

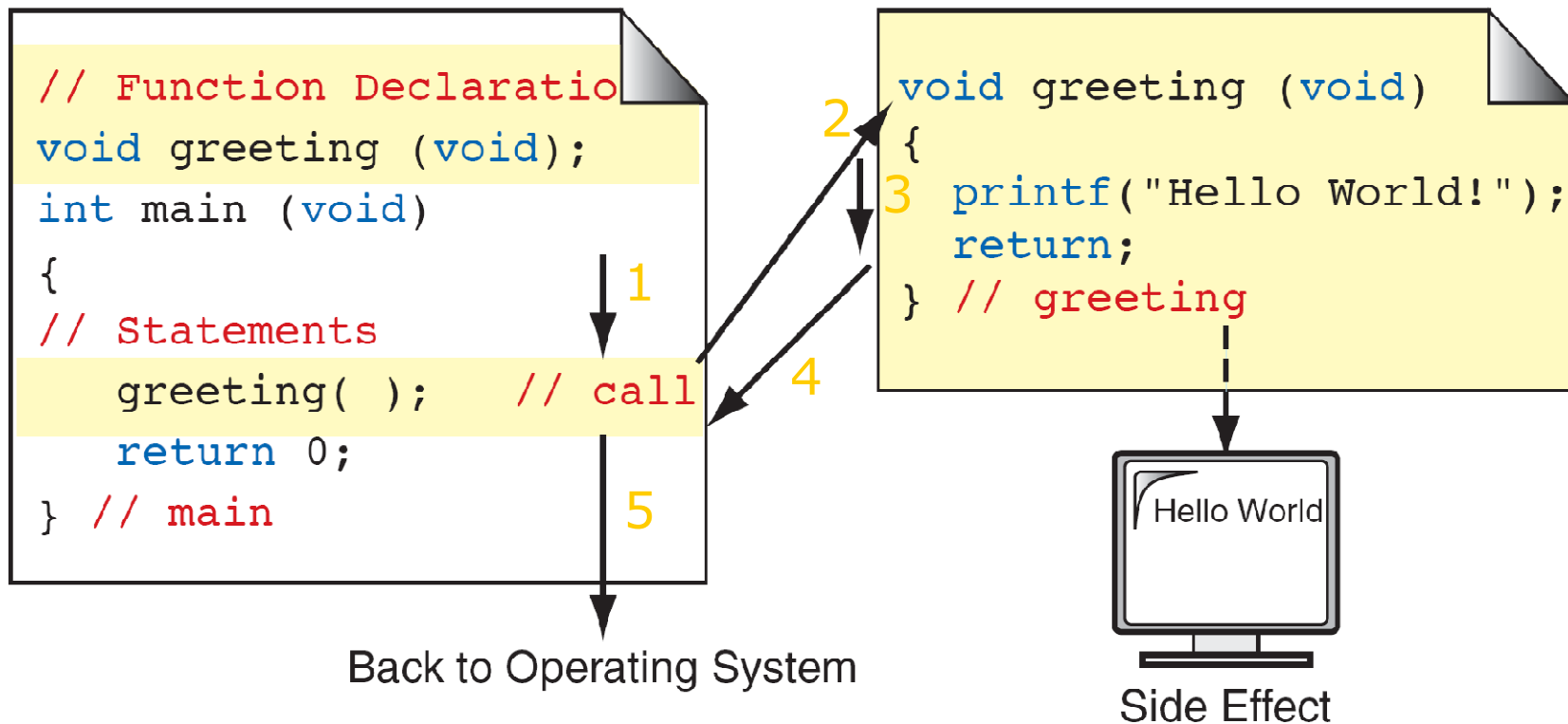
Algoritma & Pemrograman #8

by antonius rachmat c, s.kom, m.cs

Review Fungsi Minggu lalu

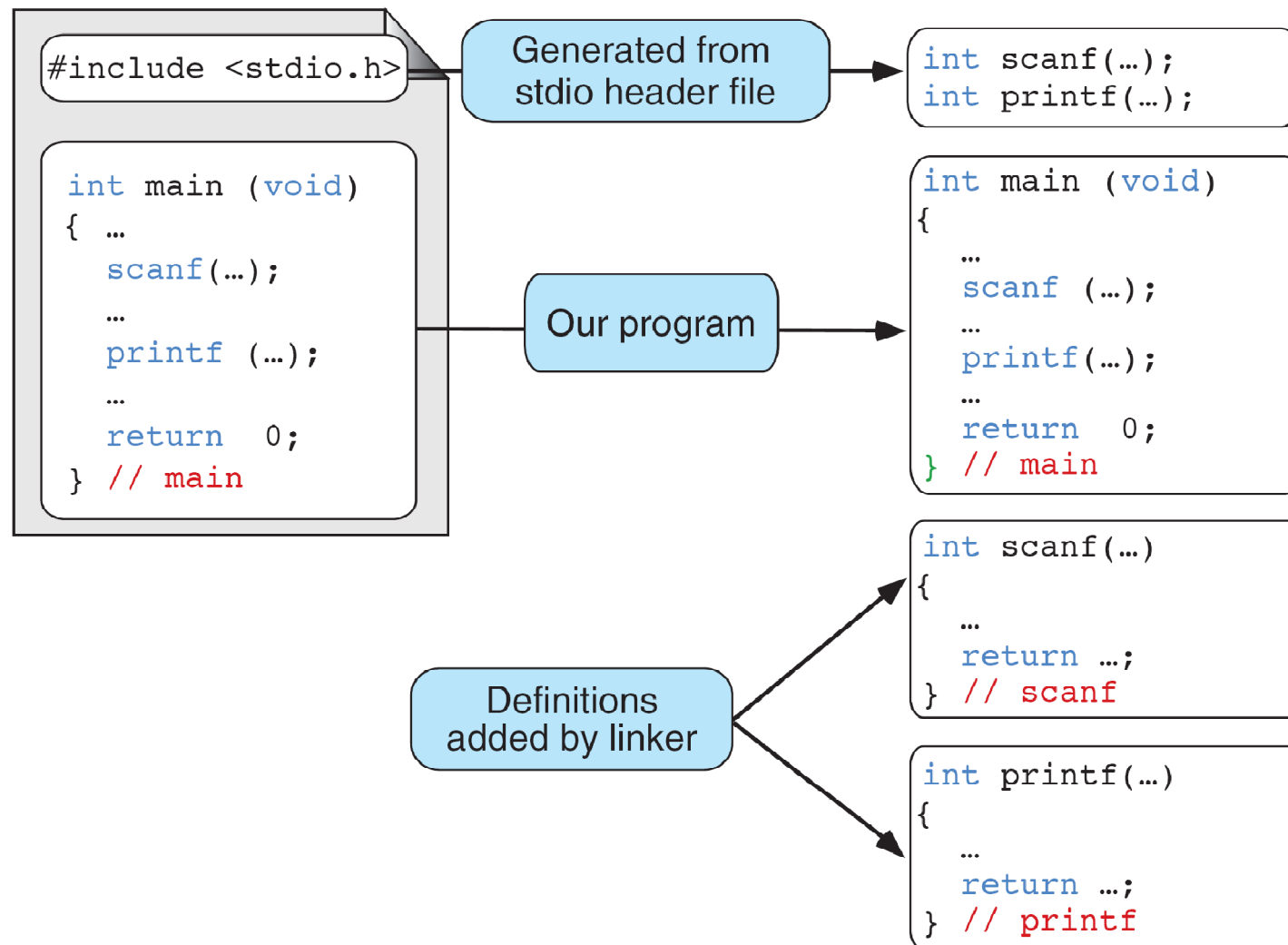
- Deklarasi dan Definisi fungsi
- Standard Library Function
- Void dan Non-void dan Parameternya

REVIEW



Declaring, Calling, and Defining Functions

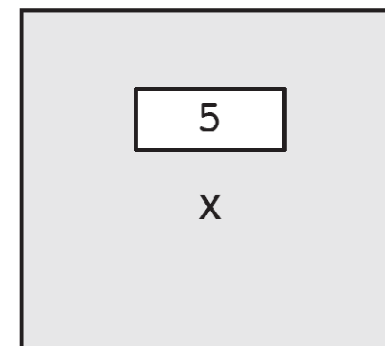
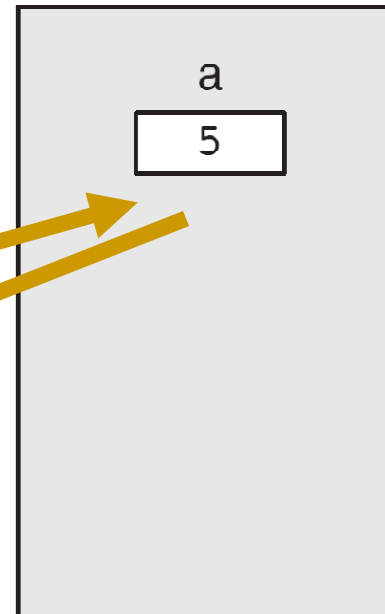
REVIEW



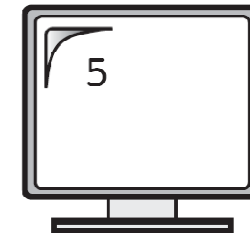
Library Functions and the Linker

```
// Function Declaration
void printOne (int x);
int main (void)
{
    // Local Declarations
    int a = 5;
    // Statements
    printOne (a); // call
    return 0;
} // main
```

```
void printOne (int x)
{
    printf("%d\n", x);
    return;
} // printOne
```



REVIEW



Side Effect

void Function with Parameters

REVIEW

```
// Function Declaration
int getQuantity (void);

int main (void)
{
    // Local Declarations
    int amt;

    // Statements
    amt = getQuantity ( );
    ...
    return 0;
} // main
```

```
int getQuantity (void)
{
    // Local Declarations
    int qty;

    // Statements
    printf("Enter Quantity");
    scanf ("%d", &qty);
    return qty;
} // getQuantity
```

Non-void Function without Parameters

REVIEW

```
// Function Declaration
int sqr (int x);
int main (void)
{
    // Local Declarations
    int a;
    int b;
    // Statements
    scanf("%d", &a);
    b = sqr (a);
    printf("%d squared: %d\n", a, b);
    return 0;
} // main
```

```
int sqr (int x)
{
    // Statements
    return (x * x);
} // sqr
```

