

## MODUL 7

### PROSEDUR 1



#### A. Tujuan Praktikum

1. Praktikan dapat mengenal prosedur dalam Dasar Pemrograman.
2. Praktikan dapat memahami konsep prosedur dalam Dasar Pemrograman.
3. Praktikan mampu untuk mengaplikasikan konsep prosedur dalam suatu program sesuai dengan kebutuhan.

#### B. Pendahuluan

Jika mendengar kata “prosedur”, tentunya sudah tidak asing di telinga teman-teman semua. Menurut wikipedia, prosedur adalah serangkaian aksi yang spesifik, tindakan atau operasi yang harus dijalankan atau dieksekusi dengan cara yang baku (sama) agar selalu memperoleh hasil yang sama dari keadaan yang sama.

Dalam sebuah prosedur akan menyimpan langkah / tata cara untuk membuat suatu hal yang diinginkan sesuai dengan judul prosedur. Contohnya ketika kita membaca prosedur membuat hot dog, maka prosedur tersebut akan berisi langkah-langkah secara runtut dan lengkap untuk membuat hot dog dari 0 hingga menjadi produk jadi. Langkah-langkah tersebut sudah tersimpan secara rapi dan terstruktur di dalam judul prosedur “Cara

Membuat Hot Dog”. Sewaktu-waktu ketika ingin membuat hot dog lagi, kalian tinggal mencari judul prosedur “Cara Membuat Hot Dog” dan di dalamnya kalian tinggal membaca & mengikuti langkah-langkah yang sudah disediakan.

Sama halnya dalam dunia pemrograman, konsep prosedur digunakan supaya programmer tidak perlu menuliskan code secara berulang di dalam main programnya, terutama jika code tersebut panjang & rumit. Programmer hanya perlu memanggil prosedur yang sudah dibuat dan meletakkannya di main program. Mungkin sebagai gambaran, kalian bisa melihat meme di bawah ini.



Terlihat bahwa ada contoh code visualisasi linked list yang panjang dan ribet. Ketika code sepanjang itu secara repetitif kita copy paste di main program, tentu akan terlihat sangat berantakan dan tidak efisien. Teman-teman praktikan yang hendak membaca code tentu juga akan bingung karena saking banyaknya code yang ada. Masalah tersebut dapat diselesaikan dengan mengimplementasikan konsep prosedur. Teman-teman tinggal memanggil judul prosedur “visualisasi(l)” ke dalam main program, code tetap berjalan, main program enak dipandang, dan problem solved.

## C. Pengertian Prosedur

Dalam dunia pemrograman, prosedur adalah **sebuah sub program yang berisi segala alur dan logika** untuk menjalankan suatu code tertentu. Prosedur akan membuat **program bisa membagi tugas-tugasnya** sesuai dengan keperluannya dan bisa kita panggil ke main program jika dibutuhkan. Penggunaan prosedur juga akan membuat program teman-teman menjadi jauh lebih **rapi, efisien, dan hemat memori**.

Prosedur akan sangat berguna untuk para programmer jika ingin menjalankan suatu code yang **berulang** maupun **tidak berulang**. Kita tidak perlu menulis ulang code tersebut karena kita hanya perlu memanggil prosedur itu saja untuk menjalankan code yang sama.

### • Struktur Prosedur

Prosedur terdiri dari 3 bagian penting, yaitu :

```
void namaProsedur(parameter){
    body
}
```

1. **Nama Prosedur** = nama untuk mengidentifikasi dan memanggil prosedur (harus unik)

2. **Parameter** = Untuk memberikan inputan supaya bisa digunakan di dalam prosedur (opsional)

