

FUNGSI 1

Tim Ajar Mata Kuliah Dasar Pemrograman 2025

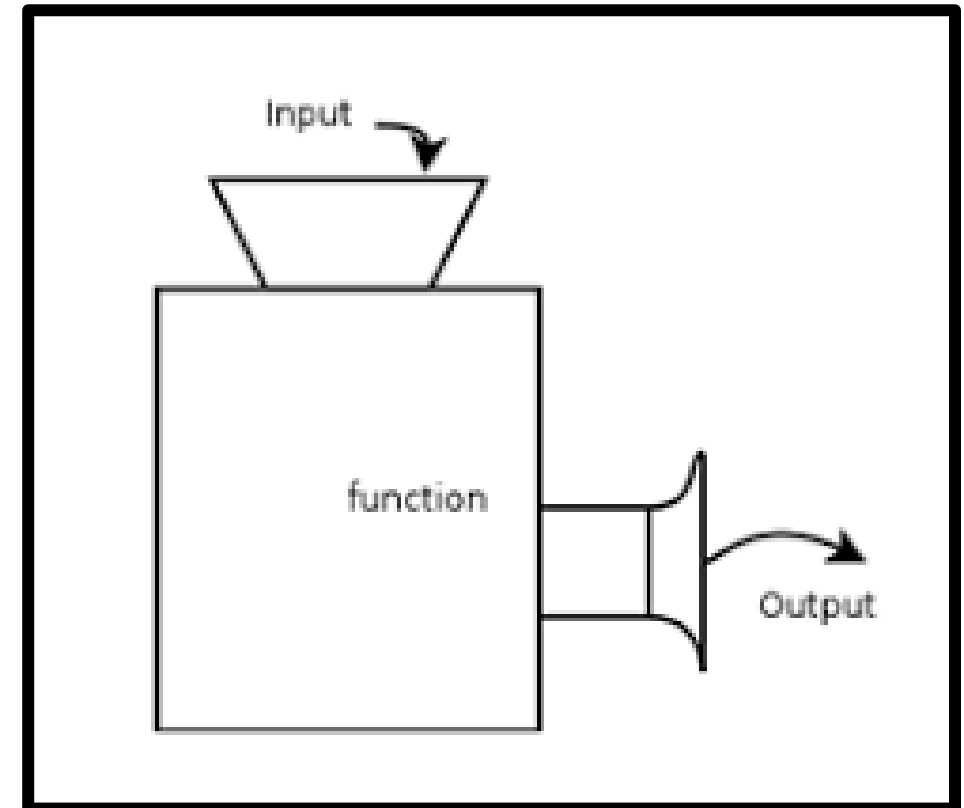
Tujuan

Setelah menempuh materi ini, mahasiswa hendaknya mampu:

1. Menguasai tentang konsep Fungsi
2. Menguasai cara pendeklarasian Fungsi
3. Menguasai cara pemanggilan Fungsi

DEFINISI FUNGSI

- **Fungsi (*function*)** adalah sejumlah instruksi yang dikelompokkan menjadi satu, berdiri sendiri, yang berfungsi untuk menyelesaikan suatu pekerjaan tertentu.
- Dengan menggunakan fungsi, program dapat disusun secara lebih terstruktur (lebih modular) dan lebih efektif



DEFINISI FUNGSI

- **Modular**

Fungsi adalah sekelompok *statement* yang **menjalankan tugas tertentu, dikelompokkan dan dipisah dalam blok tersendiri** dengan diberikan nama tertentu. Jika tugas tersebut diperlukan, program cukup memanggil nama fungsi tersebut.

- **Efektif**

Jika tugas yang sama dilakukan dalam program berulang-ulang, maka **tidak perlu dituliskan ulang kodenya**, hanya cukup memanggil fungsi yang sudah ada. Hal ini mengurangi duplikasi kode dan mempermudah pemeliharaan program.



