

# Praktikum Dasar Pemrograman

## Pengenalan Bahasa C

2023

# 1 Tujuan

- Mahasiswa mengerti cara membuat dan memanggil fungsi pada bahasa pemrograman C.
- Mahasiswa mampu menggunakan passing parameter by value dan by reference pada bahasa pemrograman C.
- Mahasiswa mampu mengerti dan mengaplikasikan konsep rekursi pada bahasa pemrograman C.

# 2 Fungsi

Fungsi adalah sebuah kumpulan statement untuk melakukan tugas spesifik, yang bisa membutuhkan input ataupun tidak, untuk menghasilkan output yang sesuai. Keuntungan menggunakan fungsi pada bahasa pemrograman C adalah:

- Beberapa cuplikan kode dapat digunakan kembali saat menggunakan fungsi. Kita dapat memanggil fungsi C berapa kali pun dalam suatu program dan dari mana saja dalam suatu program.
- Program c yang besar dapat dibagi ke dalam beberapa fungsi sehingga dapat dengan mudah untuk dilacak.

## 2.1 Function Declaration

Setiap program C mempunyai minimal satu fungsi, yaitu fungsi main(). Anda juga dapat mendefinisikan fungsi selain main() Syntax :

```
return_type function_name( parameters list){  
    // function body  
    return something;  
}
```

- Return Type.  
Tipe data yang harus dikembalikan suatu fungsi.
- function\_name.  
Nama fungsi
- parameters list.  
Parameter dari fungsi.
- Function body.  
Kumpulan statemen yang mendefinisikan apa yang dilakukan oleh fungsi.
- return something;  
merupakan statement untuk mengembalikan nilai dari fungsi. Untuk fungsi yang tidak mengembalikan nilai, dapat digunakan return\_type void. Untuk keluar dari fungsi itu hanya perlu menggunakan statement return

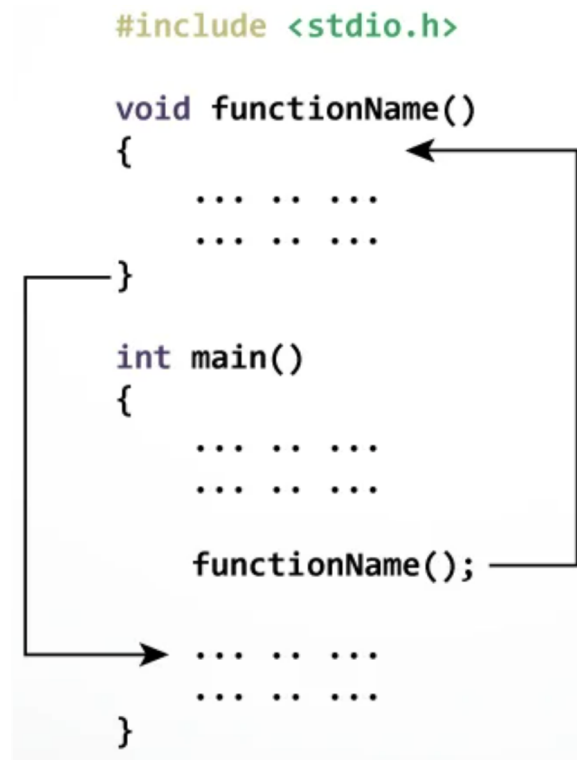
Contoh

```

1 float TriangleArea(float Base, float Height)
2 {
3     float Area;
4     Area = 0.5*Base*Height;
5     return Area;
6 }

```

## 2.2 Memanggil Fungsi



Gambar 1

```

1  #include <stdio.h>
2  // Mendeklarasikan fungsi luasSegitiga
3  // Parameter input Alas , dan Tinggi
4  // Output float
5  float TriangleArea(float Base, float Height)
6  {
7      float Area;
8      Area = 0.5*Base*Height;
9      return Area;
10 }
11 int main()
12 {
13     float Bs = 4,Hg=10,L;
14     // Memanggil Fungsi TriangleArea
15     L=TriangleArea(Bs,Hg);
16     printf("Area = %f",L);
17     return 0;
18 }

```

1. Baris 5-10: Mendefinisikan fungsi `TriangleArea` dengan

- 2 parameter input/masukan: input `Base` dan `Height` dengan tipe data `float`.
- Output bernilai tunggal dengan tipe data `float`.

## 2.3 Fungsi dengan Argumen

### 2.3.1 Argumen

Jika suatu fungsi diharapkan untuk menggunakan argumen, maka variabel sebagai parameter yang menerima nilai dari argumen tersebut harus dideklarasikan terlebih dahulu.

#### 1. Parameter :

- (a) Parameter adalah variabel dalam fungsi untuk merujuk ke salah satu bagian dari data yang diberikan sebagai input ke fungsi.
- (b) Data ini disebut argumen.

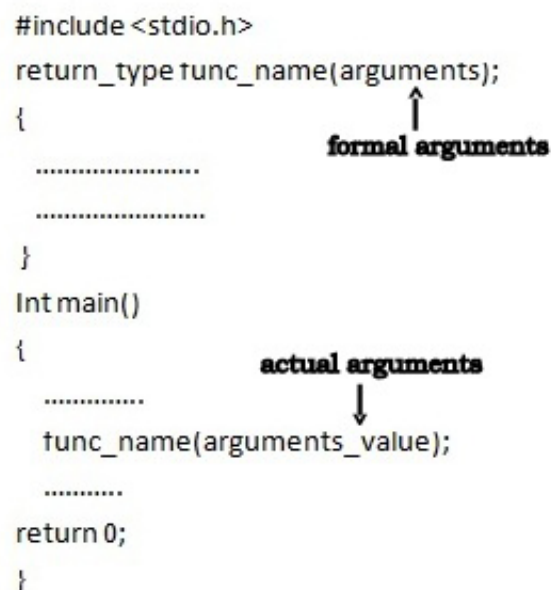
#### 2. Formal Parameter:

- (a) Parameter yang Ditulis dalam Definisi Fungsi Disebut "Parameter Formal".
- (b) Parameter formal selalu variabel, sedangkan parameter aktual tidak harus variabel.

