

## **PRAKTIKUM 5**

### **SUBPROGRAM / FUNGSI**

Program komputer yang dibuat untuk menyelesaikan permasalahan umumnya berukuran besar.

- Cara terbaik untuk menangani program besar adalah menyusunnya dari potongan-potongan program yang berukuran kecil-kecil (disebut modul) merupakan konsep dari pemrograman terstruktur yaitu pemrograman yang menitikberatkan pada pemecahan masalah yang kompleks menjadi masalah yang sederhana.
- Program yang terdiri dari modul/subprogram/prosedur/routine lebih mudah ditangani dibanding dengan program yang terdiri dari banyak sekali baris.
- Modul program dalam C disebut fungsi (function)
- Fungsi adalah blok dari kode yang dirancang untuk melakukan tugas khusus.
- Tujuan pembuatan fungsi :
  - program menjadi terstruktur
  - menghemat kode program karena dapat mengurangi duplikasi kode
  - fungsi dapat dipanggil dari program atau fungsi yang lain
  - mempersingkat/memperpendek panjang program
  - mempermudah cek kesalahan
  -

#### **PENDEKLARASIAN DAN PENDEFINISIAN FUNGSI**

- Fungsi harus dideklarasikan di dalam program pemanggil/program utama, dengan tujuan supaya program pemanggil mengenal nama fungsi tersebut serta cara mengaksesnya.
- Deklarasi fungsi diakhiri ;
- Fungsi didefinisikan dengan diawali tipe data keluaran fungsi (di depan nama fungsi), defaultnya adalah integer.
- Pendefinisian fungsi tidak diakhiri ;
- Aturan pemberian nama fungsi sama dengan aturan penulisan variabel - Blok fungsi diawali dengan { dan diakhiri dengan }

Bentuk :

```
tipe_data nama_fungsi (daftar parameter)
```

Keterangan :

daftar parameter : berisi variabel dan tipe variabel yang berfungsi sebagai masukan untuk fungsi tersebut. Masukan tersebut akan diproses untuk menghasilkan nilai tertentu sesuai dengan tipe data fungsi.

contoh: `int tukar (int x, int y)`

## **VARIABEL GLOBAL, VARIABEL LOKAL, VARIABEL STATIK VARIABEL GLOBAL**

- Variabel yang dideklarasikan di luar blok fungsi dan bersifat dikenali oleh semua bagian program.
- Data-data yang tersimpan dalam sebuah variabel dapat diakses di setiap blok fungsi
- Disarankan untuk tidak digunakan, karena variabel ini dapat men-sharing-kan data dan dapat diubah secara tidak sengaja oleh suatu blok fungsi, sehingga nilainya bisa berubah.

## **VARIABEL LOKAL**

- Variabel yang dideklarasikan dalam suatu blok fungsi tertentu dan hanya dikenal oleh blok fungsi tersebut.
- Variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya.

## **VARIABEL STATIK**

- Variabel statik sering dipakai sebagai variabel lokal.
- Beda variabel lokal dan variabel statik adalah variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya, sedangkan variabel statik akan dihapus dari memori jika program dimatikan. Contoh :

```
int i,j; /* variabel global */
main () {
int k,l; /* variabel lokal */
}
fungsi() {
static int m,n; /* variabel statik */
}
```

## PARAMETER FORMAL & PARAMETER AKTUAL (NYATA)

- Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- Parameter aktual (nyata) adalah parameter yang dapat berupa variabel atau konstanta maupun ungkapan yang dipakai dalam pemanggilan fungsi.

Cara melewatkan/mengirim parameter ke dalam fungsi :

- pengiriman parameter dengan nilai (parameter by value) 🕒 disebut juga parameter masukan
  - Nilai dari parameter aktual akan disalin ke parameter formal.
  - Nilai parameter aktual tidak dapat berubah sekalipun nilai parameter formal berubahubah.