DEKLARASI FUNGSI

```
static TypeDataKembalian namaFungsi() {  
    // statement  
    //statement  
}
```

Keterangan:

- **Static:** Jenis fungsi yang dibuat bersifat *static*, agar dapat secara langsung di panggil di fungsi main yang juga bersifat static
- **TypeDataKembalian:** tipe data dari nilai yang dikembalikan (*output*) setelah fungsi dieksekusi
- **namaFungsi():** nama fungsi yang dibuat



Contoh:

Pembuatan Fungsi:



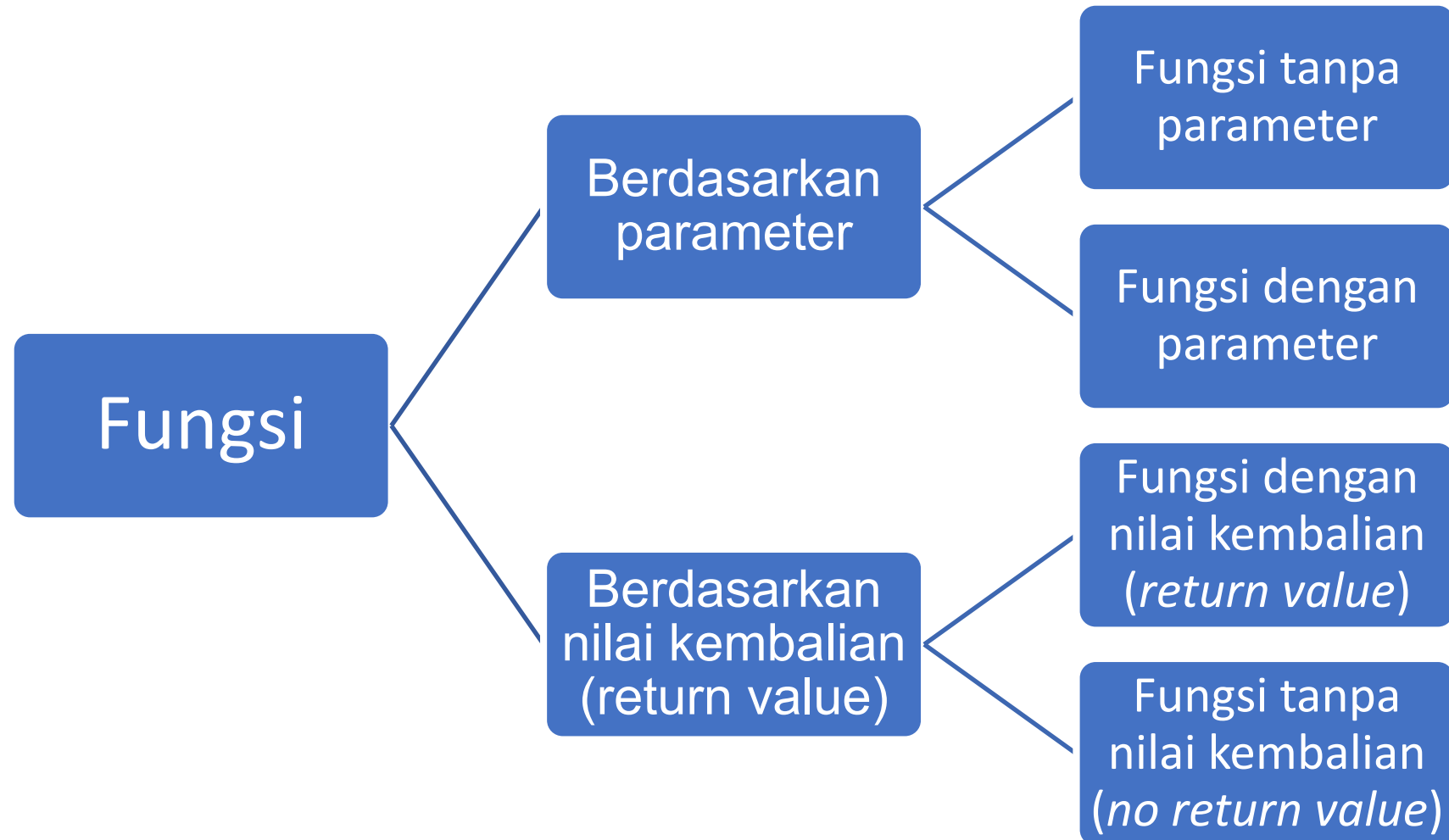
```
1 public static void Menu() {  
2     System.out.println("==== MENU RESTO KAFE =====");  
3     System.out.println("1. Kopi Hitam - Rp 15,000");  
4     System.out.println("2. Cappuccino - Rp 20,000");  
5     System.out.println("3. Latte - Rp 22,000");  
6     System.out.println("4. Teh Tarik - Rp 12,000");  
7     System.out.println("5. Roti Bakar - Rp 10,000");  
8     System.out.println("6. Mie Goreng - Rp 18,000");  
9     System.out.println("=====");  
10    System.out.println("Silakan pilih menu yang Anda inginkan.");  
11 }
```

