

# Modul Praktikum Dasar Pemrograman

Halaman ini sengaja dikosongkan.

# Daftar Isi

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Struktur Bahasa C</b>                   | <b>1</b>  |
| 1.1      | Tujuan                                     | 1         |
| 1.2      | Membuat Project Baru pada IDE Code::Blocks | 1         |
| 1.2.1    | Langkah-Langkah Untuk membuat Project Baru | 1         |
| 1.2.2    | Latihan                                    | 5         |
| 1.3      | Struktur Bahasa C                          | 5         |
| 1.3.1    | latihan                                    | 5         |
| 1.4      | Tipe Data dan Variabel                     | 6         |
| 1.4.1    | Tipe Data                                  | 6         |
| 1.4.2    | Variabel                                   | 6         |
| 1.4.2.1  | Operator Aritmatika dan Penugasan          | 6         |
| 1.4.3    | latihan                                    | 7         |
| 1.5      | Output dan Input                           | 8         |
| 1.5.1    | Fungsi printf()                            | 8         |
| 1.6      | scanf                                      | 9         |
| 1.6.1    | Escape Sequence                            | 11        |
| 1.6.2    | Latihan                                    | 13        |
| <b>2</b> | <b>Instruksi Pemilihan</b>                 | <b>15</b> |
| 2.1      | Tujuan                                     | 15        |
| 2.2      | Ekspresi Perbandingan dan Logika           | 15        |
| 2.2.1    | Ekspresi Perbandingan                      | 15        |
| 2.2.2    | Ekspresi Logika                            | 15        |
| 2.3      | Statement if                               | 16        |
| 2.4      | Statement if-else                          | 17        |
| 2.5      | Statement if-else if                       | 18        |
| 2.6      | Nested if                                  | 19        |
| 2.7      | Latihan                                    | 20        |
| <b>3</b> | <b>Perulangan dan Array</b>                | <b>21</b> |
| 3.1      | Tujuan                                     | 21        |
| 3.2      | Perulangan                                 | 21        |
| 3.2.1    | while loop                                 | 21        |
| 3.2.2    | do-while loop                              | 23        |
| 3.2.3    | for loop                                   | 24        |
| 3.3      | Array                                      | 25        |
| 3.3.1    | Array 1D                                   | 25        |

|          |   |           |
|----------|---|-----------|
| 3.3.2    | Array 2D dan Array Multidimensi lainnya . . . . . | 26        |
| 3.4      | Contoh Aplikasi Perulangan dan Array . . . . .    | 26        |
| 3.4.1    | Linear Search . . . . .                           | 26        |
| 3.4.2    | Bubble Sort . . . . .                             | 26        |
| 3.5      | Latihan . . . . .                                 | 27        |
| <b>4</b> | <b>Fungsi (Subprogram)</b>                        | <b>29</b> |
| 4.1      | Tujuan . . . . .                                  | 29        |
| 4.2      | Deklarasi fungsi . . . . .                        | 29        |
| 4.3      | Memanggil Fungsi . . . . .                        | 31        |
| 4.4      | Fungsi Dengan Argumen . . . . .                   | 32        |
| 4.4.1    | Argumen . . . . .                                 | 32        |
| 4.4.2    | Passing Parameter . . . . .                       | 33        |
| 4.4.2.1  | Passing Parameter by Value . . . . .              | 33        |
| 4.4.2.2  | Passing Parameter by Reference . . . . .          | 33        |
| 4.5      | Rekursi . . . . .                                 | 34        |
| 4.6      | Latihan . . . . .                                 | 35        |

# Bab 1

## Struktur Bahasa C

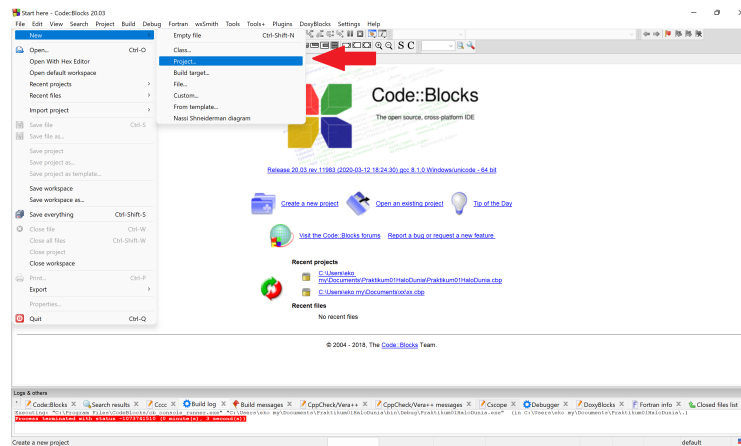
### 1.1 Tujuan

- Mahasiswa mampu membuat project baru pada suatu IDE
- Mahasiswa mengenal struktur bahasa C
- Mahasiswa mengenal jenis - jenis tipe data pada bahasa pemrograman C
- Mahasiswa mengenal jenis - jenis operator
- Mahasiswa mampu menggunakan fungsi untuk membaca data dari keyboard
- Mahasiswa mampu menggunakan fungsi `printf()` untuk mencetak ke-layar.

### 1.2 Membuat Project Baru pada IDE Code::Blocks

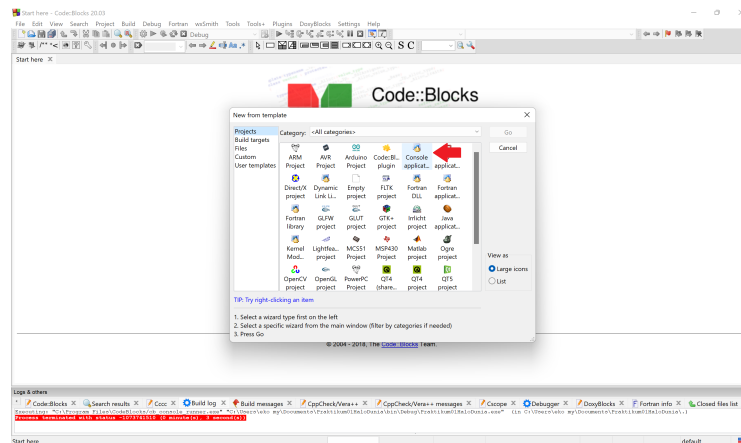
#### 1.2.1 Langkah-Langkah Untuk membuat Project Baru

