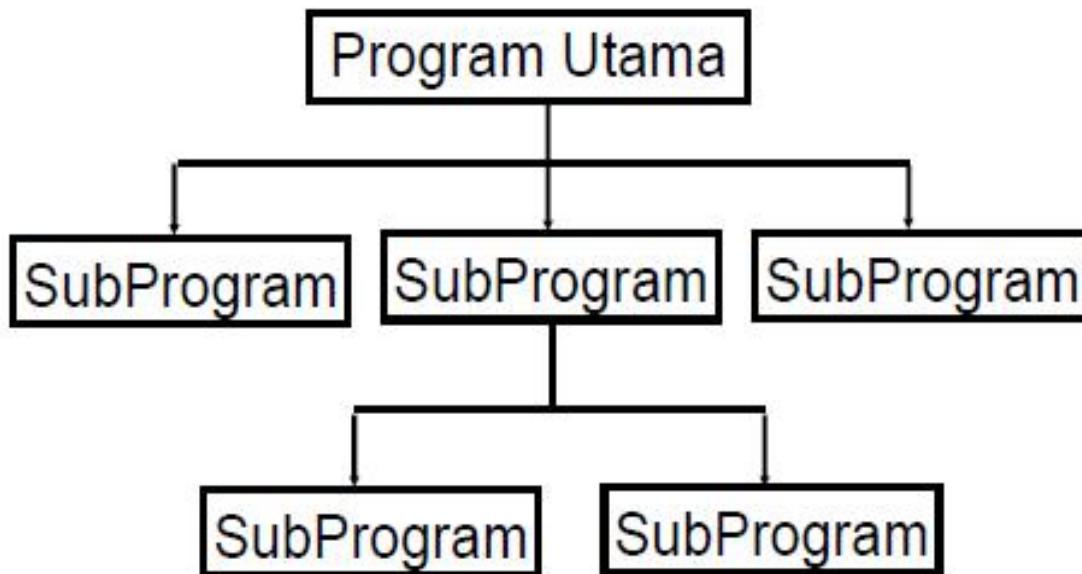


Modular Concept

Modular Concept

- ❖ Membagi program menjadi beberapa bagian *procedure* (prosedur) dan *function* (fungsi).
- ❖ Prosedur adalah bagian program yang tidak mengembalikan hasil proses ke bagian pemanggilnya.
- ❖ Fungsi adalah bagian program yang akan mengembalikan suatu nilai hasil dari proses ke bagian pemanggilnya.

Analogi



Fungsi / Function

- Modul pada bahasa C/C++ dikenal dengan nama **fungsi (function)**
- Bahasa C terdiri dari fungsi-fungsi, baik yang **langsung dideklarasikan** dalam program ataupun **dipisah** di dalam header file.
- Fungsi yang selalu ada pada program C++ adalah fungsi **main**
- Contoh fungsi standart:
 - **printf**
 - **scanf**



Tujuan pembuatan Fungsi

- ❖ Program menjadi terstruktur sehingga lebih mudah dipahami.
- ❖ Mengurangi pengulangan (duplikasi) penulisan kode program :
 - Langkah-langkah program yang sama dan dipakai berulang-ulang dapat dituliskan sekali saja sebagai fungsi.

Keuntungan

- Mudah dipahami
- Mudah digunakan kembali
- Program lebih pendek
- Mudah didokumentasi
- Mengurangi kesalahan
- Mudah mencari kesalahan
- Kesalahan yang terjadi bersifat “lokal”

Standard Library Function

- Disediakan oleh bahasa pemrograman dalam file-file header atau librarynya

Programmer Defined Function

- Dibuat sendiri oleh programmer

Penulisan Fungsi

- ❖ **tipe-keluaran** dapat berupa salah satu tipe data C, misalnya char atau int. Kalau tipenya tidak disebut maka dianggap bertipe int (secara default).
- ❖ **tubuh fungsi** berisi deklarasi variabel (kalau ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan.
- ❖ **nama_fungsi** digunakan untuk memanggil fungsi.
- ❖ **argument** berisi parameter-parameter fungsi.