Pemanggilan Fungsi:



```
1 public static void main(String[] args) {  
2     Menu();  
3 }
```

FUNGSI



Parameter Fungsi

- Fungsi memerlukan parameter ketika fungsi tersebut membutuhkan data yang asalnya dari luar fungsi untuk diolah dalam fungsi.
- Fungsi boleh tidak memiliki sama sekali parameter.
- Jumlah parameter fungsi yang bisa dimiliki fungsi menyesuaikan kebutuhan, dan tidak ada batasan maksimalnya.
- Pada saat deklarasi fungsi, penulisan parameter adalah dengan cara:
tipe_data nama _parameter.

1.a. Fungsi tanpa Parameter

- Fungsi tanpa parameter tidak memerlukan nilai yang dilewatkan sebagai input pada saat pemanggilan suatu fungsi

```
1 public static void Menu() {  
2     System.out.println("==== MENU RESTO KAFE ====");  
3     System.out.println("1. Kopi Hitam - Rp 15,000");  
4     System.out.println("2. Cappuccino - Rp 20,000");  
5     System.out.println("3. Latte - Rp 22,000");  
6     System.out.println("4. Teh Tarik - Rp 12,000");  
7     System.out.println("5. Roti Bakar - Rp 10,000");  
8     System.out.println("6. Mie Goreng - Rp 18,000");  
9     System.out.println("=====");  
10    System.out.println("Silakan pilih menu yang Anda inginkan.");  
11 }
```

Di dalam kurung
tidak ada parameter

1.b. Fungsi dengan Parameter ...(1)

- Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi. Parameter berperan sebagai *input* untuk fungsi.
- Deklarasi:

```
static TipeDataKembalian namaFungsi(TipeData namaParameter,  
    TipeData namaParameterLain) {  
    // statement  
}
```


1.b. Fungsi dengan Parameter ...(2)

- **Parameter:** sebagai tempat untuk data masukan yang akan diolah dalam fungsi. Banyaknya parameter menyesuaikan kebutuhan. Setiap parameter terdiri dari tipe data dan nama parameter (misal: `int a`, `float b`), sama persis seperti deklarasi variabel.
- Parameter ditulis di antara *parenthesis* (...) setelah nama fungsi.
- Bila terdapat lebih dari satu parameter, maka dipisah dengan tanda koma dan masing-masing parameter harus dideskripsikan tipe datanya.