1. Tekan File > New > Project



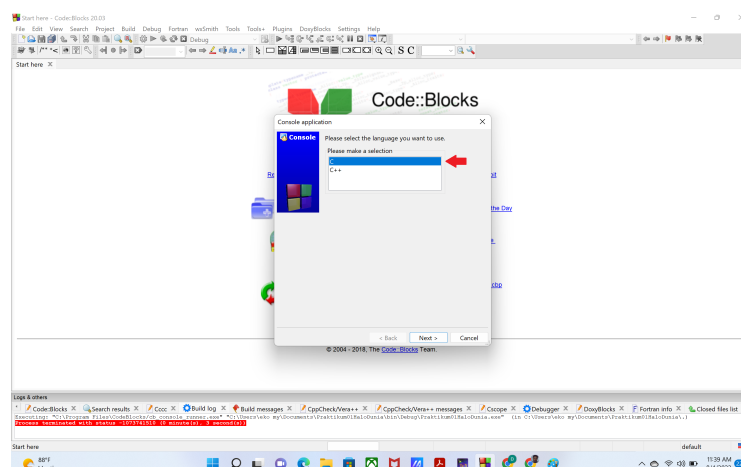
Gambar 1.1

## 2. Klik Console Application



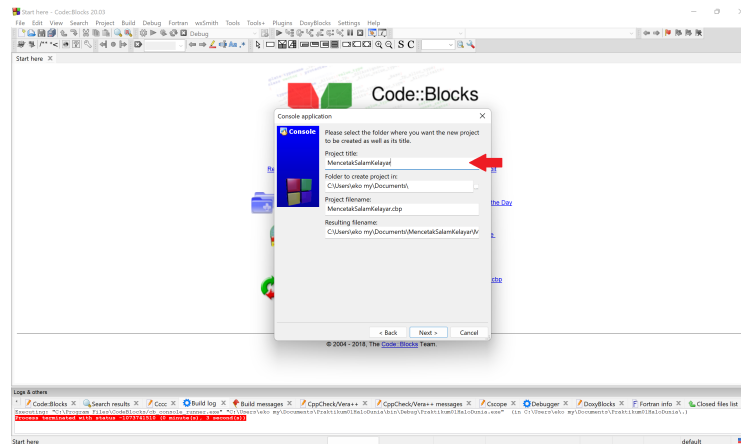
Gambar 1.2

## 3. Pilih bahasa pemrograman C



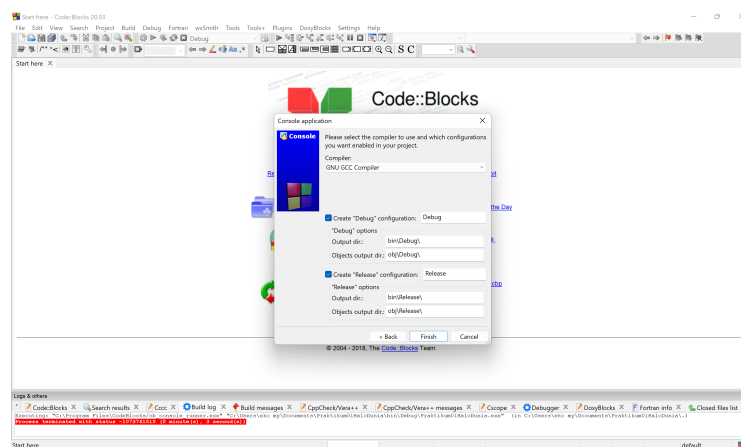
Gambar 1.3

## 4. Beri nama pada project



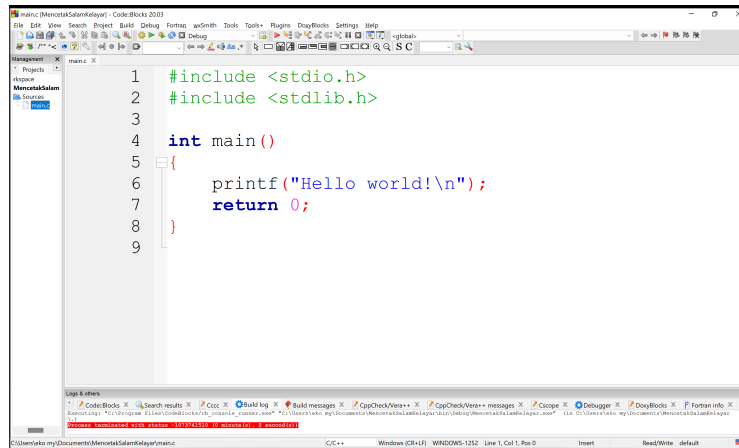
Gambar 1.4

## 5. Pilih compiler GNU GCC Compiler kemudian tekan Finish



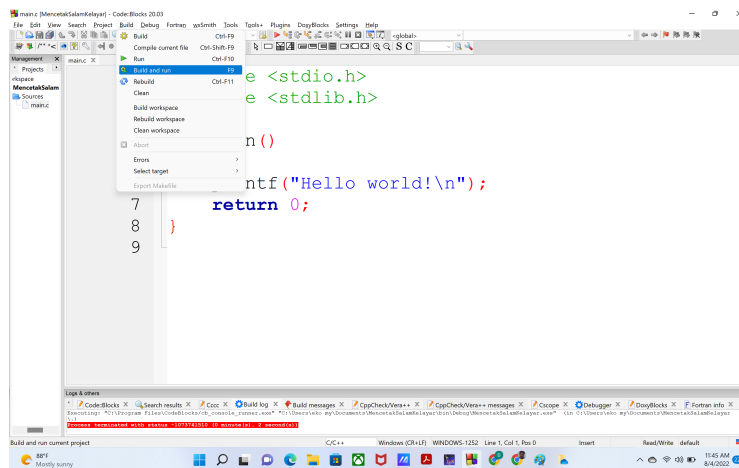
Gambar 1.5

## 6. Tulis kode pada gambar 1.6 pada text editor Code::Blocks



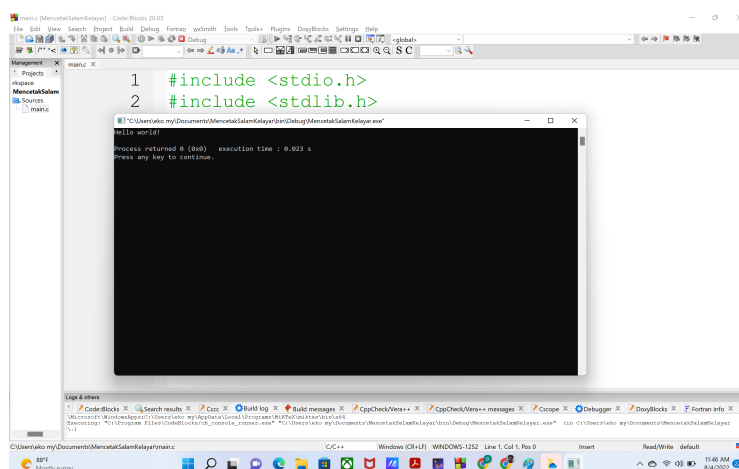
Gambar 1.6

7. Tekan Build—>Build and Run atau F9



Gambar 1.7

8. Output dari program akan terlihat pada console.



Gambar 1.8



### 1.2.2 Latihan

Buatlah project dengan nama HaloDunia dan ketikan program seperti pada Gambar 1.6 tetapi dengan tulisah Hello World! diganti dengan Halo Dunia!