Contoh :

A,B parameter aktual x, y

parameter formal A x

B y

Pada saat pemanggilan suatu fungsi, misal :

A bernilai 20 🕒 x juga bernilai 20

B bernilai 30 🕒 y juga bernilai 30

- pengiriman parameter secara acuan (parameter by reference) 🕒 disebut juga parameter masukan/keluaran
  - perubahan – perubahan yang terjadi pada nilai parameter formal di fungsi akan mempengaruhi nilai parameter aktual.
  - merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi
  - dipakai untuk mengubah isi suatu variabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi

## FUNGSI YANG MENGEMBALIKAN NILAI

- Pada dasarnya semua fungsi mengembalikan nilai, tetapi untuk fungsi yang tidak mengembalikan nilai disebut fungsi bertipe **void**.
- Untuk mengembalikan nilai sebuah fungsi, digunakan kata **return** yang diikuti dengan nilai yang akan dikembalikan.
- Dalam sebuah fungsi return bisa mengembalikan beberapa nilai, tetapi setiap kata return hanya bisa mengembalikan sebuah nilai saja.

### CONTOH PERMASALAHAN 23 : menggunakan fungsi yang tidak mengembalikan nilai

Algoritma fungsi\_garis  
Deklarasi  
function garis () → integer  
Deskripsi  
garis()  
write("NO NIM NAMA NILAI")  
garis()  
  
function garis() → integer  
Deklarasi  
Deskripsi  
write("=====")

Output program :

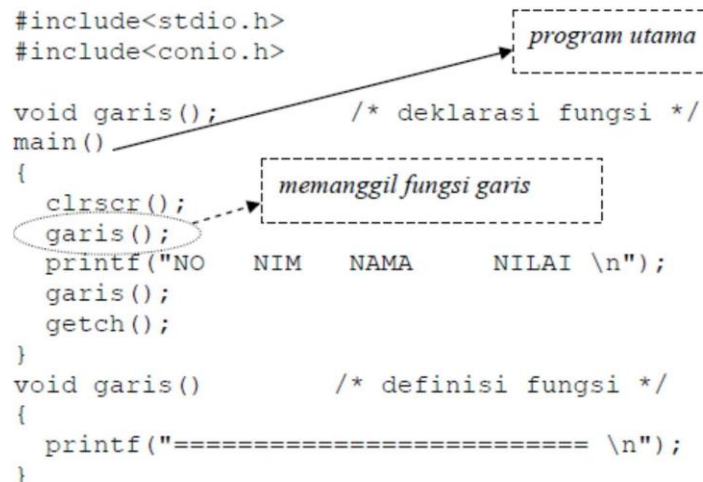
=====  
NO NIM NAMA NILAI  
=====

/\* Program Duapuluh empat \*/  
/\* Program sederhana menggunakan fungsi \*/  
/\* yang tidak mengembalikan nilai \*/

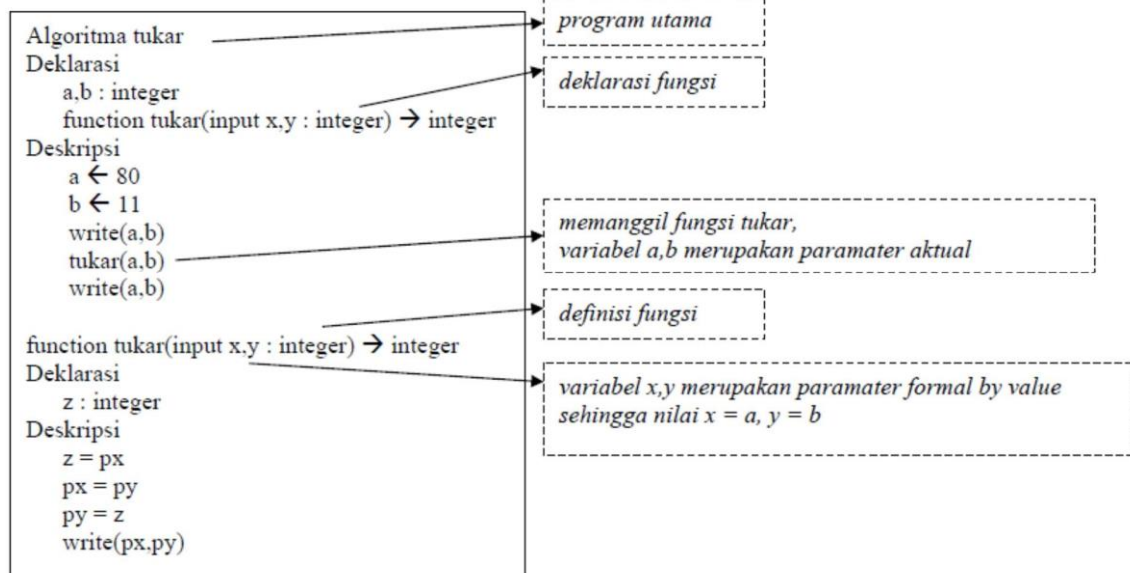
```
#include<stdio.h>
#include<conio.h>

void garis(); /* deklarasi fungsi */
main()
{
    clrscr();
    garis(); /* memanggil fungsi garis */
    printf("NO NIM NAMA NILAI \n");
    garis();
    getch();
}

void garis() /* definisi fungsi */
{
    printf("===== \n");
}
```