Returned
stored here

a

b

x

Calling a Function That Returns a Value

Scope Variable

- ❑ Sebuah variabel di dalam sebuah fungsi memiliki jangkauan tertentu.
- ❑ Skop variabel terdiri dari:
 - Variabel lokal
 - Variabel global
 - Variabel statis

Variabel lokal

- ❑ Variabel yang hanya dikenal di daerah yang lokal saja, misalnya di dalam sebuah fungsi/prosedur **tertentu saja** dan tidak dikenal di daerah lainnya.
- ❑ Harus dideklarasikan di dalam **blok** yang bersangkutan
- ❑ Variabel lokal diciptakan ketika fungsi dipanggil dan akan dihapus dari memori bila eksekusi terhadap fungsi selesai.
- ❑ Dalam C, **tidak ada inisialisasi otomatis**

Two values received
from calling function

```
double average (int x,int y)
{
    double sum;
    sum = x + y;
    return (sum / 2);
} // average
```

One value returned
to calling function

parameter variables

x
y

local variable

sum

Function Local Variables

Variabel lokal (2)

Contoh-05a.

```
#include<stdio.h>
void CETAK();
void main()
{
    CETAK();
}
```

DEKLARASI fungsi

tipe : void

karena tak ada nilai
yang dikirim ke fungsi
utama main()

```
void CETAK()
{
    int A,B,T;
    A=5; B=2;
    T = A+B;
    printf("%d", T);
}
```

Fungsi CETAK ini, merupakan suatu subprogram tersendiri yang dapat membuat variabel sendiri. Semua variabel yang dibuat sendiri disini, variabel tersebut disebut bersifat LOKAL, yang artinya hanya berlaku dalam fungsi ini saja. Tidak berlaku di fungsi utama main(), atau dalam fungsi yang lainnya.

Tercetak : 7

Variabel lokal (3)

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

```
void CETAK();
```

```
void main(){
```

```
    int A,B,T;
```

```
    A=5; B=2;
```

```
    T=A+B;
```

```
    CETAK();
```

```
}
```

```
void CETAK(){
```

```
    printf("%d",T);    //terjadi error, T tidak dikenal
```

```
}
```

Variabel Lokal (4)

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

int TAMBAH(int A,int B);

int main(){
    int hasil;
    hasil = TAMBAH(2,3);
    printf("Hasil = %d",hasil);
    getch();
}

int TAMBAH(int A,int B){
    int C;
    C = A + B;
    {
        float C;
        C = 100;
    }
    return(C);
}
```

Hasilnya: 5

Mengapa tidak bernilai 100? Hal ini karena variabel C di deklarasikan di dalam blok sendiri sehingga dianggap berbeda dengan variabel C yang berisi nilai 5

Variabel Global

- ❑ Variabel yang dikenal diseluruh daerah di dalam program, di dalam dan luar fungsi.
- ❑ Dideklarasikan di luar suatu blok statemen atau di luar fungsi-fungsi yang menggunakannya.
- ❑ Variabel global dapat dideklarasikan kembali di dalam fungsi. (**redeclare**)
 - dan yang digunakan adalah variabel lokalnya
- ❑ Kerugian penggunaan variabel global:
 - Memboroskan memori komputer karena komputer masih menyimpan nilainya walaupun sudah tidak diperlukan lagi.
 - Mudah terjadi kesalahan program karena satu perubahan dapat menyebabkan perubahan menyeluruh pada program.
 - Pembuatan fungsi lebih sulit, karena harus diketahui variable global apa saja yang digunakan.
 - Pendeteksian kesalahan program lebih sulit dilakukan.

Contoh

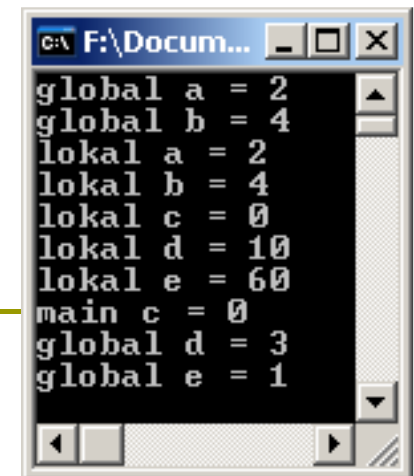
```
#include <stdio.h>
#include <conio.h>

int d=3,e=1;

void coba_lokal(int a,int b){
    int c = 0;
    int d = 10;
    int e;
    e = (a+b) * (c+d);
    printf("lokal a = %d\n",a);
    printf("lokal b = %d\n",b);
    printf("lokal c = %d\n",c);
    printf("lokal d = %d\n",d);
    printf("lokal e = %d\n",e);
}

int main(){
    int a=2;
    int b;
    b = 4;
    int c=0;

    printf("main a = %d\n",a);
    printf("main b = %d\n",b);
    coba_lokal(a,b);
    printf("main c = %d\n",c);
    printf("global d = %d\n",d);
    printf("global e = %d\n",e);
    getch();
}
```