1.b. Fungsi dengan Parameter ...(3)

- **Contoh:**

Pembuatan Fungsi dengan parameter:



```
1 public static void Menu(String namaPelanggan, boolean isMember) {  
2     System.out.println("Selamat datang, " + namaPelanggan + "!");  
3  
4     if (isMember) {  
5         System.out.println("Anda adalah member, dapatkan diskon 10% untuk setiap pembelian!");  
6     }  
7 }
```

Pemanggilan Fungsi dan memberi nilai parameter:



```
1 Menu("Andi", true);
```



2.a. Fungsi tanpa nilai kembalian

- Fungsi **void** tidak memerlukan return.

```
static void namaFungsi  
(TipeData namaParameter)  
{  
    // statement  
}
```

Function does not
return a Value



```
void area();
```



Nothing specified in
Brackets - Function
does not take any argument

2.a. Fungsi tanpa nilai kembalian

- **Contoh**

Fungsi tanpa nilai kembalian (void)

```
static void salam() {  
    System.out.println(x:"Halo, selamat belajar fungsi!");  
}
```

Pemanggilan fungsi tanpa nilai kembalian (void)

```
public static void main(String[] args) {  
    salam();  
}
```

2.b. Fungsi yang mengembalikan Nilai...(1)

- Sebuah fungsi yang dapat mengembalikan nilai *output* sehingga bisa diolah pada proses berikutnya.
- Pengembalian nilai pada fungsi menggunakan *keyword return*.
- Fungsi yang memiliki tipe data fungsi **selain void yang memerlukan return**.
- **Nilai yang di-return-kan** dari suatu fungsi harus **sesuai dengan tipe data fungsi**. Misalnya jika tipe data fungsi int, maka nilai yang di-return-kan harus nilai itu.

2.b. Fungsi yang mengembalikan Nilai...(2)

- Sebuah fungsi yang dapat mengembalikan nilai *output* sehingga bisa diolah pada proses berikutnya.
- Pengembalian nilai pada fungsi menggunakan *keyword* **return**.
- Deklarasi fungsi:

```
static TypeDataKembalian namaFungsi (TipeData namaParameter) {  
    // statement  
    return variabelOutput;  
}
```




2.b. Fungsi yang mengembalikan Nilai...(3)

- **Contoh**

Pembuatan Fungsi dengan parameter dan return value:

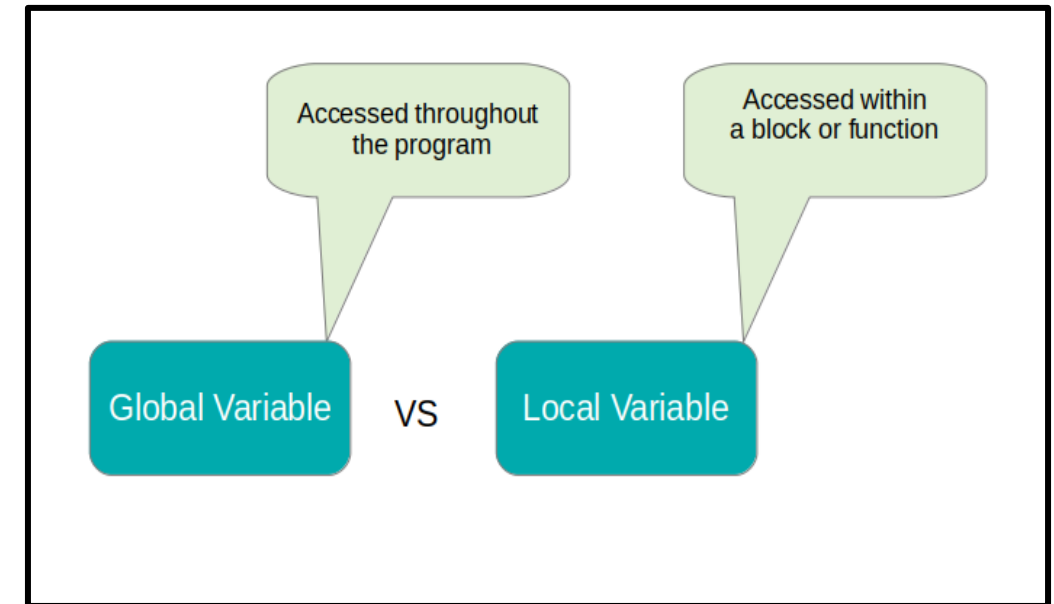
```
static int luasPersegi(int sisi){  
    int luas = sisi * sisi;  
    return luas;  
}
```

