Lab Pemrograman I

Function

Dosen : Eko Budi Setiawan

Definisi Function

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya terpisah dari program yang menggunakannya

Tujuan pemakaian Fungsi

- ❖Program menjadi terstuktur
- ❖Dapat mengurangi pengulangan (duplikasi) kode

Dasar Fungsi

Fungsi standar C yang mengemban tugas khusus contohnya adalah :

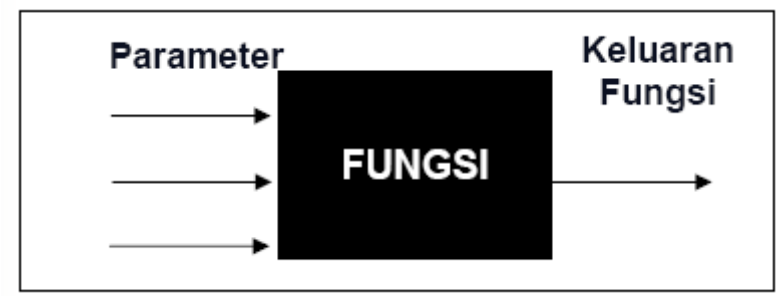
printf (), untuk menampilkan informasi atau data ke layar

scanf (), untuk membaca kode tombol yang diinputkan

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen

Nilai masukan ini akan diolah oleh fungsi

Hasil akhir fungsi berupa sebuah nilai (disebut sebagai return value atau nilai keluaran fungsi)



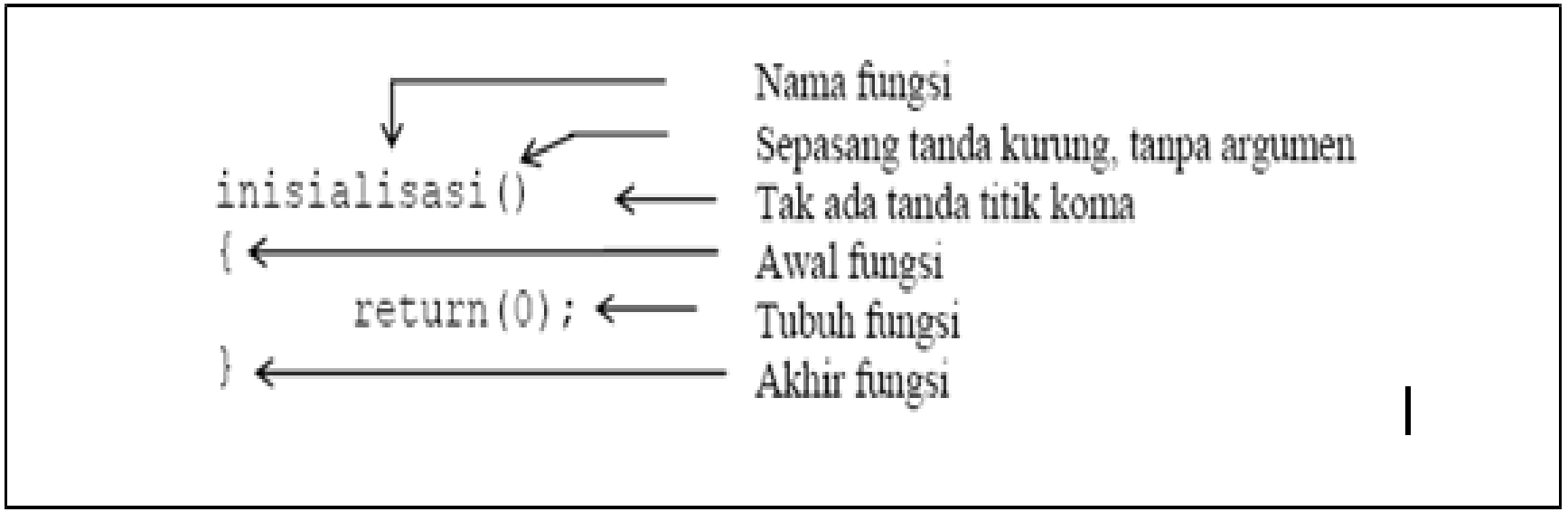
Bentuk umum dari sebuah fungsi adalah sebagai berikut :

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

Keterangan :
tipe-keluaran-fungsi, dapat berupa salah satu tipe data C, misalnya char atau int. Jika penentu tidak disebutkan maka dianggap bertipe int (secara default).

