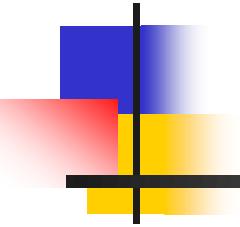


# Fungsi



---

## **DASAR PEMROGRAMAN**



# TUJUAN

---

- Menjelaskan pengertian Fungsi
- Membuat Fungsi
- Memecah program dalam beberapa fungsi.
- Mengerti parameter dalam Fungsi
- Mengerti variabel dalam Fungsi



# Fungsi (function)

---

- Adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya.
- Elemen utama dari program bahasa C berupa fungsi-fungsi dari kumpulan fungsi pustaka (standar) dan fungsi yang dibuat sendiri oleh pemrogram.
- Contoh fungsi standart:
  - printf
  - scanf



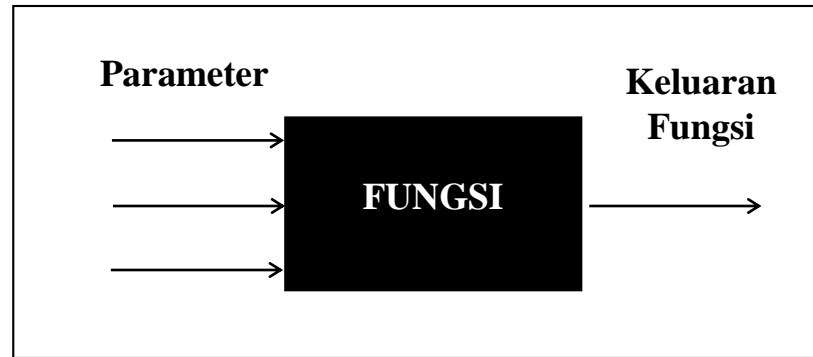
# Tujuan pembuatan Fungsi

---

- 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.

# Dasar Fungsi

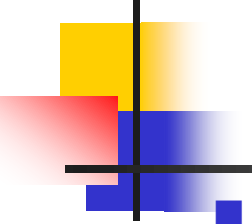
Fungsi sering digambarkan sebagai "kotak gelap"



Bentuk Umum

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

# 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.

# Definisi Fungsi

```
int inisialisasi()
```

```
{
```

```
    return(0);
```

```
}
```

```
inisialisasi()
```

```
{
```

```
    return(0);
```

```
}
```

Diagram illustrating the components of a function definition:

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

# Memberikan Nilai Keluaran 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 (pakai void)

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

- Fungsi yang mempunyai output.

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



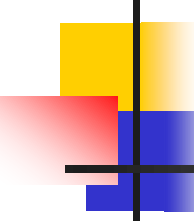
# Contoh Fungsi

---

```
#include<stdio.h>
void info_program();    //Prototype Fungsi
main()
{
    printf("\nInfo Pembuat Program \n");
    info_program();

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

# Contoh Fungsi



```
#include <stdio.h>
int kuadrat (int b); //Prototipe Fungsi
```

---

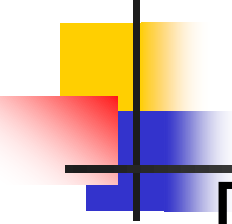
```
main()
{
    int pangkat;
    printf("Kuadrat 2 adl %d \n", kuadrat(2));
    printf("Kuadrat 3 adl %d \n", kuadrat(3));
    pangkat = kuadrat (5);
    printf("Kuadrat 5 adl %d \n", pangkat);
}
```

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

}      