Pemanggilan Fungsi dan memberi nilai parameter:

```
System.out.println("Luas Persegi dengan sisi 5 = " + luasPersegi(5));  
  
int luasan = luasPersegi(6);
```

SCOPE OF VARIABLE... (1)

- **Variabel Lokal**: variabel yang dideklarasikan dalam suatu fungsi, dan hanya bisa diakses atau dikenali dari dalam fungsi itu sendiri.
- **Variabel Global**: variabel yang dideklarasikan di luar blok fungsi, dan bisa diakses atau dikenali dari fungsi manapun.
- Variabel Global pada Java diberi awalan **static** agar variabel tersebut bisa dipanggil/ diakses langsung dari fungsi/ metode static lain pada kelas yang sama (atau melalui nama kelas).



SCOPE OF VARIABLE... (2)

```
public class Fungsi1 {
```

```
    static int a = 10, b = 5;  
    static double c;
```

Variabel Global

```
    static int Kali() {  
        int hasilKali = a * b;  
        return hasilKali;  
    }
```

Variabel Lokal

```
    static void Tambah() {  
        int hasilTambah = a + b;  
        System.out.println("Hasil Tambah adalah " + hasilTambah);  
    }
```

Variabel Lokal

```
    public static void main(String[] args) {  
        System.out.println("Hasil Kali adalah " + Kali());  
        Tambah();  
    }
```

```
}
```

Fungsi Pass by Value Vs Pass by Reference


- **Pass by Value** mengirimkan parameter berdasarkan nilai variabel asalnya yang akan dihubungkan terhadap parameter fungsi pemanggil.
- **Pass by Reference** mengirimkan parameter berdasarkan alamat dari nilai tertentu, maka dari itu bila ada nilai yang dirubah dari alamat asalnya maka akan terjadi perubahan juga terhadap nilai parameter yang dipanggil.

pass by reference

cup = 

fillCup()

pass by value

cup = 

fillCup()

Fungsi Pass by Value Vs Pass by Reference

```
public class PassbyValue {  
    static void UbahNilai(int j){  
        j=33;  
    }  
  
    public static void main(String[] args) {  
        int i=10;  
        System.out.println(i);  
        UbahNilai(i);  
        System.out.println(i);  
    }  
}
```

run:

10

10

```
public class PassbyRef {  
    static void UbahArray (int[] arr){  
        for (int i=0; i<arr.length; i++){  
            arr[i]=i+50;  
        }  
    }  
  
    public static void main(String[] args) {  
        int[] umur = {10, 11, 12};  
  
        for (int i = 0; i < umur.length; i++) {  
            System.out.println(umur[i]);  
        }  
  
        UbahArray(umur);  
        for (int i=0; i<umur.length; i++){  
            System.out.println(umur[i]);  
        }  
    }  
}
```

run:

10

11

12

50

51

52

Sebuah Fungsi dapat meng-CALL Fungsi Lain

```
public class FungsiCall {  
    public static void main(String[] args) {  
        int hasil=Hitung(5,2);  
        System.out.println("Hasil Akhirnya adalah "+hasil);  
    }  
  
    static int Tambah(int x, int y){  
        int Z=x+y;  
        return Z;  
    }  
  
    static int Hitung (int c, int d){  
        int E;  
        c*=2;  
        d*=2;  
        E=Tambah(c,d);  
        return E;  
    }  
}
```

```
run:  
Hasil Akhirnya adalah 14  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Dua Fungsi dapat saling mengCALL