## 1.3 Struktur Bahasa C

**Listing 1.1:** Contoh program sederhana dalam bahasa C

```
1 #include <stdio.h>
2
3 int main()
4 {
5     //Mencetak Kelayar
6     printf("Halo Dunia");
7     return 0;
8 }
```

Program pada Listing 1.1 merupakan program sederhana untuk mencetak "Halo Dunia" ke layar. Berikut penjelasan dari tiap baris yang ada pada program tersebut:

- Baris 1 : `#include <stdio.h>`  
header file library untuk fungsi input dan output seperti `printf()` (sperti yang digunakan pada baris ke 6)
- Baris 2 : Baris Kosong.
- Baris 3 : `int main()`  
adalah fungsi utama yang akan di jalankan terlebih dahulu.
- Baris 4 : `{`  
Memulai blok statement fungsi `main()`
- Baris 5 : `//Mencetak Ke Layar`  
Komentar untuk menjelaskan program, Komentar akan diabaikan oleh program.
- Baris 6 : `printf("Halo Dunia");`  
Mencetak tulisan mencetak "Halo Dunia" ke layar.
- Baris 7 : `return 0;`  
Mengakhir fungsi `main()` (fungsi berakhir ketika mengembalikan nilai)
- Baris 8 : `}`  
Menutup block `main()`

### 1.3.1 latihan

Cobalah menukar baris 6 dan 7 pada Listing 1.1. Apa yang terjadi?  
Bagaimana jika `return 0;` diganti dengan `return 1;`?

## 1.4 Tipe Data dan Variabel

### 1.4.1 Tipe Data

Pada bahasa C, terdapat beberapa tipe data untuk merepresentasikan data yang berupa bilangan bulat, bilangan real, karakter, string, dan lain-lain. Berikut adalah beberapa tipe data pada C.

**Tabel 1.1:** Beberapa Tipe Data pada C

| Tipe data | Size         | Keterangan   |
|-----------|--------------|--|
| int       | 2 or 4 bytes | Menyimpan bilangan integer                         |
| float     | 4 bytes      | Menyimpan bilangan pecahan sampai 8 digit desimal. |
| double    | 8 bytes      | Menyimpan bilangan pecahan sampai 15 digit desimal |
| char      | 1 byte       | Menyimpan satu buah karakter.                      |

Untuk menampilkan data pada layar, setiap tipe data memiliki format specifier yang dapat digunakan pada formatted string. Berikut adalah format specifier untuk beberapa tipe data.

**Tabel 1.2:** Format Specifier

| Format Specifier | Tipe Data    |
|------------------|--------------|
| %d or %i         | int          |
| %f               | float        |
| %lf              | double       |
| %c               | char         |
| %s               | Untuk string |

Masih ada lebih banyak tipe data dari pada yang dituliskan pada Tabel 1.1. Tipe-tipe data ini dan spesifikasinya bisa ditemukan dengan mudah di internet.

### 1.4.2 Variabel

Variabel adalah tempat untuk menyimpan data. Untuk mendeklarasikan variabel, dapat dilakukan dengan cara seperti berikut:

**Listing 1.2:** Deklarasi Variabel C

```
1 JenisTipeData NamaVariabel;
```

#### 1.4.2.1 Operator Aritmatika dan Penugasan

Operasi penugasan dapat dilakukan pada variabel yang tidak memiliki atribut `const` (l-value) sedangkan operator aritmatika dapat melakukannya juga untuk variabel `const` (l-value dan r-value). Berikut merupakan beberapa jenis operator aritmatika.

**Tabel 1.3:** Beberapa Tipe Data pada C

| Operator | Nama        | Contoh   |
|----------|-------------|----------|
| +        | Penjumlahan | $x + y$  |
| -        | Pengurangan | $x - y$  |
| *        | Perkalian   | $x * y$  |
| /        | Pembagian   | $x/y$    |
| %        | Modulus     | $x \% y$ |

Dan berikut beberapa jenis operator penugasan.

**Tabel 1.4:** Operator-operator penugasan

| Operator | Contoh             | Sama Seperti         |
|----------|--------------------|----------------------|
| =        | $x = 5$            | $x = 5$              |
| +=       | $x += 3$           | $x = x + 3$          |
| -=       | $x -= 3$           | $x = x - 3$          |
| *=       | $x *= 3$           | $x = x * 3$          |
| /=       | $x /= 3$           | $x = x / 3$          |
| %=       | $x \% = 3$         | $x = x \% 3$         |
| &=       | $x \& = 3$         | $x = x \& 3$         |
| -=       | $x \text{---} = 3$ | $x = x \text{---} 3$ |
| ^=       | $x \wedge = 3$     | $x = x \wedge 3$     |
| >>=      | $x >> = 3$         | $x = x >> 3$         |
| <<=      | $x << = 3$         | $x = x << 3$         |

Terdapat juga "shorthand" untuk beberapa operasi penugasan seperti  $x+=1$  dan  $x-=1$  yaitu  $++$  dan  $--$  yang disebut dengan increment dan decrement. Shorthand ini digunakan seperti

```
x++;
x--;
++x;
--x;
```

### 1.4.3 latihan

**Listing 1.3:** Contoh Error Operator Penugasan pada Variabel Const

```
1 #include <stdio.h>
2 int main()
3 {
4     //deklarasi variabel const
5     const int x=0;
6     x=1;
7     return 0;
8 }
```

Cobalah program pada Listing 1.3, apa yang terjadi?

## 1.5 Output dan Input

### 1.5.1 Fungsi printf()

Fungsi `printf` pada C digunakan untuk mencetak string ke output yang dilengkapi dengan format specifier yang dimulai dengan `%` pada string.

```
printf(const char *format,v1,v2,...,vn)
```

format specifier untuk beberapa tipe data dapat dilihat pada Tabel 1.2

**Contoh 1.5.1** Mencetak text ke layar.

**Listing 1.4:** Mencetak Tulisan "Pemrograman C Ke layar

```
1  #include <stdio.h>
2  int main()
3  {
4      // Menampilkan tulisan yang terletak diantara tanda
5  petik dua
6      printf("C Programming");
7      return 0;
8  }
```

- Seluruh program C harus berisi fungsi `main()` tempat program memulai menjalankan kode.
- Fungsi `printf()` adalah library untuk mengirim output yang telah diformat ke layar. Fungsi `printf()` mencetak string dalam tanda dua tanda petik.
- Untuk menggunakan fungsi `printf()` dalam program harus disertakan file header `stdio.h` dengan menggunakan statement `#include <stdio.h>`.
- Statement `return 0;` dalam fungsi `main()` menunjukkan status "Exit" dari program.