```
int z;
z = b*b;
return(z);
```

# Prototype Fungsi



---

Digunakan untuk menjelaskan kepada kompiler mengenai :

- tipe keluaran fungsi
- jumlah parameter
- tipe dari masing-masing parameter.

# Penggunaan Prototype Fungsi

## Fungsi yang mempunyai output

**int kuadrat (int b);**

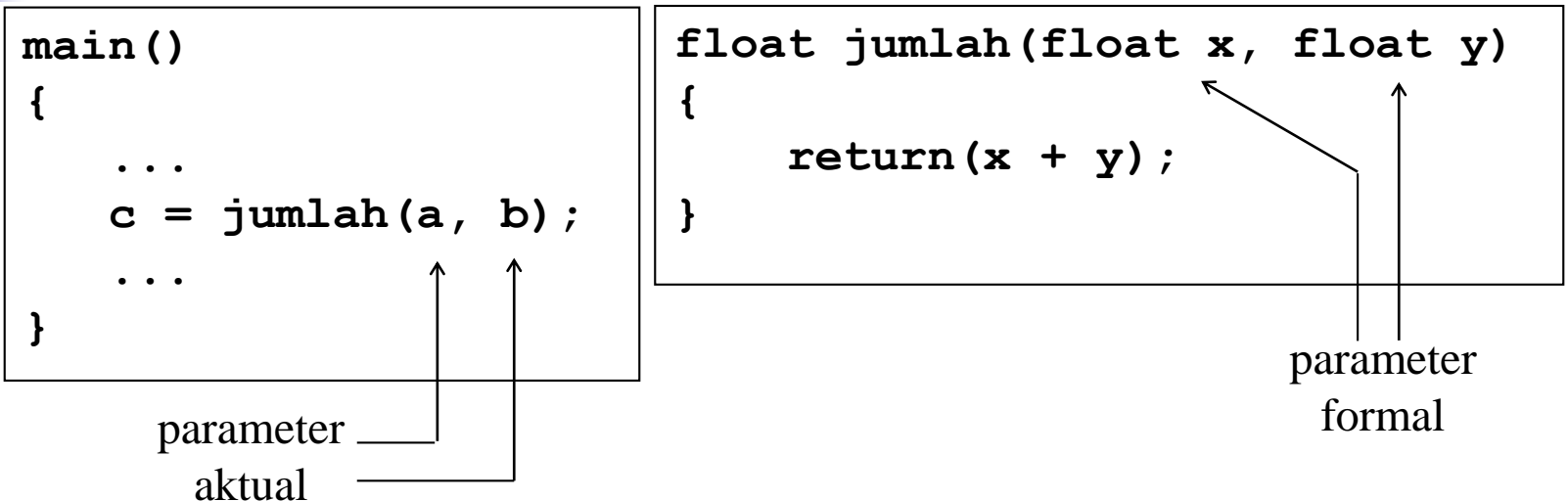
← Nama fungsi  
← Diakhiri dengan titik koma  
↑ Tipe parameter  
↑ Tipe keluaran fungsi

## Fungsi yang tidak mempunyai output

**void info\_program (void)**

↑ menyatakan bahwa  
info\_program() tidak  
memiliki parameter

# Parameter Formal dan Parameter Aktual



**Parameter aktual tidak selalu berupa variabel.**

**Contoh :**

**c = jumlah (20.5 , 4);**

**c = jumlah (2+4 , 4/2 );**



# Melewatkan Parameter

---

- Ada 2 cara melewati parameter dalam fungsi:
  - Pemanggilan dengan nilai (*call by value*)
  - Pemanggilan dengan referensi (*call by reference*)



# Call by Value

---

- Melewatkan nilai ke fungsi *by Value*.
- Seluruh fungsi yang telah dibuat didepan adalah Call by Value (Pemanggilan dengan Nilai).



# Contoh Fungsi by value



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

```
void fungsi_nilai (int );
```

```
main()
```

```
{
```

```
    int a;
```

```
    a = 10;
```

```
    printf("nilai a sebelum fungsi = %d\n", a);
```

```
    fungsi_nilai (a);
```

```
    printf("nilai a setelah fungsi = %d\n", a);
```

```
}
```

```
void fungsi_nilai (int b)
```

```
{
```

```
    b = b + 5;
```

```
    printf ("nilai a di fungsi = %d\n",b);
```

```
}
```



# Call by Reference

---

- Melewatkan nilai ke fungsi *by Reference*

# Contoh Fungsi by referensi



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

```
void fungsi_nilai (int *b );
```

```
main()
```

```
{
```

```
    int a;
```

```
    a = 10;
```

```
    printf("nilai a sebelum fungsi = %d\n", a);
```

```
    fungsi_nilai (&a);
```

```
    printf("nilai a setelah fungsi = %d\n", a);
```

```
}
```

```
void fungsi_nilai (int *b)
```

```
{
```

```
    *b = *b + 5;
```

```
    printf ("nilai a di fungsi = %d\n",*b);
```

```
}
```