```
public class FungsiCallFungsi {  
    public static void main(String[] args) {  
        int hasil=Hitung(5,2);  
        System.out.println("Hasil Akhirnya adalah "+hasil);  
    }  
  
    static int Tambah(int x, int y){  
        int Z=x+y;  
        while (Z<50){  
            x+=2;  
            y+=2;  
            Z=Hitung(x,y);  
        }  
        return Z;  
    }  
  
    static int Hitung (int c, int d){  
        int E;  
        c*=2;  
        d*=2;  
        E=Tambah(c,d);  
        return E;  
    }  
}
```

run:

Hasil Akhirnya adalah 80

BUILD SUCCESSFUL (total time: 2 seconds)

Java Varargs (Variable Arguments)

- Varargs dipakai dengan cara menempatkan parameter-parameter dalam sebuah array dan array tersebut yang akan menjadi parameter dari fungsi.
- Apabila tidak diketahui berapa jumlah dari parameter suatu fungsi dengan pasti, kita bisa menggunakan **Variable Length Argument** (Varargs).
- Deklarasi:

```
accessModifier methodName(datatype... arg) {  
    // method body  
}
```


Rule Varargs (Variable Arguments)

- Setiap fungsi hanya diperkenankan memiliki 1 *variable arguments*
- Variabel argument diletakkan di paling akhir jika memiliki lebih dari 1 parameter.

```
void method(String... gfg, int... q)
{
    // Compile time error as there
    // are two varargs
}
```

(1)

```
void method(int... gfg, String q)
{
    // Compile time error as vararg
    // appear before normal argument
}
```

(2)



Java Varargs (Variable Arguments)

Contoh

```
public class ContohVarargs1 {  
    static void tampilIsi(int ...a){  
        System.out.println("Jumlah parameter ada "+ a.length);  
        System.out.println("isinya : ");  
  
        for(int i=0; i<a.length; i++){  
            System.out.println("Parameter ke-"+i+" : "+a[i]);  
        }  
  
        System.out.println();  
    }  
  
    public static void main(String args[]){  
        tampilIsi(10); // hanya ada satu parameter  
        tampilIsi(4,5,3); // ada 3 parameter  
    }  
}
```

Hasil run:

```
run:  
Jumlah parameter ada 1  
isinya :  
Parameter ke-0 : 10  
  
Jumlah parameter ada 3  
isinya :  
Parameter ke-0 : 4  
Parameter ke-1 : 5  
Parameter ke-2 : 3
```



Java Varargs (Variable Arguments)

Contoh

```
static int sum(int... angka) {  
    int total = 0;  
    for (int a : angka) {  
        total += a;  
    }  
    return total;  
}  
  