**Contoh 1.5.2** Mencetak bilangan bulat.

```
1  #include <stdio.h>
2  int main()
3  {
4      int testInteger = 5;
5      printf("Number = %d", testInteger);
6      return 0;
7  }
8  }
```

Pada contoh ini digunakan format specifier `%d` untuk mencetak tipe data `int`. `%d` pada tex akan digantikan oleh isi dari `testInteger`.

**Contoh 1.5.3** Output bilangan pecahan (float atau double)

- Alas : mempunyai tipe data `float`
- Tinggi : mempunyai tipe data `float`
- Luas : mempunyai tipe data `float`
- Persamaan menghitung luas segitiga:

$$Luas = \frac{1}{2} \times Alas \times Tinnggi \quad (1.1)$$

```

1  #include <stdio.h>
2
3  int main()
4  {
5      //Mendeklarasikan variabel
6      float Alas;
7      float Tinggi;
8      float Luas;
9      //Inisialisasi variabel
10     Alas = 10;
11     Tinggi = 5;
12     //MEnghitung Luas Segitiga
13     Luas = 0.5*Alas*Tinggi;
14     //Mencetak luas segitiga ke layar
15     printf("Luas = %f",Luas);
16     return 0;
17 }
18
19

```

Penjelasan program :

**Baris 1:** MEnggunakan fungsi yang terdapat pada `stdio.h` agar dapat menggunakan fungsi `printf()`

**Baris 3** Deklarasi fungsi utama `main()` untuk memulai program.

**Baris 6-8** Mendeklarasikan variabel `Alas`, `Tinggi` dan `Luas` bertipe data `float` untuk menyimpan data parameter luas segitiga.

**Baris 10 dan 11** Memberi nilai ke Variabel `Alas=10` dan `Tinggi=5`

**Baris 13** Menghitung luas alas sesuai dengan persamaan [1.5.1](#)

**Baris 15** Mencetak `Luas` ke layar dengan menggunakan perintah `printf`. Untuk mencetak bilangan pecahan bertipe `float` perintah `printf` memerlukan *format specifier*. dalam contoh tersebut digunakan format `%f`.

## 1.6 scanf

Fungsi `scanf(const char*format, ...)` membaca input dengan format.

## 1. Sintaks

```
scanf(const char *format, ...)
```

## 2. Parameter

format string pada C yang terdiri dari satu atau lebih yang terdiri dari Karakter Whitespace, Karakter Non-whitespace dan Format specifiers.

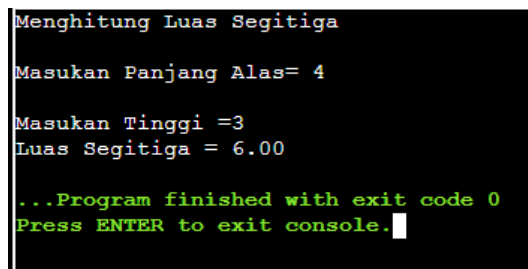
## 3. Return Value

Ketika berhasil maka fungsi mengembalikan jumlah item dari argumen yang berhasil di baca.

### 1.6.1 Menghitung luas segitiga dengan parameter Alas bertipe data dan Tinggi yang diinputkan.

```

1 #include <stdio.h>
2
3 int main()
4 {
5     float Alas ,Tinggi,Luas;
6
7     printf("Menghitung Luas Segitiga\n");
8     printf("\nMasukan Panjang Alas= ");
9     scanf("%f",&Alas);
10    printf("\nMasukan Tinggi =");
11    scanf("%f",&Tinggi);
12    Luas = 0.5*Alas *Tinggi;
13    printf("Luas Segitiga = %.2f", Luas);
14    return 0;
15 }
16
```



```

Menghitung Luas Segitiga
Masukan Panjang Alas= 4
Masukan Tinggi =3
Luas Segitiga = 6.00
...Program finished with exit code 0
Press ENTER to exit console.

```

Gambar 1.9

**Baris 1** `#include <stdio.h>`

Menggunakan file header `stdio.h`

**Baris 4** `float Alas,Tinggi,Luas;`

Mendeklarasikan variabel `Alas`, `Tinggi` dan `Luas` bertipe data `float`. Tipe data yang digunakan adalah `float` karena input dan output parameter luas segitiga adalah sembarang bilangan pecahan positif.

**Baris 7** `printf("Menghitung Luas Segitiga\n");`

Mencetak informasi aplikasi.

**Baris 8** `printf("\nMasukan Panjang Alas= ");`

Memberikan informasi ke user bahwa parameter yang dimasukan adalah panjang alas.

**Baris 9** `scanf("%f",&Alas);`

**Baris 10** `printf("\nMasukan Tinggi =");`

**Baris 11** `scanf("%f",&Tinggi);`

**Baris 12** `Luas = 0.5*Alas *Tinggi;`

**Baris 13** `printf("Luas Segitiga = %.2f", Luas);`, .2 pada %.2f menandakan bahwa hanya 2 angka di belakang koma(decimal point) yang perlu dicetak.

**Contoh 1.6.2** Program memasukan Nama dan Alamat email dari keyboard.

Pada contoh ini dipelajari bagaimana cara menginputkan string atau text dari keyboard dan mencetak kelayar. Input dari contoh program ini ada dua yang terdiri dari `sNama` dan `sAlamatEmail`. Oleh karena text berisi banyak karakter maka masing-masing variabel dideklarasikan sebagai kumpulan karakter dengan jumlah karakter untuk `sNama=20` dan `sAlamatEmail=30`.

```

1  #include <stdio.h>
2
3  int main ()
4  {
5      char sNama[20], sAlamatEmail[30];
6
7      printf("Masukan Nama: ");
8      scanf("%19s", sNama);
9
10     printf("Masukan Alamat email : ");
11     scanf("%29s", sAlamatEmail);
12
13     printf("Nama : %s\n", sNama);
14     printf("Alamat Email:%s", sAlamatEmail);
15     return(0);
16 }
17

```

Penjelasan program :

**Contoh 1.6.2** Program menghitung luas segitiga dengan panjang Alas dan Tinggi yang diinputkan melalui keyboard.

### 1.6.1 Escape Sequence

Escape Sequence adalah urutan karakter yang digunakan untuk memformat output dan tidak ditampilkan ketika dicetak ke layar. Setiap karakter mempunyai fungsi tertentu.