Bentuk Umum

```
tipe-keluaran-fungsi nama-fungsi (deklarasi argumen)
{
    tubuh fungsi
}
```

Penulisan Fungsi

```
inialisasi() {  
    return(0);  
}
```

Annotations:

- inialisasi() ← Nama fungsi
- { ← Sepasang tanda kurung, tanpa argumen
- ← Tak ada tanda titik koma
- ← Awal fungsi
- return(0) ; ← Tubuh fungsi
- } ← Akhir fungsi

Pemanggilan Fungsi

```
int inisialisasi ();  
main()  
{  
    int x, y;  
  
    x = inisialisasi(); ←  
    printf("x = %d\n", x);  
    y = inisialisasi(); ←  
    printf("y = %d\n", y);  
}
```

```
int inisialisasi()  
{  
    return(0);  
}
```

definisi fungsi
pemanggilan fungsi

Tipe Fungsi

❖ Fungsi yang tidak mempunyai output (void)

```
void info_program()
{
    printf("Designed Program by \n");
    printf("Lab. Kom. Digital \n");
    printf("STSN \n");
}
```

❖ Fungsi yang mempunyai output (non-void).

```
int kuadrat(int b)
{
    return(b * b);
}
```

Tipe Fungsi :Void

- Fungsi yang void sering disebut juga prosedur
- Disebut void karena fungsi tersebut tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut.

Tipe Fungsi :Void

Tidak adanya tipe
data di dalam
deklarasi fungsi.

Menggunakan
keyword void.

Tidak memiliki
nilai kembalian
fungsi

Tipe Fungsi : Non-void

- Fungsi non-void disebut juga **function**
- Disebut non-void karena mengembalikan nilai kembalian yang berasal dari keluaran hasil proses function tersebut

Tipe Fungsi : Non-void

Ada keyword return

Ada tipe data yang mengawali deklarasi fungsi

Tidak ada keyword void

Memiliki nilai kembalian

Dapat dianalogikan sebagai suatu variabel yang memiliki tipe data tertentu sehingga dapat langsung ditampilkan hasilnya

Contoh Fungsi void

```
#include<stdio.h>

void info_program(); //Prototype Fungsi

main()
{
    printf("\nInfo Pembuat Program \n");
    info_program();
    getchar();
    info_program();
}

void info_program() //Definisi Fungsi
{
    printf("Program Designed by \n");
    printf("Lab. Kom. Digital \n");
    printf("STSN \n");
}
```

Contoh Fungsi non-void

```
#include <stdio.h>

int kuadrat (int b); //Prototype Fungsi

main()
{
    int pangkat;
    printf("Kuadrat 2 adl %d \n", kuadrat(2));
    printf("Kuadrat 3 adl %d \n", kuadrat(3));
    pangkat = kuadrat (5);
    printf("Kuadrat 5 adl %d \n", pangkat);
}

int kuadrat(int b) //Definisi Fungsi
{
    return(b * b);
}
```

Example I

❖ Fungsi pembagian

```
1 #include <stdio.h>
2 #include <conio.h>
3
4 double pembagian(double bil1, double bil2)
5 {
6     double hasil_pembagian;
7     hasil_pembagian = bil1/bil2;
8     return hasil_pembagian;
9 }
10
11 main()
12 {
13     double x,y,z;
14     x = 10;
15     y = 3;
16
17     z = pembagian(x,y);
18     printf("hasil pembagian dari %.0lf dibagi %.0lf adalah %.2lf", x, y, z);
19     getch();
20 }
```

Example 2

❖ Fungsi pembagian

```
1 #include <stdio.h>
2 #include <conio.h>
3
4
5 double pembagian(double bil1, double bil2); Prototype fungsi
6
7 main()
8 {
9     double x,y,z;
10    x = 10;
11    y = 3;
12
13    z = pembagian(x,y);
14    printf("hasil pembagian dari %.0lf dibagi %.0lf adalah %.2lf", x, y, z);
15    getch();
16 }
17
18 double pembagian(double bil1, double bil2)
19 {
20     double hasil_pembagian;
21     hasil_pembagian = bil1/bil2;
22     return hasil_pembagian;
23 }
```

Prototipe Fungsi

- ❖ Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan,
- ❖ deklarasi fungsi dikenal dengan sebutan prototipe fungsi.
- ❖ Prototipe fungsi berupa :
 - ❖ Tipe nilai fungsi
 - ❖ Nama Fungsi
 - ❖ Jumlah dan tipe parameter (argumen)
 - ❖ Dan diakhiri dengan titik koma, sebagaimana pada pendeklarasian variabel.