tubuh fungsi berisi deklarasi variabel (jika ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan

Contoh :



return () → merupakan pernyataan untuk memberikan nilai akhir dari fungsi. Pada fungsi inisialisasi() diatas, nilai akhir dari fungsi adalah NOL

Proses Pemanggilan Fungsi

```
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

Hasil Eksekusi :

x = 0

y = 0

Contoh fungsi dengan nilai akhir :

```
//contoh fungsi dengan nilai akhir
#include <stdio.h>
#include <conio.h>

int minimum(int x,int y);

main ()
{
   int hasil;
   int a=20;
   int b=44;
   hasil=minimum(a,b);
   printf("Nilai minimum = %d\n",hasil);
   printf("Nilai minimum = %d\n",minimum(12,15));

}

int minimum(int x,int y)
{
   if(x<y)
      return (x);
   else
      return (y);
}
```

```
(Inactive C:\TCWIN45\BIN\FUNGSI1.EXE)
Nilai minimum = 20
Nilai minimum = 12
```

Fungsi dengan nilai keluaran bukan integer

Fungsi harus dideklarasikan diawal program (sebelum digunakan)
Fungsi didefinisikan dengan diawali penentu tipe fungsi (didepan nama fungsi)

```
float minimum(float x,float y);
.
.
.

float minimum(float x, float y)
{
        if (x < y)
                return(x);
        else
                return(y);
}
```

← Deklarasi fungsi (di awal program)
Diakhiri dengan titik koma

← Definisi fungsi
Tanpa titik koma

Contoh fungsi bertipe bukan integer (misal : float)

```
//contoh fungsi bertipe bukan integer (misal :float)
#include <stdio.h>
#include <conio.h>

float minimum (float x, float y);

main()
{
   float hasil;
   float a=20.5, b=44.5;

   hasil=minimum(a,b);
   printf("Nilai minimum = %g\n",hasil);
   printf("Nilai minimum = %g\n",minimum(3.5,2.5));

}

float minimum(float x, float y)
{
   if (x<y)
      return (x);
   else
      return (y);
}
```

```
(Inactive C:\TCWIN45\BIN\FUNGSI2.EXE)
Nilai minimum = 20.5
Nilai minimum = 2.5
```

Prototipe Fungsi

Prototipe fungsi digunakan untuk menjelaskan kepada kompiler mengenai :

- ❖Tipe keluaran fungsi
- ❖Jumlah parameter
- ❖Tipe dari masing-masing parameter

Bagi kompiler, informasi dalam prototipe akan dipakai untuk memeriksa kebasahan (validitas) parameter dalam pemanggilan fungsi. Salah satu keuntungannya adalah, kompiler akan melakukan konversi seandainya antara tipe parameter dalam fungsi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukkan kesalahan bila jumlah parameter dalam definisi dan saat pemanggilan berbeda

Contoh prototipe fungsi

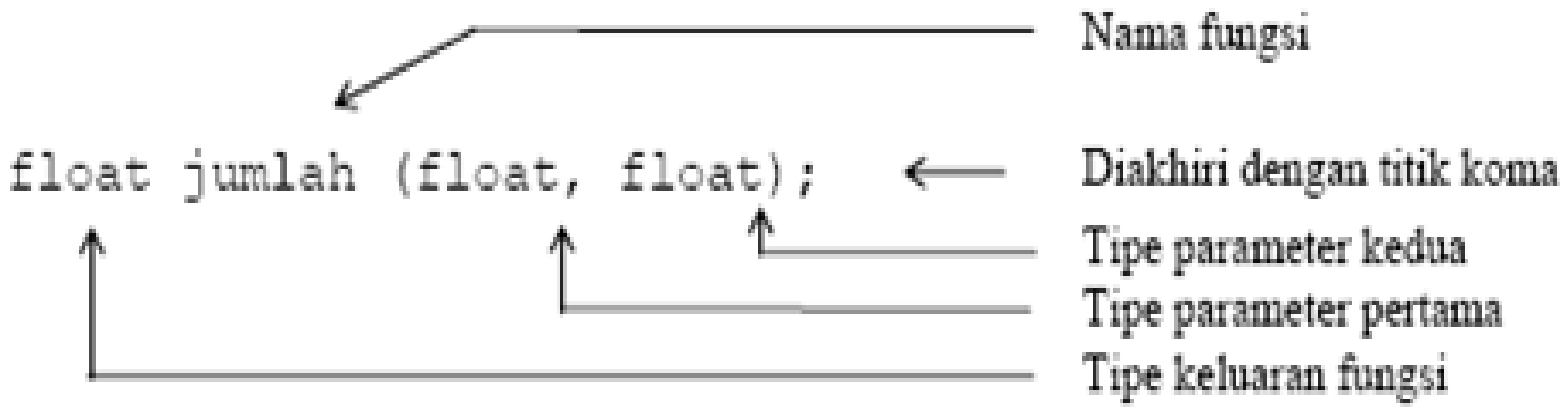
Contoh prototipe fungsi :