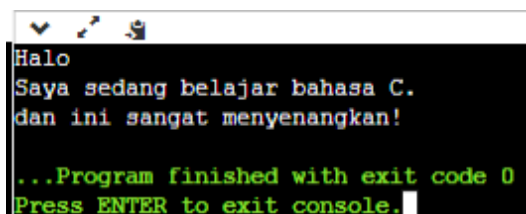
**Tabel 1.5:** Escape Sequence

| Escape sequence | Fungsi          |  |
|-----------------|-----------------|--|
| \a              | bell, alarm     |  |
| \b              | Backspace       |  |
| \f              | Ganti halaman   |  |
| \n              | Ganti baris     |  |
| \r              | Carriage return |  |
| \t              | tab horisontal  |  |
| \v              | tab vertikal    |  |
| \'              | Petik tunggal   |  |
| \"              | Petik Ganda     |  |
| \?              | Tanda tanya     |  |
| \\              | Backslash       |  |

**Contoh 1.6.1** Mencetak ke layar dan ganti baris menggunakan escape sequence `\n` untuk ganti baris.

```

1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Halo \nSaya sedang belajar bahasa C.\ndan ini
6         sangat menyenangkan!");
7     return 0;
8 }
```

**Gambar 1.10**

**Contoh 1.6.1** Mencetak kelayar dan ganti baris menggunakan escape sequence `\t` untuk mengatur tab.

```

1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Nama \t\t: Rahmad Rahardi\n");
5     printf("Alamat \t\t: Bendungan Hilir Jakarta\n");
6     printf("Tempat Lahir \t: Jakarta\n");
7     printf("Tanggal Lahir \t: 30 Pebruari 2000\n");
8
9     return (0);
10 }
```



A screenshot of a Windows console window with a black background and white text. The text displays the output of a program that has read user input using the scanf function. The output shows personal details: Name, Address, Birthplace, and Birthdate. At the end, it states the program finished successfully and prompts the user to press the Enter key to exit the console.

```
Nama      : Rahmad Rahardi
Alamat    : Bendungan Hilir Jakarta
Tempat Lahir : Jakarta
Tanggal Lahir : 30 Pebruari 2000

...Program finished with exit code 0
Press ENTER to exit console.
```

Gambar 1.11

### 1.6.2 Latihan

Cobalah buat suatu program yang dapat menerima input berupa nama dan NRP kemudian menampilkannya pada layar.

Halaman ini sengaja dikosongkan.

## Bab 2

# Instruksi Pemilihan

### 2.1 Tujuan

- Mahasiswa mengenal dan mampu menggunakan ekspresi-ekspresi logika dan perbandingan pada bahasa pemrograman C
- Mahasiswa mengenal dan mampu menggunakan syntax-syntax percabangan pada bahasa pemrograman C

### 2.2 Ekspresi Perbandingan dan Logika

#### 2.2.1 Ekspresi Perbandingan

Berikut adalah operator-operator yang digunakan pada suatu ekspresi perbandingan

**Tabel 2.1:** Operator Perbandingan

| Operator | Nama                                   | Contoh Ekspresi |
|----------|--|-----------------|
| ==       | Sama Dengan                            | x == y          |
| !=       | Tidak Sama Dengan                      | x != y          |
| >        | Lebih Besar                            | x > y           |
| <        | Kurang Dari                            | x < y           |
| >=       | Lebih besar sama atau sama dengan dari | x >= y          |
| <=       | kurang atau sama dengan dari           | x <= y          |

Suatu ekspresi perbandingan akan mengembalikan nilai berupa **true** atau **false** yang ditandai dengan nilai 0 atau 1. Sebagai contoh:

```
printf("%d",0>1); // akan mengoutputkan angka 0 ke layar  
printf("%d",0<1); // akan mengoutputkan angka 1 ke layar
```

#### 2.2.2 Ekspresi Logika

Berikut adalah operator-operator logika yang digunakan pada suatu ekspresi logika

**Tabel 2.2:** Operator Perbandingan

| Operator | Nama | Contoh Ekspresi            |
|----------|------|----------------------------|
| &&       | AND  | $x < 5 \ \&\& \ x < 10$    |
|          | OR   | $x < 5 \    \ x < 4$       |
| !        | NOT  | $!(x < 5 \ \&\& \ x < 10)$ |

Sama seperti ekspresi perbandingan, ekspresi logika akan mengembalikan nilai berupa true atau false

## 2.3 Statement if

statement if digunakan untuk menentukan blok kode C yang dijalankan apabila ekspresi kondisi bernilai benar (TRUE),

```
//blok kode sebelum if
if (Kondisi)
{
    // blok kode yang akan dijalankan ketika Kondisi bernilai benar.
}
// blok kode setelah if
```

Sebagai contoh, perhatikan program berikut

**Listing 2.1:** Contoh Penggunaan statement if

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya, hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
9
10     if (uangSaya >= hargaRoti)
11     {
12         printf("saya bisa beli roti\n");
13     }
14     printf("hehe");
15     return 0;
16 }
```

Output dari program ini adalah

```
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=10000` maka output dari program ini akan menjadi

```
saya bisa beli roti
hehe
```

## 2.4 Statement if-else

Statement else digunakan untuk menentukan blok kode yang di jalankan apabila kondisi salah.

```
//blok kode sebelum if
if (Kondisi)
{
// Blok kode yang dijalankan apabila kondisi benar.
} else
{
// Blok kode yang dijalankan apabila kondisi salah.
}
// blok kode setelah if
```

Berikut contoh penggunaan if-else

**Listing 2.2:** Contoh Penggunaan statement if-else

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya ,hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
9
10     if (uangSaya>=hargaRoti)
11     {
12         printf("saya bisa beli roti\n");
13     }
14     else
15     {
16         printf("saya tidak bisa beli roti\n");
17     }
18     printf("hehe");
19     return 0;
20 }
```

Output dari program ini adalah

```
saya tidak bisa beli roti
hehe
```

Jika baris ke 7 diganti dengan uangSaya=10000 maka output dari program ini akan menjadi

```
saya bisa beli roti
hehe
```

## 2.5 Statement if-else if

Statement `else if` digunakan untuk menjalankan blok kode apabila kondisi statement `if` atau `else if` sebelumnya bernilai salah.