```
F:\Docum...
global a = 2
global b = 4
lokal a = 2
lokal b = 4
lokal c = 0
lokal d = 10
lokal e = 60
main c = 0
global d = 3
global e = 1
```

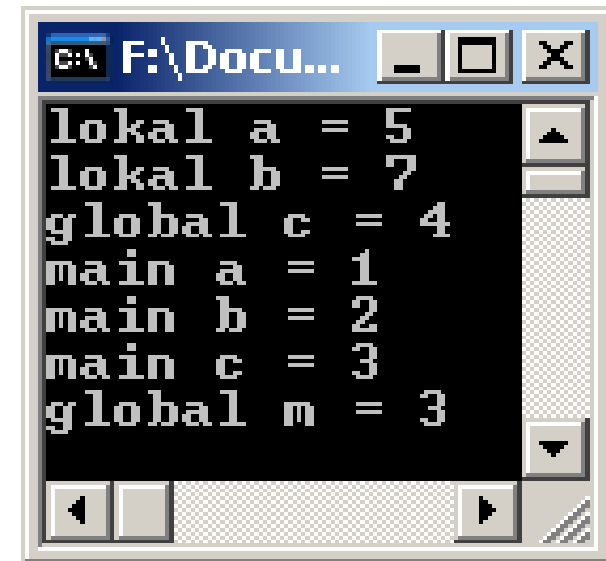
Contoh

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

int c = 4;
int m = 3;

void lokal() {
    int a = 5;
    int b = a + 2;
    printf("lokal a = %d\n",a);
    printf("lokal b = %d\n",b);
    //karena tidak ada c, maka ambil global
    printf("global c = %d\n",c);
}

int main() {
    int a = 1;
    int b = 2;
    int c = 3;
    lokal();
    printf("main a = %d\n",a);
    printf("main b = %d\n",b);
    //walaupun global c ada tapi c yang digunakan yang
    di main
```



```
lokal a = 5
lokal b = 7
global c = 4
main a = 1
main b = 2
main c = 3
global m = 3
```


Variabel statis

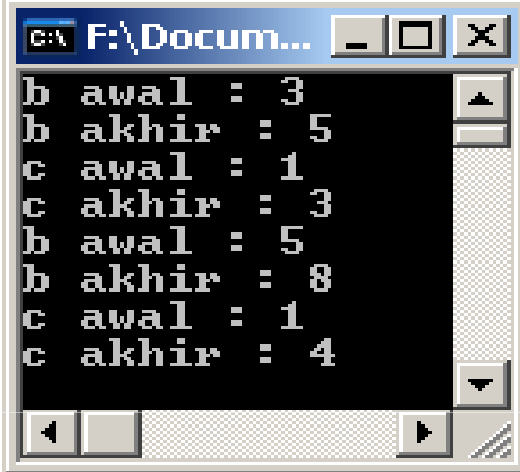
```
#include <stdio.h>
#include <conio.h>

void coba_static(int a) {
    static int b=3;
    int c=1;

    printf("b awal : %d\n",b) ;
    b += a;
    printf("b akhir : %d\n",b) ;

    printf("c awal : %d\n",c) ;
    c += a;
    printf("c akhir : %d\n",c) ;
}

int main() {
    int a=2;
    coba_static(a) ;
    a=3;
    coba_static(a) ;
    getch() ;
}
```



```
F:\Docum...
b awal : 3
b akhir : 5
c awal : 1
c akhir : 3
b awal : 5
b akhir : 8
c awal : 1
c akhir : 4
```

Variabel statis (2)

- ❑ Jika variabel statis bersifat **lokal**, maka hanya dikenal dalam fungsi tersebut saja.
- ❑ Jika variabel statis bersifat **global**, maka dikenal di seluruh program
- ❑ Inisialisasi hanya dilakukan **sekali**, yaitu pada saat fungsi dipanggil **pertama kali**.
- ❑ Adalah variabel yang memiliki **nilai tetap**, artinya nilai dari variabel tersebut akan tetap diingat oleh program, sehingga dapat digunakan untuk menyimpan state nilai pada saat pemanggilan fungsi berikutnya.
- ❑ Nilai variabel statis akan memiliki nilai sesuai dengan **nilai terakhirnya**.

```
/* This is a sample to demonstrate scope. The techniques
   used in this program should never be used in practice.
*/
```

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

```
int fun (int a, int b);
```

Global area

```
int main (void)
```

```
{
```

```
    int    a;
```

```
    int    b;
```

```
    float  y;
```

```
    ...
```

```
    { // Beginning of nested block
```

```
      float a = y / 2;
```

```
      float y;
```

```
      float z;
```

```
      ...
```

```
      z = a * b;
```

```
      ...
```

```
    } // End of nested block
```

```
    ...
```

```
} // End of main
```

main's area

Nested block
area

```
int fun (int i, int j)
```

```
{
```

```
    int a;
```

```
    int y;
```

```
    ...
```

```
} // fun
```

fun's area

Note

Variables are in scope
from declaration until
the
end of their block.