Variabel Lokal

Ciri-ciri variabel lokal adalah:

- ❖ Identifier yang dideklarasikan di dalam fungsi, termasuk daftar parameter.
- ❖ Jangkauan terbatas pada fungsi itu sendiri.

Variabel Global

Ciri-ciri variabel global adalah:

- ❖ Identifier yang dideklarasikan di luar fungsi dan ditempatkan di atas semua fungsi dalam suatu program.
- ❖ Jangkauan meliputi seluruh program.
- ❖ Identifier yang dideklarasikan secara global, dapat dideklarasikan kembali (*redeclared*) di subprogram.

Jangkauan Variable Lokal dan Global

```
int x;
fungsi1() {
    ...
}

int y;
fungsi2() {
    int z;
    ...
}

main()
{
    int z;
    int y;
    ...
}
```

scope dari variabel x
z hanya dikenal oleh fungsi2()
scope dari variabel y
z dan y hanya dikenal oleh main
z di main berbeda dgn z di fungsi2()

Passing Parameter

Ada 2 cara melewaskan parameter dalam fungsi:

- ❖ Passing by value
- ❖ Passing by reference

Passing By Value

Passing by Value

- ❖ yang dikirimkan adalah nilai (value) dari variable yang dikirim sebagai argumen.
- ❖ Perubahan terhadap argumen yang terjadi di dalam fungsi **tidak** akan mempengaruhi nilai dari variable tersebut dari fungsi pemanggil (hanya 1 arah yaitu masuk ke fungsi).
- ❖ Pengiriman suatu nilai dapat dilakukan dengan suatu ungkapan, tidak hanya sebuah variable.

Passing By Value

```
1 #include <stdio.h>
2 void fungsi_nilai (int );
3 main()
4 {
5     int a;
6     a = 10;
7     printf("nilai a sebelum fungsi = %d\n", a);
8     fungsi_nilai (a);
9     printf("nilai a setelah fungsi = %d\n", a);
10 }
11
12 void fungsi_nilai (int b)
13 {
14     b = b + 5;
15     printf ("nilai a di fungsi = %d\n",b);
16 }
```

Passing By Reference

- ❖ Yang dikirimkan adalah ***reference*** (pointer/acuan) dari variable yang dikirim sebagai argumen.
- ❖ Perubahan argumen di dalam fungsi, akan mempengaruhi nilai variable pada pemanggil. Karena yang dikirimkan adalah pointer.
- ❖ Jenis data yang dikirim sebagai argumen harus berupa variable, tidak dapat berupa bilangan konstan.

Passing By Reference

```
1 #include <stdio.h>
2 void fungsi_nilai (int *b );
3 main()
4 {
5     int a;
6     a = 10;
7     printf("nilai a sebelum fungsi = %d\n", a);
8     fungsi_nilai (&a);
9     printf("nilai a setelah fungsi = %d\n", a);
10 }
11
12 void fungsi_nilai (int *b)
13 {
14     *b = *b + 5;
15     printf ("nilai a di fungsi = %d\n",*b);
16 }
```

Fungsi Rekursif

- ❖ Suatu proses dari fungsi yang memanggil dirinya sendiri.
- ❖ Harus ada kondisi yang mengakhiri prosesnya. Jika tidak maka proses tidak akan pernah berhenti sampai memori yang digunakan tidak dapat menampung lagi.

Contoh Fungsi Rekursif

❖ Menampilkan deret bilangan genap s/d 20

```
1 #include <stdio.h>
2 #include <conio.h>
3
4 int deretGenap(int i)
5 {
6     if(i<=20)
7     {
8         printf("%d ",i);
9         deretGenap(i+=2);
10    }
11 }
12
13 main()
14 {
15     deretGenap(0);
16     getch();
17 }
```