#### 3. Actual Parameter:

- (a) Parameter yang Ditulis ketika memanggil fungsi.
- (b) Dapat berupa angka, ekspresi, atau bahkan panggilan fungsi.

```
#include <stdio.h>
return_type tunc_name(arguments);
{
    .....
    .....
}
Int main()
{
    .....
    tunc_name(arguments_value);
    .....
return 0;
}
```



Gambar 2

## 2.4 Parameter Passing

Passing parameter merupakan aktivitas menyalurkan nilai pada parameter saat memanggil fungsi. Pada umumnya, dikenal dua macam passing parameter yaitu:

- Pass parameter by value, yaitu menyalurkan **nilai** dari tiap parameter yang diberikan.
- Pass parameter by reference, yaitu menyalurkan **alamat** dari tiap parameter yang diberikan.

### 2.4.1 Passing Parameter by Value

Listing 1: Passing by Value

```
1 #include <stdio>
2 int swapAndReturnTheSum(int x, int y) {
3     int z;
4     z = x;
5     x = y;
6     y = z;
7     return x+y;
8 }
9 int main()
10 {
11     int a = 1;
12     int b = 2;
13     int sum = swapAndReturnTheSum(a,b);
14     printf("sum: %d\n",sum);
15     printf("values of a dan b now:\n");
16     printf("a: %d\n",a);
17     printf("b: %d\n",b);
18 }
```

Perhatikan potongan kode pada Listing 1. Baris 3-6 dari kode tersebut adalah operasi untuk menukar nilai dari 2 variabel. Namun, apabila program tersebut dijalankan, maka akan muncul output

```
sum: 3
values a and b now:
a: 1
b: 2
```

Nilai dari a dan b tidak bertukar. Untuk passing parameter by value, apapun yang dilakukan pada function body tidak akan berpengaruh pada parameter yang "dipassingkan". Nilai dari parameter aktual akan diassign pada parameter formal.

### 2.4.2 Passing Parameter by Reference

Perhatikan baris 2 pada potongan kode berikut:

Listing 2: Passing by Reference

```
1 #include <stdio>
2 int swapAndReturnTheSum(int &x, int &y) {
3     int z;
4     z = x;
```

```

5     x = y;
6     y = z;
7     return x+y;
8 }
9 int main()
10 {
11     int a = 1;
12     int b = 2;
13     int sum = swapAndReturnTheSum(a,b);
14     printf("sum: %d\n",sum);
15     printf("values of a dan b now:\n");
16     printf("a: %d\n",a);
17     printf("b: %d\n",b);
18 }

```

Apabila program tersebut dijalankan, maka akan muncul output

```

sum: 3
values of a and b now
a: 2
b: 1

```

Ketika fungsi `swapAndReturnTheSum(a,b)` dipanggil, alamat memori variabel `a` dan `b` "dipassingkan" pada fungsinya. Sehingga pada potongan kode di baris 4-6, `x` dan `y` akan mengacu pada memori parameter aktual yang dimasukkan di baris ke 13. Ketika melakukan passing by reference, kita tidak bisa memanggil fungsi dengan parameter yang tidak memiliki alamat memori. Sebagai contoh `tukarDanKembalikansumnya(1,2)` tidak bisa dilakukan karena angka 1 dan 2 bukan variabel dan tidak memiliki alamat memori.

## 2.5 Tugas Pendahuluan

1. Buatlah fungsi yang dapat menerima 2 buah bilangan bulat `a` dan `b` kemudian mengembalikan nilai dari  $a^b$
2. Masalah-masalah apa yang akan lebih mudah diselesaikan dengan menggunakan fungsi?

## 3 Rekursi

Rekursi merujuk kepada definisi suatu hal yang dilakukan secara berulang-ulang.

Rekursi adalah ketika suatu fungsi dalam function bodynya memanggil fungsi itu sendiri. Sebagai contoh, perhatikan potongan kode berikut:

**Listing 3:** Factorial dengan rekursi

```

1 int factorial(int n) {
2     if (n==1)
3         return 1;
4     return n*factorial(n-1);
5 }

```

The factorial function calls another factorial function in line 4. Initially, the function  $factorial(n)$  is called. This function however will return  $n \times factorial(n-1)$ , kemudian  $factorial(n-1)$  akan mengembalikan  $(n-1) \times factorial(n-1-1)$ . Akhirnya menjadi seperti ini:

$$\begin{aligned} factorial(n) &= n \times factorial(n-1) \\ &= n \times (n-1) \times factorial(n-2) \\ &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times factorial(1) \\ &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \end{aligned}$$

### 3.1 Tugas Pendahuluan

1. Diberikan sebuah baris bilangan 1, 5, 14, 30, ... dst. Buatlah sebuah program yang mengimplementasikan fungsi rekursif untuk menentukan bilangan ke-n dari pola tersebut.