```
// blok kode sebelum if
if (Kondisi1)
{
    /* blok kode yang akan dijalankan ketika
    Kondisi1 bernilai benar*/
}
else if (Kondisi2)
{
    /* blok kode yang akan dijalankan ketika Kondisi1 bernilai
    salah dan Kondisi2 bernilai benar */
}
else if (Kondisi3)
{
    /* blok kode yang akan dijalankan ketika
    Kondisi1 dan Kondisi2 bernilai
    salah dan Kondisi3 bernilai benar */
}
...
else if (KondisiN)
{
    /* blok kode yang akan dijalankan ketika
    Kondisi1 hingga KondisiN-1 bernilai
    salah dan KondisiN bernilai benar */
}
else
{
    /* blok kode yang akan dijalankan ketika
    Kondisi1 hingga KondisiN bernilai salah */
}
// blok kode setelah if
```

Berikut contoh penggunaan `if-else if`

**Listing 2.3:** Contoh Penggunaan statement `if-else if`

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya, hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
9
10     if (uangSaya > hargaRoti)
```

```
11 {  
12     printf("saya bisa beli roti\n");  
13 }  
14 else if(uangSaya==hargaRoti)  
15 {  
16     printf("saya bisa beli roti tapi uang saya akan  
langsung habis\n");  
17 }  
18 else  
19 {  
20     printf("saya tidak bisa beli roti\n");  
21 }  
22 printf("hehe");  
23 return 0;  
24 }
```

Output dari program ini adalah

```
saya tidak bisa beli roti  
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=10000` maka output dari program ini akan menjadi

```
saya bisa beli roti tapi uang saya akan langsung habis  
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=12000` maka output dari program ini akan menjadi

```
saya bisa beli roti  
hehe
```

## 2.6 Nested if

nested if merupakan konsep di mana di dalam suatu blok if terdapat statement if.

```
//blok kode sebelum if  
if (Kondisi)  
{  
    if (Kondisi2)  
    {  
        // do something  
    }  
    else  
    {  
        // do another thing  
    }  
}  
else
```

```
{  
    // do something else  
}
```

Berikut contoh penggunaan nested if

**Listing 2.4:** Contoh Penggunaan nested if

```
1  include <stdio.h>  
2  
3  int main()  
4  {  
5      //Deklarasi variabel  
6      int uangSaya ,hargaRoti ,uangTeman;  
7      uangSaya = 5000;  
8      hargaRoti = 10000;  
9      uangTeman = 42069;  
10  
11  
12     if (uangSaya>hargaRoti)  
13     {  
14         printf("saya bisa beli roti\n");  
15     }  
16     else if(uangSaya==hargaRoti)  
17     {  
18         printf("saya bisa beli roti tapi uang saya akan  
19         langsung habis\n");  
20     }  
21     else  
22     {  
23         if(uangTeman+uangSaya >= hargaRoti)  
24         {  
25             printf("saya bisa beli roti jika meminjam uang  
26             teman\n");  
27         }  
28         else  
29         {  
30             printf("saya tidak bisa beli roti\n");  
31         }  
32     }  
33     printf("hehe");  
34     return 0;  
35 }
```

## 2.7 Latihan

Coba buat program yang menerima input 3 buah bilangan bulat A, B, dan C. Outputkanlah 3 bilangan bulat itu ke layar dengan urutan paling kecil ke paling besar. Lakukanlah ini dengan menggunakan statement if, if else, if else if, atau nested if.



# Bab 3

## Perulangan dan Array

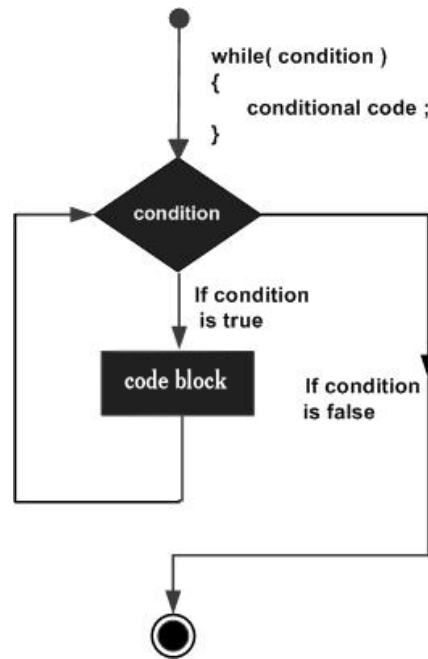
### 3.1 Tujuan

- Mahasiswa dapat mengenal dan menggunakan perulangan while pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan perulangan do-while pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan perulangan for pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan array dimensi satu maupun multidimensi.
- Mahasiswa mampu memanfaatkan perulangan untuk mengolah data pada array.

### 3.2 Perulangan

#### 3.2.1 while loop

Perulangan while akan menjalankan blok kode yang berada di dalamnya selama kondisi perulangan masih bernilai benar.



Gambar 3.1: Diagram Alir dari While

Syntaxnya pada bahasa C adalah sebagai berikut:

```

while(Kondisi)
{
    // blok kode yang akan diulang-ulang
}
  
```

Sebagai contoh, perhatikan kode berikut

Listing 3.1: Contoh Penggunaan while

```

1 int main()
2 {
3     int uangSaya, hargaRoti;
4     uangSaya = 10000;
5     hargaRoti = 2000;
6     while(uangSaya >= hargaRoti)
7     {
8         printf("Beli roti 1, uang saya sisa %d", uangSaya -
9             hargaRoti);
10        uangSaya -= hargaRoti;
11    }
12    printf("Uang saya tidak cukup lagi");
13    return 0;
14 }
  
```

Output dari program pada Listing 3.1 adalah

```

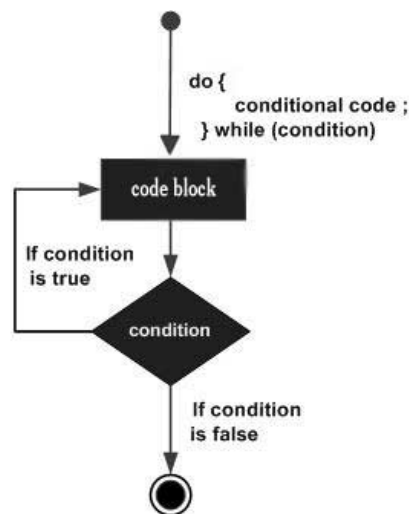
Beli roti 1, uang saya sisa 8000
Beli roti 1, uang saya sisa 6000
Beli roti 1, uang saya sisa 4000
Beli roti 1, uang saya sisa 2000
  
```

Beli roti 1, uang saya sisa 0  
 Uang saya tidak cukup lagi

Pada contoh ini, operasi pada baris 9 membuat variabel `uangSaya` berkurang 2000 pada setiap pengulangan hingga akhirnya nilai `uangSaya` tidak lebih dari atau sama dengan `hargaRoti` lagi.

### 3.2.2 do-while loop

do-while loop sebenarnya sama seperti while loop hanya saja do-while akan menjalankan perintah pada blok kode didalamnya terlebih dahulu sebelum melakukan pengecekan kondisi.