**CONTOH PERMASALAHAN 24 : menggunakan fungsi dengan parameter by value**



```

/* Program Duapuluh lima */
/* Program tukar menggunakan fungsi */
/* dengan parameter by value */
#include<stdio.h>
#include<conio.h>
void tukar(int x,int y); /*deklarasi fungsi*/
main() {
    int a,b;
    system("cls");
    a=80;
    b=11;
    printf("Nilai sebelum pemanggilan fungsi \n");
    printf("a = %i b = %i \n",a,b);
    tukar(a,b);
    printf("Nilai setelah pemanggilan fungsi \n");
    printf("a = %i b = %i \n",a,b);
    getch();
}
void tukar(int px,int py) /* definisi fungsi */
{
    int z;
    z=px;
    px=py;
    py=z;
    printf("Nilai diakhir fungsi \n");
    printf("px = %i py = %i \n",px,py);
}

```

Output program :

Nilai sebelum pemanggilan fungsi

a = 80

b = 11

Nilai di akhir fungsi

px = 11

py = 80

Nilai setelah pemanggilan fungsi

a = 80

b = 11

Output program :

Nilai sebelum pemanggilan fungsi

a = 80

b = 11

Nilai di akhir fungsi

px = 11

py = 80

Nilai setelah pemanggilan fungsi

a = 80

b = 11

**CONTOH PERMASALAHAN 2 :** menggunakan fungsi dengan parameter by reference

Algoritma tukar

Deklarasi

a,b : integer

function tukar(input \*px,\*py :  
integer)

Deskripsi

a ← 80

b ← 11

write(a,b)

tukar(a,b)

write(a,b)

function tukar(input \*px,\*py :  
integer)

Deklarasi

z : integer

Deskripsi

z = \*px

\*px = \*py

\*py = z

write(\*px,\*py)

```

/* Program Duapuluh enam */
/* Program tukar menggunakan fungsi */
/* dengan parameter by reference */
#include<stdio.h>
#include<conio.h>
void tukar(int *px,int *py); /*deklarasi fungsi*/
main() {
    int a,b;
    system("cls");
    a=80;
    b=11;
    printf("Nilai sebelum pemanggilan fungsi \n");
    printf("a = %i b = %i \n",a,b);
    tukar(&a,&b);
    printf("Nilai setelah pemanggilan fungsi \n");
    printf("a = %i b = %i \n",a,b);
    getch();
}
void tukar(int *px,int *py) /* definisi fungsi */
{
    int z;
    z=*px;
    *px=*py;
    *py=z;
    printf("Nilai diakhir fungsi \n");
    printf("px = %i py = %i \n",*px,*py);
}

```

Output program :

Nilai sebelum pemanggilan fungsi

a = 80 b = 11

Nilai di akhir fungsi

px = 11 py = 80

Nilai setelah pemanggilan fungsi

a = 11 b = 80