```
float jumlah (float x, float y);
```

atau

```
float jumlah (float, float);
```

Penjelasannya adalah sebagai berikut :



Contoh prototipe fungsi

```
//contoh pemakaian prototipe fungsi
#include <stdio.h>

float jumlah (float,float); //prototipe fungsi

main()
{
   int a=6;
   int b=3;
   float c;

   c=jumlah(a,b);
   printf("c=a+b = %g\n",c);
   printf("Hasil penjumlahan = %g\n",jumlah (20.1, 0.9));
}

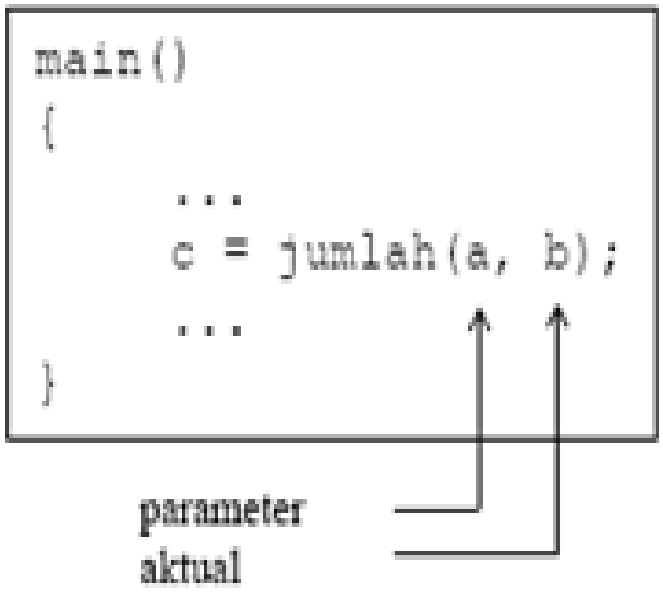
float jumlah (float x,float y)
{
   return(x+y);
}
```

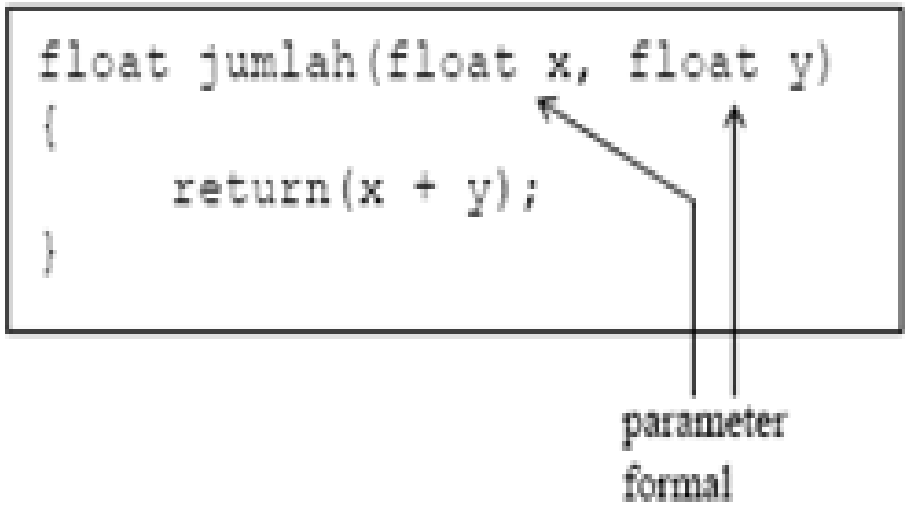
```
(Inactive C:\TCWIN45\BIN\FUNGSI4.EXE)
c=a+b = 9
Hasil penjumlahan = 21
```

Parameter Formal dan Parameter aktual

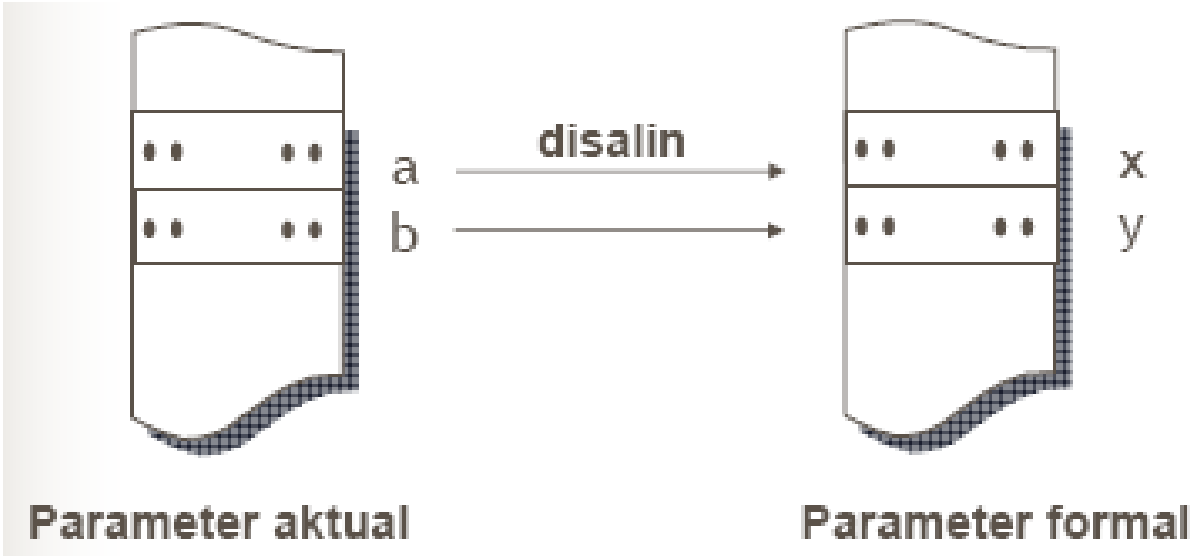
Parameter Formal → Variabel yang ada pada daftar parameter dalam definisi fungsi

Parameter Aktual → Parameter (tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi





Pemanggilan dengan Nilai (Call by Value)



❖Nilai dari parameter aktual akan disalin ke parameter formal

❖Nilai parameter aktual tidak bisa berubah meskipun nilai parameter formalnya berubah

Contoh pemanggilan dengan Nilai (Call by Value)