Scope for Global and Block Areas

```
int A,E;
```

```
void main()
```

```
{
```

```
    /* blok main */
```

```
    float C;
```

```
    {
```

```
        /* blok statemen 1 */
```

```
        int D;
```

```
        ...
```

```
    }
```

```
}
```

```
// variabel E bersifat global untuk blok dibawahnya
```

```
double E;
```

```
double Fungsi(void){
```

```
    double F;
```

```
    ...
```

```
}
```

```
int Fungsi2(void){
```

```
    char G;
```

```
    /* blok statement 2 */
```

```
    {
```

```
        int H;
```

```
        ...
```

```
    }
```

```
    /* blok statement 3 */
```

```
    {
```

```
        int I;
```

```
        ...
```

```
    }
```

```
}
```

Program	x	f1	y1	y	x	b	y	z
#include <stdio.h>								
int x;								
float func_1(int f1){								
int y1;								
....								
}								
char y;								
int func_2(){								
int x;								
...								
}								
int func_3(){								
...								
{								
int b;								
....								
}								
}								
void main(){								
char y;								
int z;								
....								
}								

Argumen Fungsi

- ❑ Sebuah fungsi bisa memiliki argumen-argumen yang bersifat opsional.
- ❑ Argumen-argumen tersebut berfungsi sebagai parameter inputan yang berupa variabel-variabel bagi fungsi tersebut (bersifat **lokal**).
- ❑ Argumen harus bertipe data tertentu.
- ❑ Terdapat 2 jenis parameter:
 - **Parameter formal**: parameter yang ditulis pada deklarasi fungsi.
 - **Parameter aktual**: parameter yang diinputkan dalam program pemanggil fungsi tersebut.
 - ❑ Dapat berupa variabel atau langsung berupa nilai tertentu sesuai dengan tipe data yang dideklarasikan untuk masing-masing parameter fungsi

Parameter formal dan aktual

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

```
int JUMLAH(int X, int Y);
```

X, Y disebut parameter formal

```
void main(){
```

```
    int A,B,T;
```

Variabel A,B,C lokal dalam main

```
    A=5; B=2;
```

```
    T = JUMLAH(A,B);
```

A dan B disebut parameter aktual

```
    printf("%d",T);
```

```
}
```

```
int JUMLAH(int X, int Y){
```

X, Y disebut parameter formal

```
    int H;
```

Variabel X,Y lokal dalam JUMLAH

```
    H = X + Y;
```

```
    return(H);
```

```
}
```

Pengiriman parameter

- Komunikasi antar fungsi dilakukan dengan saling bertukar data
- Hasil dari suatu fungsi dapat diperoleh dari
 - hasil baliknya (**return**),
 - dengan menggunakan variabel **Global**,
 - hasil proses dari suatu fungsi dapat diperoleh, karena variabel yang dipakai dalam fungsi bersifat global.
 - hasil dapat juga diperoleh dari parameter aktual yang dikirimkan ke parameter formal, karena parameter formal seolah-olah akan mengirimkan kembali nilai hasil proses dalam fungsi. (disebut **by reference**)

Pengiriman Parameter

- Pengiriman secara nilai (***by value***)
 - Secara **default** pengiriman parameter di dalam C adalah by value
 - Pengubahan nilai di dalam fungsi tidak bisa mengubah nilai di luar fungsi
- Pengiriman secara acuan (***by reference***)
 - Berhubungan dengan Pointer
 - Dibahas detail di Struktur data

By Value

- Yang dikirimkan ke fungsi adalah **nilainya**, *bukan alamat memori* letak dari datanya
- Fungsi yang menerima kiriman nilai ini akan menyimpannya di **alamat terpisah** dari nilai aslinya yang digunakan oleh program yang memanggil fungsi tersebut
- Karena itulah perubahan nilai di dalam fungsi **tidak akan berpengaruh** pada nilai asli di program yang memanggil fungsi walaupun keduanya menggunakan nama variabel yang sama
- Pengiriman by value adalah pengiriman **searah**, dari program pemanggil fungsi ke fungsi yang dipanggilnya
- Pengiriman by value dapat dilakukan untuk suatu **statement**, *tidak hanya* untuk suatu variabel, value, array atau konstanta saja.

By Value (2)

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

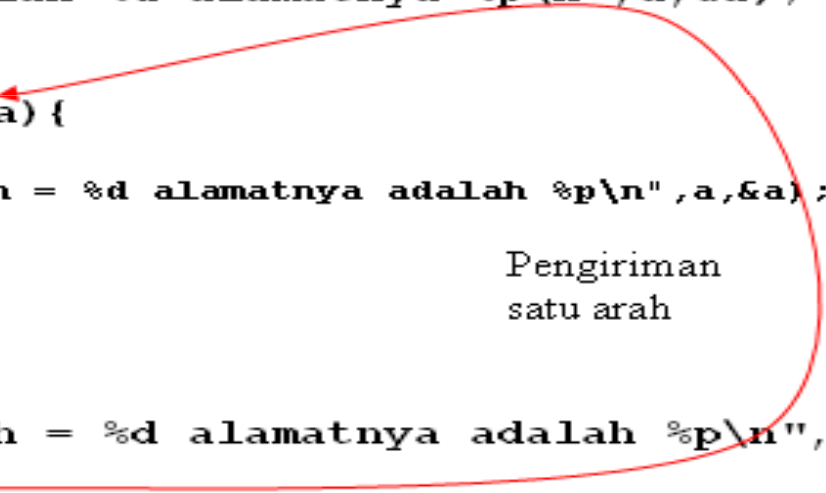
int a=4;

void getAGlobal(){
    printf("A Global adalah %d alamatnya %p\n", a, &a);
}

void fungsi_by_value(int a){
    a = a * 3;
    printf("A by value adalah = %d alamatnya adalah %p\n", a, &a);
}

void main(){
    int a = 5;
    getAGlobal();
    printf("A main adalah = %d alamatnya adalah %p\n", a, &a);
    fungsi_by_value(a);
    printf("A main setelah fungsi dipanggil adalah = %d
alamatnya adalah %p\n", a, &a);
    getch();
}
```

Pengiriman
satu arah



By Value (3)

□ Hasil:

```
A Global adalah 4 alamatnya 00402000  
A main adalah = 5 alamatnya adalah 0022FF44  
A by value adalah = 15 alamatnya adalah 0022FF20  
A main setelah fungsi dipanggil adalah = 5 alamatnya adalah 0022FF44
```

□ Memory:

a di *global*
nilai 4
alamat 00402000


a di *main*
nilai 5
alamat 0022FF44

a di
fungsi_by_value
nilai 15
alamat
0022FF20

a di *main after*
function
nilai 5
alamat
0022FF44

By Value (4)

```
...  
void fungsi_by_value(int a){  
    a = a * 3;  
    printf("A by value adalah = %d alamatnya adalah %p\n",a,&a);  
}  
  
void main(){  
    int a = 5;  
    getAGlobal();  
    printf("A main adalah = %d alamatnya adalah %p\n",a);  
  
    fungsi_by_value(5*a+1);  
    getch();  
}  
...
```



Statement

Contoh by value

```
Lokal A = 7.000000, alamat A = 0022FF10
Lokal B = 8.333333
Lokal C = a
Main A = 25.000000, alamat A = 0022FF44
Main A/3 = 8.333333
Main C = a
```

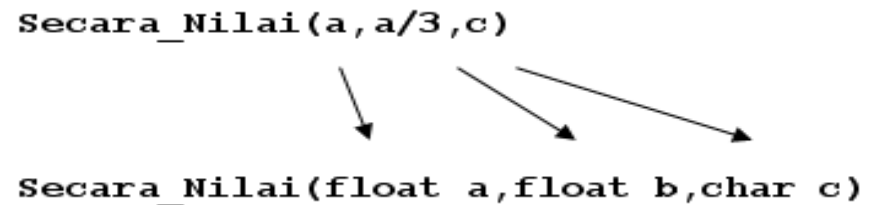
```
#include <stdio.h>
#include <conio.h>

void Secara_Nilai(float a,float b,char c){
    float *Alamat_A;
    Alamat_A = &a;
    a = 7;
    printf("Lokal A = %f, alamat A = %p\n",a,Alamat_A);
    printf("Lokal B = %f\n",b);
    printf("Lokal C = %c\n",c);
}