Gambar 3.2: Diagram Alir do-while

Syntaxnya pada bahasa C adalah sebagai berikut:

```
do{
    // blok kode yang akan diulang-ulang
}while(Kondisi)
```

Sebagai contoh, perhatikan kode berikut

Listing 3.2: Contoh Penggunaan do-while

```

1 int main()
2 {
3     int uangSaya, hargaRoti;
4     uangSaya = 10000;
5     hargaRoti = 12000;
6     do{
7         printf("Beli roti 1, uang saya sisa %d", uangSaya -
8             hargaRoti);
9         uangSaya -= hargaRoti;
10    }while(uangSaya >= hargaRoti)
11    printf("Uang saya tidak cukup lagi");
12    return 0;
13 }
```

Output dari program pada Listing 3.2 adalah

```
Beli roti 1, uang saya sisa -2000
Uang saya tidak cukup lagi
```

### 3.2.3 for loop

Misalkan terdapat blok kode while dengan bentuk seperti ini:

```
statementInisialisasi; // contoh: int i = 0;
while(kondisi){
    // do something
    statementUpdate; // contoh: i++
}
```

ini akan setara dengan

```
for(statementInisialisasi;kondisi;statementUpdate){
    // do something
}
```

Sebagai contoh, perhatikan program berikut:

**Listing 3.3:** Contoh Penggunaan for

```
1 int main()
2 {
3     int i=0;
4     for(i=1;i<10;i++){
5         printf("%d ",i);
6     }
7     return 0;
8 }
```

Output dari program ini adalah

```
1 2 3 4 5 6 7 8 9
```

Berikut kode pada Listing 3.3 jika diubah menjadi bentuk while-loop

**Listing 3.4:** For dalam bentuk while

```
1 int main()
2 {
3     int i=0;
4     i=1;
5     while(i<10){
6         printf("%d ",i);
7         i++;
8     }
9     return 0;
10 }
```

## 3.3 Array

Array atau biasa disebut larik adalah koleksi data dimana setiap elemen mempunyai nama yang sama dan bertipe sama. Setiap elemen diakses berdasarkan indeks elemennya.

### 3.3.1 Array 1D

Variabel array dimensi satu dideklarasikan dengan menentukan jenis elemen dan jumlah elemen yang di perlukan oleh array.

Sintaks :

```
TipeData NamaVariabel [UkuranArray ];
```

1. TipeData.  
Jenis elemen data elemen array :float,int,char dsb
2. NamaVariabel  
Namariabel mengikuti aturan pemberian nama variabel,
3. UkuranArray  
konstanta integer lebih besar dari 0.

Untuk menginisialisasi array dimensi satu, dapat dilakukan dengan cara seperti berikut:

```
int contoh_array[5] = {4,2,0,6,9};
```

Data di dalam array dapat akses dengan menggunakan suatu bilangan yang merupakan index dari array tersebut. Perhatikan potongan kode berikut.

**Listing 3.5:** Contoh Mengakses Array 1D

```
1 int main()  
2 {  
3     int arr[5] = {4,2,0,6,9};  
4     printf("%d\n",arr[0]);  
5     printf("%d\n",arr[4]);  
6     int i = 0;  
7     printf("%d\n",arr[i]);  
8     for(i=0;i<5;i++)  
9         printf("%d",arr[i]);  
10 }
```

Potongan kode pada Listing 3.5 akan memberikan output

```
4  
9  
4  
42069
```

### 3.3.2 Array 2D dan Array Multidimensi lainnya

Array dimensi dua pada dasarnya hanya merupakan array dimensi satu dari array dimensi satu. Oleh karena itu, untuk mendeklarasikan array dimensi dua kita dapat menggunakan syntax seperti berikut.

```
TipeData namaVariabel[UkuranArray1][UkuranArray2];
```

Hal ini berlaku juga untuk array dengan dimensi lebih dari dua.

```
TipeData namaVariabel[ukuranArray1]...[ukuranArrayN];
```

Banyaknya elemen yang akan dihasilkan dari deklarasi array seperti diatas adalah  $ukuranArray_1 \times ukuranArray_2 \times \dots \times ukuranArray_n$

Untuk menginisialisasi suatu array multidimensi dapat dilakukan sama seperti array biasa:

```
int arr[2][2] = {{1,2},{3,4}};
```

## 3.4 Contoh Aplikasi Perulangan dan Array

### 3.4.1 Linear Search

Linear Search adalah teknik untuk mencari suatu elemen dengan mengunjungi secara berurutan tiap - tiap elemen. Berikut adalah contoh mencari bilangan terbesar pada suatu array dengan menggunakan linear search

**Listing 3.6:** Mencari Bilangan Terbesar

```
1 int main()
2 {
3     int arr[5] = {4,2,0,6,9};
4     int currentMax = arr[0];
5     int i;
6     for(i=0;i<5;i++){
7         if(currentMax < arr[i]){
8             currentMax = arr[i];
9         }
10    }
11    printf("Bilangan terbesar adalah %d",currentMax);
12 }
```

### 3.4.2 Bubble Sort

Bubble Sort adalah suatu algoritma untuk mengurutkan data. Berikut adalah implementasinya pada bahasa C.

**Listing 3.7:** Bubble Sort

```
1 int main()
2 {
3     int arr[5] = {4,2,0,6,9};
4     int tmp;
```

```
5  for(int i=0;i<5;i++){
6      for(int j = 0; j < 5-i-1;j++){
7          if(arr[j]>arr[j+1]){
8              tmp = arr[j];
9              arr[j] = arr[j+1];
10             arr[j+1] = tmp;
11         }
12     }
13 }
14
15 for(int i=0;i<5;i++){
16     printf("%d ",arr[i]);
17 }
18 }
```

## 3.5 Latihan

- Cobalah inisialisasi suatu array multidimensi dengan menggunakan perulangan for.
- Buatlah suatu program untuk mengisi data pada suatu array berdasarkan input dari keyboard.
- Apakah yang akan terjadi jika suatu array `arr` diakses dengan `arr[-1]`?
- Apakah yang akan terjadi jika suatu array `arr` dengan ukuran 5 diakses dengan `arr[5]`?
- Perhatikan potongan kode berikut

```
for(i=0;i<10;i++){
    for(j=i;j<10;j++){
        printf("A");
    }
}
```

Ada berapa banyak huruf A yang akan muncul pada layar jika program tersebut dijalankan?

Halaman ini sengaja dikosongkan.