3. **Body** = Isi dari prosedur berupa alur dan logika dari suatu code

### • Contoh Penggunaan Prosedur

```
void hitungLuasPersegi(int *LuasPersegi, int panjang, int lebar);
void hitungLuasSegitiga(int *LuasSegitiga, int alas, int tinggi);
void tampilData(int LuasPersegi, int LuasSegitiga);

int main(int argc, char const *argv[]){
    int luasPersegi, luasSegitiga, panjang, lebar, alas, tinggi;

    //Hitung Luas Persegi
    panjang = 10;
    lebar = 5;
    hitungLuasPersegi(&luasPersegi, panjang, lebar);

    //Hitung Luas Segitiga
    hitungLuasSegitiga(&luasSegitiga, 10, 3);

    //Print Data Luas
    tampilData(luasPersegi, luasSegitiga);
    return 0;
}

void hitungLuasPersegi(int *LuasPersegi, int panjang, int lebar){
    *LuasPersegi = panjang * lebar;
}

void hitungLuasSegitiga(int *LuasSegitiga, int alas, int tinggi){
    *LuasSegitiga = (alas * tinggi) / 2;
}

void tampilData(int LuasPersegi, int LuasSegitiga){
    printf("\n\t ==[TAMPAK DATA] ==\n");
    printf("Luas Persegi    : %d", LuasPersegi);
    printf("\nLuas Segitiga : %d", LuasSegitiga);
}
```

Deklarasi Prosedur

Pemanggilan Prosedur

Pendefinisian Prosedur

- Deklarasi Prosedur

```
void <namaProsedur> (<parameter1, parameter2, ....>);
```

Pendeklarasian prosedur biasa dilakukan di **file header**. Namun, dikarenakan pada dasar pemrograman belum diajarkan konsep ADT, maka pendeklarasian prosedur **dapat dilakukan di bagian margin atas pada program** (bagian sebelum masuk ke main program).

```
main.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar);
5  void hitungLuasSegitiga(int *luasSegitiga, int alas, int tinggi);
6  void tampilData(int luasPersegi, int luasSegitiga);
7
8  int main(int argc, char const *argv[]){
9      int luasPersegi, luasSegitiga, panjang, lebar, alas, tinggi;
10
11      //Hitung Luas Persegi
12      panjang = 10;
13      lebar = 5;
14      hitungLuasPersegi(&luasPersegi, panjang, lebar);
15
16      //Hitung Luas Segitiga
17      hitungLuasSegitiga(&luasSegitiga, 10, 3);
18
19      //Print Data Luas
20      tampilData(luasPersegi, luasSegitiga);
21      return 0;
22  }
23
24  void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
25      *luasPersegi = panjang * lebar;
26  }
27
28  void hitungLuasSegitiga(int *luasSegitiga, int alas, int tinggi){
29      *luasSegitiga = (alas * tinggi) / 2;
30  }
31
32  void tampilData(int luasPersegi, int luasSegitiga){
33      printf("\n\t ===[TAMPIL DATA] ===\n");
34      printf("Luas Persegi      : %d", luasPersegi);
35      printf("\nLuas Segitiga : %d", luasSegitiga);
36  }
```

Terletak Pada Bagian  
Margin Atas Program,  
Tepat Sebelum Masuk Ke  
Main Program

Sebuah prosedur akan tetap berjalan meskipun tidak dideklarasikan. Namun, hal tersebut akan memicu **warning conflicting types** karena program tidak menemukan body dari prosedur terkait saat dieksekusi.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char const *argv[]){
5      int luasPersegi, luasSegitiga, panjang, lebar, alas, tinggi;
6
7      //Hitung Luas Persegi
8      panjang = 10;
9      lebar = 5;
10     hitungLuasPersegi(&luasPersegi, panjang, lebar);
11
12     //Hitung Luas Segitiga
13     hitungLuasSegitiga(&luasSegitiga, 10, 3);
14
15     //Print Data Luas
16     tampilData(luasPersegi, luasSegitiga);
17     return 0;
18 }
19
20 void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
21     *luasPersegi = panjang * lebar;
22 }
23
24 void hitungLuasSegitiga(int *luasSegitiga, int alas, int tinggi){
25     *luasSegitiga = (alas * tinggi) / 2;
26 }
27
28 void tampilData(int luasPersegi, int luasSegitiga){
29     printf("\n\t ===[TAMPIL DATA] ===\n");
30     printf("Luas Persegi      : %d", luasPersegi);
31     printf("\nLuas Segitiga : %d", luasSegitiga);
32 }
33

```

Tidak Ada Deklarasi Pada Margin Atas Program

Muncul Warning Pada Compiler

Line	Col	File	Message
21	6	C:\Users\Eric Daniswara\Documents\Modul Prosedur Da...	[Warning] conflicting types for 'hitungLuasPersegi'
10	5	C:\Users\Eric Daniswara\Documents\Modul Prosedur Da...	[Note] previous implicit declaration of 'hitungLuasPersegi' was here
25	6	C:\Users\Eric Daniswara\Documents\Modul Prosedur Da...	[Warning] conflicting types for 'hitungLuasSegitiga'
13	5	C:\Users\Eric Daniswara\Documents\Modul Prosedur Da...	[Note] previous implicit declaration of 'hitungLuasSegitiga' was here

Line: 9 Col: 15 Sel: 54 Lines: 33 Length: 850 Insert Done parsing in 0.016 seconds

Bisa terlihat pada gambar program di atas, tidak ada pendeklarasian prosedur di bagian margin atas program. Program akan tetap berjalan dengan baik tanpa error. Prosedur juga akan tetap berjalan dengan benar. Namun, jika teman-teman perhatikan di bagian compiler akan muncul banyak tulisan berwarna oranye yang berisi warning. Hal tersebut dapat dihilangkan dengan melakukan pendeklarasian terlebih dahulu.

Sebelum masuk lebih lanjut, teman-teman harus tahu bahwa ada 2 konteks dalam prosedur. Konteks tersebut antara lain pendefinisian prosedur dan pemanggilan prosedur. Pendefinisian prosedur merupakan proses dimana programmer membuat suatu prosedur hingga siap digunakan di main program. Lalu untuk pemanggilan prosedur merupakan proses programmer menggunakan / memanggil prosedur yang telah ia buat baik di main program atau pun di prosedur lain.

- **Pendefinisian Prosedur**