void main(){
    float a=25,*Alamat_A;
    char c = 'a';
    Alamat_A = &a;
    Secara_Nilai(a,a/3,c);
    printf("Main A = %f, alamat A = %p\n",a,Alamat_A);
    printf("Main A/3 = %f\n", (a/3));
    printf("Main C = %c\n",c);
    getch();
}
```

Penjelasan

- ❑ Parameter aktual yang dikirimkan adalah datanya, yaitu **Secara_Nilai(a,a/3,c)**
- ❑ Alamat nilai a pada main dan a pada fungsi Secara_Nilai berbeda, yaitu **2447:2456** dan **2447:2466**
- ❑ Perubahan nilai a dalam fungsi Secara_Nilai menjadi 7 tidak mengubah nilai a pada main yaitu tetap **25**
- ❑ Pengirimannya satu arah

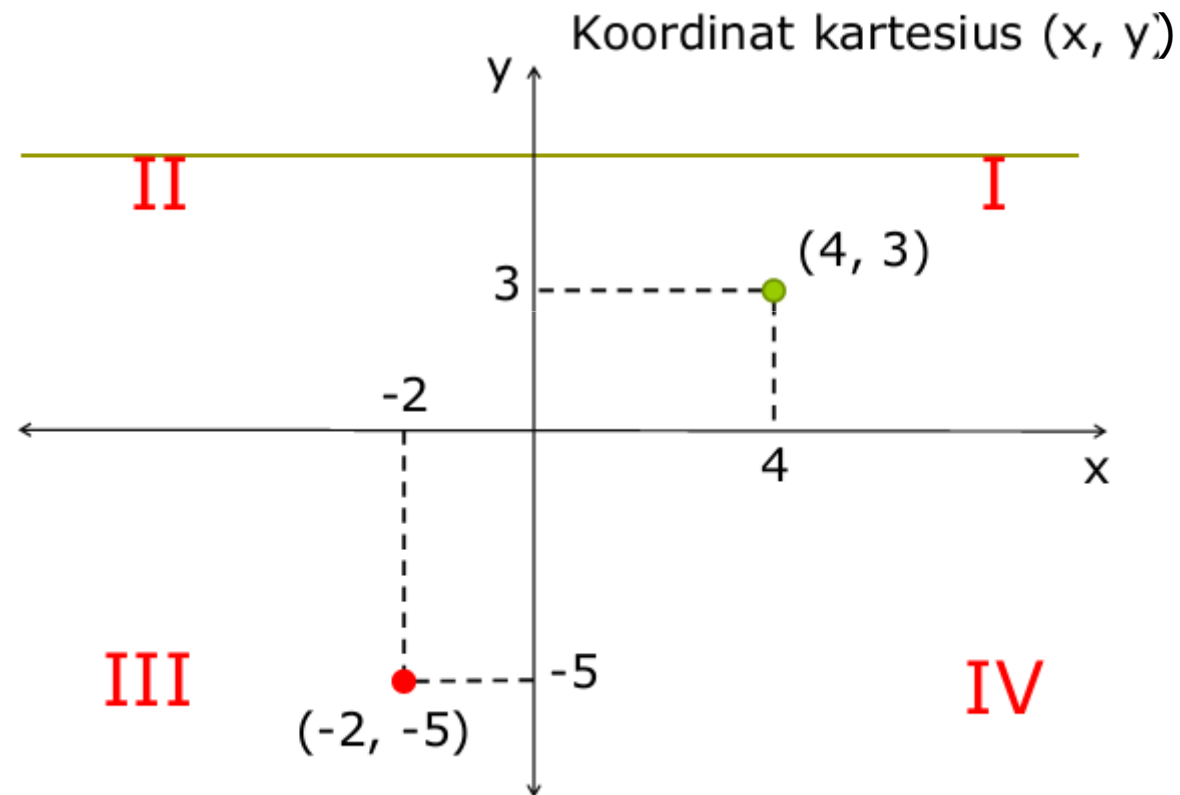


- ❑ Pengiriman parameter dapat berupa ungkapan (statement) yaitu **a/3**