```c
/* File program : tukar1.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi un

#include <stdio.h>
void tukar (int, int);
main()
{
int a = 88, b = 77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
tukar(a,b);
printf("\nNilai setelah pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
}

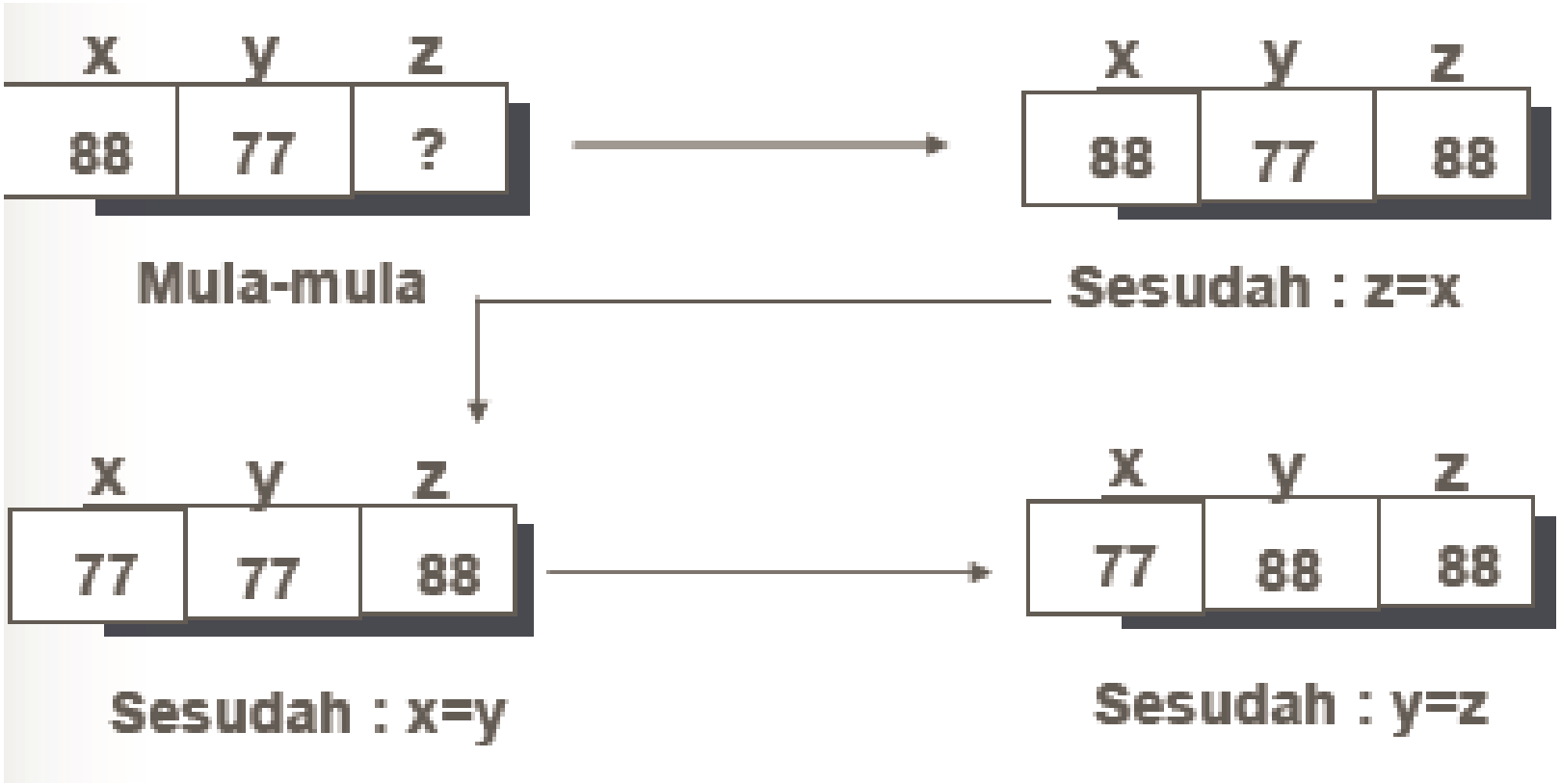
void tukar(int x, int y)
{
int z;
z = x;
x = y;
y = z;
printf("\nNilai di akhir fungsi tukar()\n");
printf("x = %d y = %d\n", x, y);
}
```

```
Nilai sebelum pemanggilan fungsi
a = 88 b = 77