public static void main(String[] args) {  
    int t1 = sum(...angka:1, 2, 3);           // t1 = 6  
    int t2 = sum(...angka:10, 20, 30, 40);    // t2 = 100  
    System.out.println(t1);  
    System.out.println(t2);  
}
```

Hasil run:

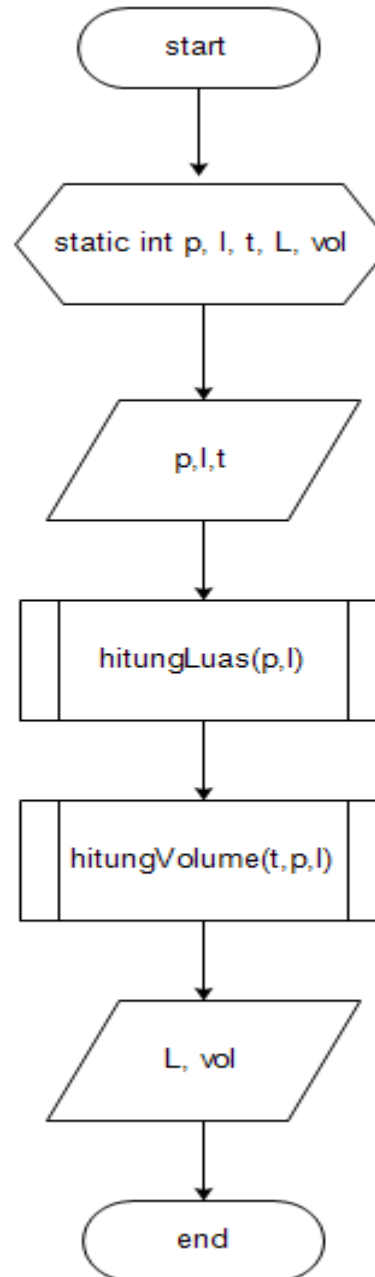
```
6  
100
```

Flowchart Fungsi

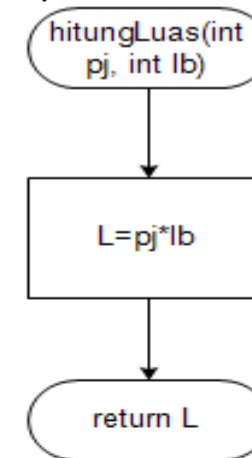
Contoh 1

- Buatlah *flowchart* untuk menghitung luas persegi dan volume balok menggunakan fungsi.

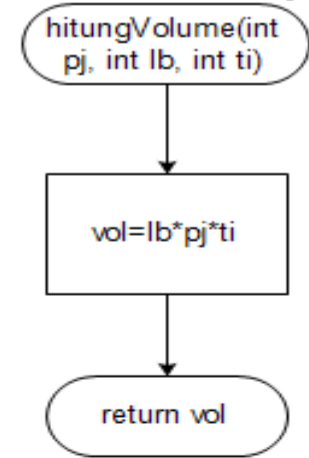
Flowchart : main()



Flowchart : hitungLuas (int pj, int lb)



Flowchart : hitungVolume (int pj, int lb, int ti)



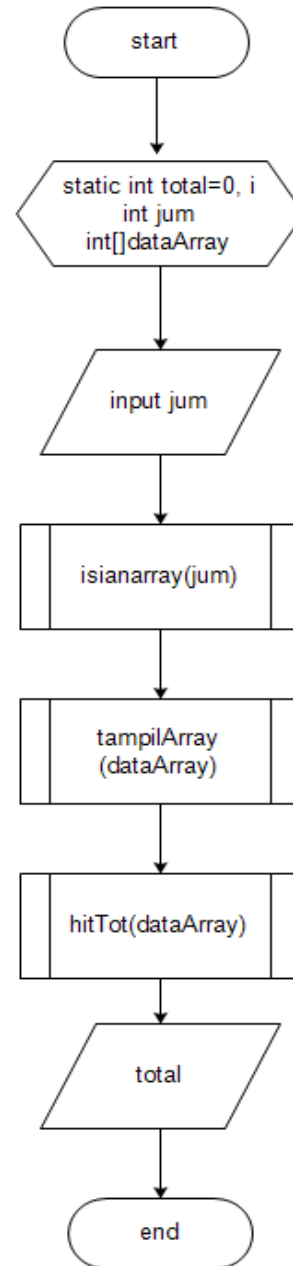
Contoh 2

- Buatlah flowchart untuk menghitung total nilai dari N mahasiswa, dimana program yang dibuat terdiri dari 3 fungsi yaitu:
 1. Fungsi input N nilai (N adalah jumlah banyaknya nilai yang diinputkan)
 2. Tampil nilai
 3. Total nilai

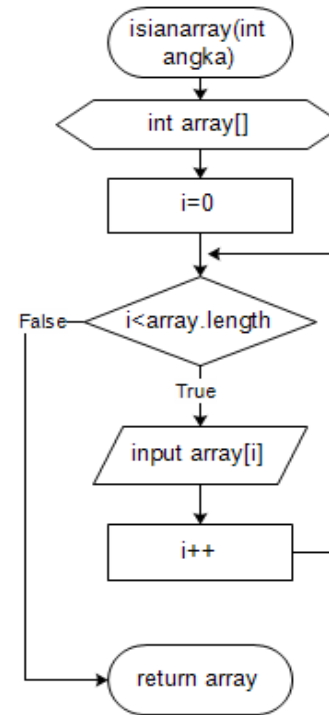


Contoh 2

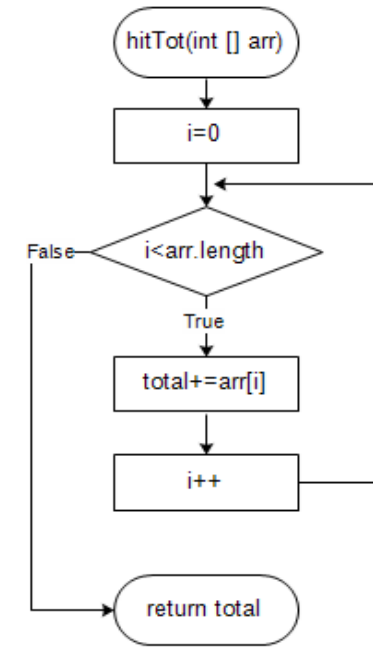
Flowchart : main()



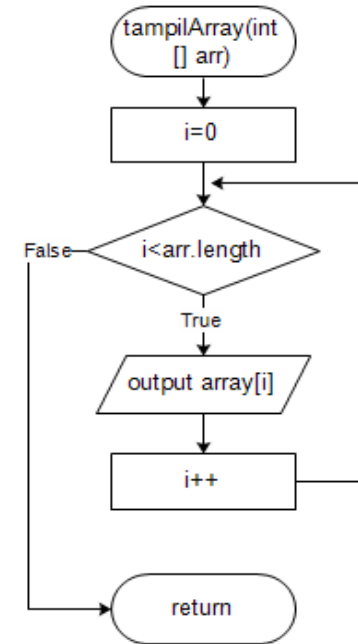
Flowchart : isianarray (angka)



Flowchart : hitTot (int[]arr)



Flowchart : tampilArray (int[]arr)



Tugas

1. Ibu Mariana mengelola sebuah kafe. Berikut adalah rekap penjualan 5 menu dari hari pertama hingga ketujuh:

	Hari ke 1	Hari ke 2	Hari ke 3	Hari ke 4	Hari ke 5	Hari ke 6	Hari ke 7
Kopi	20	20	25	20	10	60	10
Teh	30	80	40	10	15	20	25
Es Kelapa Muda	5	9	20	25	10	5	45
Roti Bakar	50	8	17	18	10	30	6
Gorengan	15	10	16	15	10	10	55

Buatlah flowchart untuk menggambarkan proses pengolahan dan menampilkan beberapa informasi penjualan pada kafe tersebut. Flowchart terdiri dari beberapa fungsi sebagai berikut:

- a. Fungsi untuk menginputkan data penjualan
- b. Fungsi untuk menampilkan seluruh data penjualan dari hari pertama hingga hari terakhir
- c. Fungsi untuk menampilkan menu yang memiliki penjualan tertinggi
- d. Fungsi untuk menampilkan rata-rata penjualan untuk setiap menu

Tugas

2. Di suatu kelas, terdapat 5 mahasiswa yang mengikuti 3 mata kuliah: Algoritma dan Struktur Data, Basis Data, dan Statistik. Nilai tiap mahasiswa untuk tiap mata kuliah sudah tersedia dalam bentuk tabel (5×3). Buatlah flowchart yang terdiri dari beberapa fungsi untuk algoritma yang:
 - a. Fungsi untuk membaca data nilai 5 mahasiswa untuk 3 mata kuliah.
 - b. Fungsi untuk menghitung rata-rata nilai setiap mahasiswa.
 - c. Fungsi untuk menentukan status setiap mahasiswa:
 - “LULUS” jika rata-rata ≥ 60
 - “TIDAK LULUS” jika rata-rata < 60
 - d. Fungsi untuk menampilkan nama, rata-rata nilai, dan status kelulusan masing-masing.