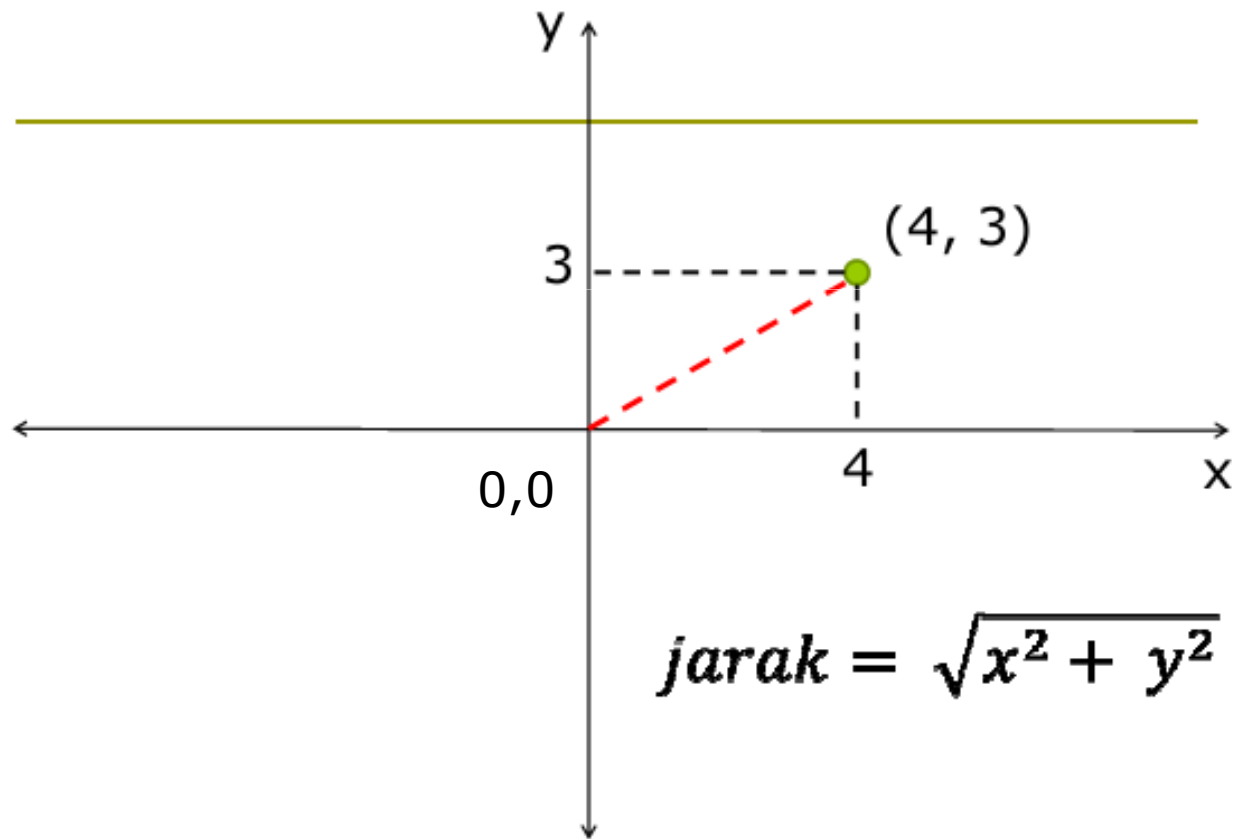
Latihan

- Buatlah fungsi untuk:
 - Menentukan kuadran suatu titik
 - Menentukan jarak suatu titik dengan titik pusat $(0,0)$
 - Pencerminkan sebuah titik terhadap sumbu x , sumbu y
 - Pencerminkan terhadap garis $Y=X \rightarrow R(b,a)$
 - Pencerminkan terhadap garis $X=h \rightarrow R(2h-a,b)$
 - Pencerminkan terhadap garis $Y=k \rightarrow R(a,2k-b)$

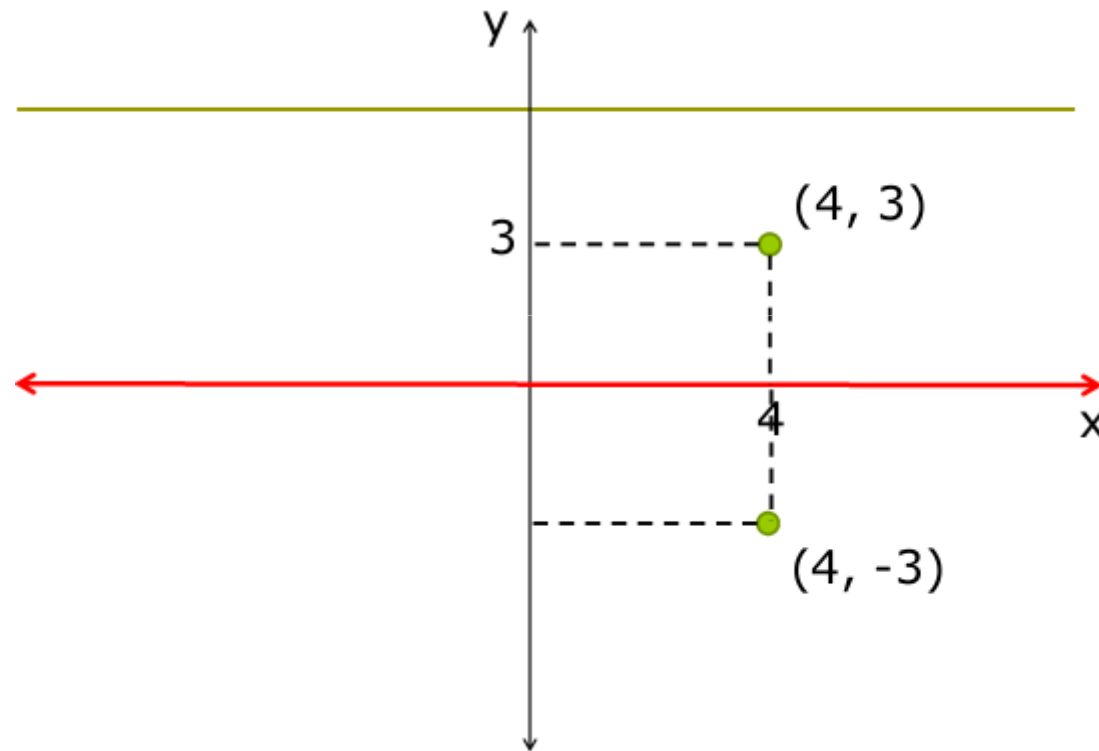
Kuadran



Jarak titik dengan 0,0



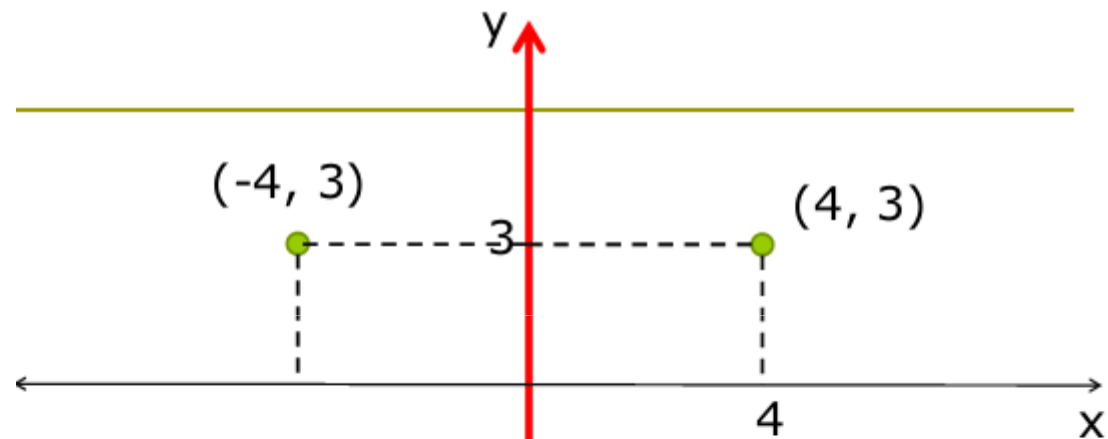
Perncerminan terhadap sumbu X



Pencerminan terhadap sumbu x

$$R(a, -b)$$

Pencerminan terhadap sumbu Y



Pencerminan terhadap sumbu y

$$R(-a, b)$$

NEXT

- ▣ Array (1 Dimensi)