Nilai di akhir fungsi tukar()
x = 77 y = 88

Nilai setelah pemanggilan fungsi
a = 88 b = 77
```

Proses penukaran nilai



Pemanggilan dengan Referensi

❖Upaya untuk melewatkan alamat dari suatu variabel kedalam fungsi

❖Untuk mengubah isi variabel diluar fungsi dimana pelaksanaan pengubahan dilakukan didalam fungsi

Modifikasi pada parameter formal :

```
tukar(int *px, int *py)
{
  int z;

  z= *px;
  *px=*py;
  *py=z;
  ……
```

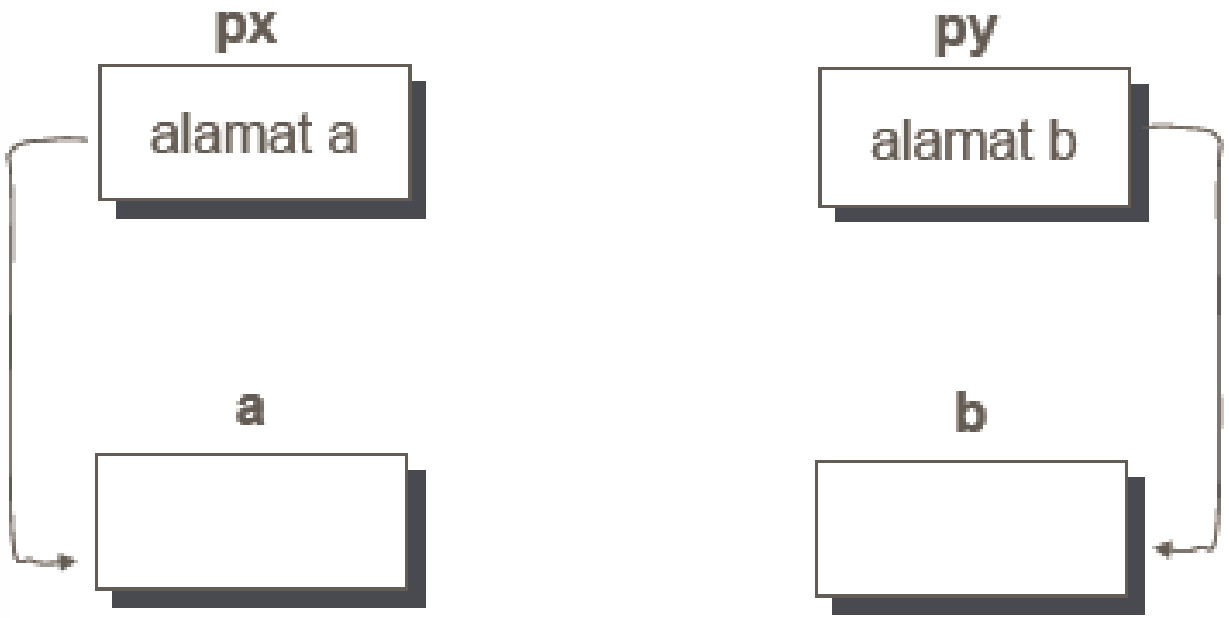
Variabel pointer → menunjuk ke variabel lain

- Modifikasi untuk parameter aktual :