# Bab 4

## Fungsi (Subprogram)

### 4.1 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.

Keuntungan fungsi dalam c adalah :

- Dengan menggunakan fungsi maka kode program yang sama dapat digunakan berkali-kali.
- We can call C functions any number of times in a program and from any place in a program.
- Program c yang besar dapat dibagi ke dalam beberapa fungsi sehingga dapat dengan mudah untuk dilacak.
- Dapat digunakan kembali.

### 4.2 Deklarasi fungsi

Setiap program C memiliki setidaknya satu fungsi, yaitu main(), dan semua program yang paling dapat mendefinisikan fungsi tambahan.

Sintaks :

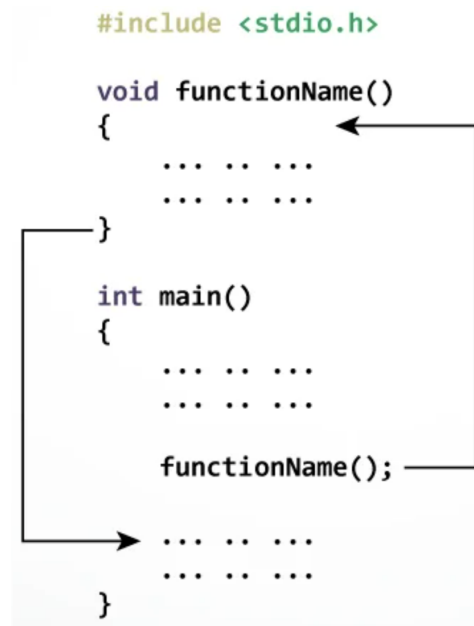
```
return_type Nama_Fungsi( Daftar parameter){  
    //badan fungsi  
    return sesuatu;  
}
```

- Return Type.  
Tipe data yang akan dikembalikan fungsi.
- Nama\_Fungsi.  
Nama fungsi.
- Parameters.  
Nilai atau argumen yang menjadi input parameter. Urutan nilai yang dimasukkan ke fungsi berurutan sesuai dengan parameter yang dimasukkan ke fungsi.
- Badan fungsi.  
Kumpulan statemen yang mendefinisikan apa yang dilakukan oleh fungsi.
- `return sesuatu;`  
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 LuasSegitiga(float Alas, float Tinggi)
2 {
3     float Luas;
4     Luas = 0.5*Alas*Tinggi;
5     return Luas;
6 }
```

## 4.3 Memanggil Fungsi



Gambar 4.1

```

1  #include <stdio.h>
2  // Mendeklarasikan fungsi luasSegitiga
3  // Parameter input ALas , dan Tingtgi
4  // Output float
5  float LuasSegitiga(float Alas, float Tinggi)
6  {
7      float Luas;
8      Luas = 0.5*Alas*Tinggi;
9      return Luas;
10 }
11 int main()
12 {
13     float Al = 4,Tg=10,L;
14     //Memanggil fungsi LuasSegitiga
15     L=LuasSegitiga(Al,Tg);
16     printf("Luas segitiga = %f",L);
17     return 0;
18 }

```

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

- Dua parameter input :  
input `Alas` dan `Tinggi` dengan tipe data `float`.
- Satu output dengan tipe data `float`

## 4.4 Fungsi Dengan Argumen

### 4.4.1 Argumen

#### 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. Parameter Formal :

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

#### 3. Parameter Aktual :

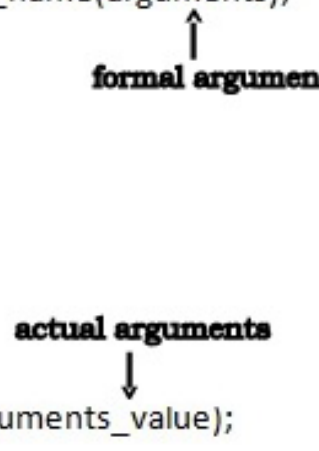
- (a) Parameter yang Ditulis ketika memanggil fungsi
- (b) Dapat berupa angka, ekspresi, atau bahkan panggilan fungsi.

```

#include <stdio.h>
return_type func_name(arguments);
{
    .....
    .....
}

Int main()
{
    .....
    func_name(arguments_value);
    .....
    return 0;
}

```



Gambar 4.2

### 4.4.2 Passing Parameter

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

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

#### 4.4.2.1 Passing Parameter by Value

Listing 4.1: Passing by Value

```
1 #include <stdio>
2 int tukarDanKembalikanJumlahnya(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 jumlah = tukarDanKembalikanJumlahnya(a,b);
14     printf("jumlah: %d\n", jumlah);
15     printf("nilai a dan b sekarang:\n");
16     printf("a: %d\n", a);
17     printf("b: %d\n", b);
18 }
```

Perhatikan potongan kode pada Listing 4.1. Baris 3-6 dari kode tersebut adalah operasi untuk menukar nilai dari 2 variabel. Namun, apabila program tersebut dijalankan, maka akan muncul output

```
jumlah: 3
nilai a dan be sekarang:
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.

#### 4.4.2.2 Passing Parameter by Reference

Perhatikan baris 2 pada potongan kode berikut:

**Listing 4.2:** Passing by Reference

```

1 #include <stdio>
2 int tukarDanKembalikanJumlahnya(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 jumlah = tukarDanKembalikanJumlahnya(a,b);
14     printf("jumlah: %d\n", jumlah);
15     printf("nilai a dan b sekarang:\n");
16     printf("a: %d\n", a);
17     printf("b: %d\n", b);
18 }

```

apabila program tersebut dijalankan, maka akan muncul output

```

jumlah: 3
nilai a dan be sekarang:
a: 2
b: 1

```

Ketika fungsi `tukarDanKembalikanJumlahnya(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 `tukarDanKembalikanJumlahnya(1,2)` tidak bisa dilakukan karena angka 1 dan 2 bukan variabel dan tidak memiliki alamat memori.

## 4.5 Rekursi

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

**Listing 4.3:** Contoh Rekursi Faktorial

```

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

```

Dapat dilihat bahwa fungsi faktorial pada function bodynya memanggil faktorial pada baris 4. Pada awalnya jika fungsi *faktorial(n)* dipanggil maka dia

akan mencoba untuk mengembalikan  $n \times \text{faktorial}(n - 1)$ ,  $\text{faktorial}(n - 1)$  akan mengembalikan  $(n - 1) \times \text{faktorial}(n - 1 - 1)$ , sehingga:

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

## 4.6 Latihan

- Buatlah fungsi yang dapat menerima 2 buah bilangan bulat a dan b kemudian mengembalikan nilai dari  $a^b$
- Buatlah program dengan algoritma bubble sort tetapi proses penukaran 2 elemen pada array dilakukan dengan fungsi.
- Masalah-masalah apa yang akan lebih mudah diselesaikan dengan menggunakan fungsi?