```
tukar(&a,&b);
```

variabel diawali dengan & menyatakan alamat dari a dan alamat dari b

Pemanggilan dengan Referensi



Variabel pointer px menunjuk ke variabel a dan variabel pointer py menunjuk ke variabel b

Contoh pemanggilan dengan Referensi

```
/* File program : tukar2.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi

#include <stdio.h>
void tukar (int *px, int *py); //prototype fungsi
main()
{
int a= 88,b = 77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
tukar(&a,&b); //alamat a dan alamat b
printf("\nNilai setelah pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
}

void tukar(int *px, int *py)
{
int z;
z = *px;
*px = *py;
*py = z;
printf("\nNilai di akhir fungsi tukar()\n");
printf("x = %d y = %d\n", *px, *py);
}
```

```
(Inactive E:\JAWABA~1\NONAME01.EXE)
Nilai sebelum pemanggilan fungsi
a = 88 b = 77

Nilai di akhir fungsi tukar()
x = 77 y = 88

Nilai setelah pemanggilan fungsi
a = 77 b = 88
```

Contoh Lain

```
/*contoh program menggunakan parameter formal dan aktual
  menentukan apakah bilangan yang diinputkan ganjil atau genap */

#include <stdio.h>
int ganjil(int);

main()
{
    int a, hasil;
    printf("Masukan sembarang bilangan bulat : ");
    scanf("%d", &a);
    hasil=ganjil(a);

    if(hasil==0)
       printf("Bilangan tersebut genap\n");
    else if(hasil==1)
       printf("Bilangan tersebut ganjil\n");
}

int ganjil (int x)
{
    if(x%2 == 0)
       return (0);
    else
       return (1);
}
```

Contoh Lain

```
#include <stdio.h>
void hitung_rata_rata();

void main()
{
    printf("Menghitung rata-rata N buah bilangan Bulat \n");
    printf("\nMemanggil dan Melaksanakan Fungsi\n");
    printf("\=======================================\n");
    hitung_rata_rata();
    printf("\nSukses");
}

void hitung_rata_rata()
    {

    int bil, data, i,jumlah;
    float rata;
    jumlah=0;
    rata=0;
    printf("Data yang diinputkan =    ");scanf("%i",&bil);
    for  (i=1;i<=bil;i++)
        {
        printf  ("Data ke %d   =  ",i);scanf("%d",&data);
        jumlah=jumlah+data;
        }
    rata=float(jumlah)/bil;
    printf("\nHasil Akhir    :");

    printf("\nRata-ratanya adalah : %g",rata);
    }
```

Contoh Lain

```
#include <stdio.h>
#include <conio.h>
int f(int x);


main()
{
   int a,b,c;
   a=f(2);
   b=f(3)+f(1);
   c=f(f(2));
   printf("f(2)       = %d\n",a);
   printf("f(3)+f(1)  = %d\n",b);
   printf("f(f(2))    = %d\n",c);
   printf("f(5)       = %d\n",f(5));
   getch();
   return 0;
   }

int f(int x)
{
   return x*x+3*x+1;
}
```

```
C:\TCWIN45\BIN\FUNGSIFX.EXE
f(2)       = 11
f(3)+f(1)  = 24
f(f(2))    = 155
f(5)       = 41
```