```
void <namaProsedur> (<parameter1, parameter2, ...>){
    <body> //Berisi code yang menjalankan suatu aksi
           //ketika prosedur ini dipanggil
}
```

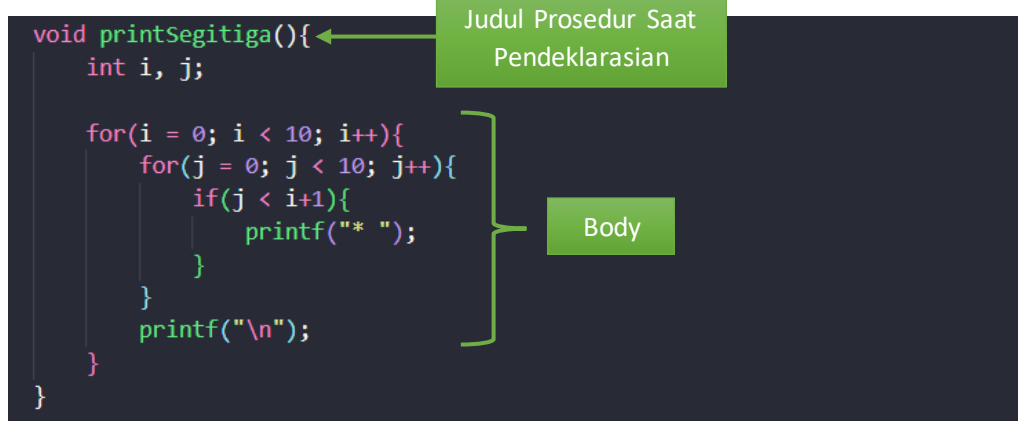
Bagian pendefinisian prosedur merupakan hal yang sangat penting selama penerapan konsep prosedur. Bagian ini akan **mendefinisikan prosedur terkait dengan perintah / alur code tertentu sesuai dengan konteks prosedur** yang dibuat.

Pendefinisian prosedur biasa dilakukan di **file source**. Namun dikarenakan pada dasar pemrograman masih belum diajarkan konsep ADT, maka pembuatan pendefinisian prosedur **bisa dilakukan di bagian margin bawah pada program (tepat di bawah main program)**

```
main.c > ...
1  ✓ #include <stdio.h>
2  ✓ #include <stdlib.h>
3
4  void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar);
5  void hitungLuasSegitiga(int *luasSegitiga, int alas, int tinggi);
6  void tampilData(int luasPersegi, int luasSegitiga);
7
8  int main(int argc, char const *argv[]){
9      int luasPersegi;
10     int panjang = 10;
11     int lebar = 5;
12
13     hitungLuasPersegi(&luasPersegi, panjang, lebar);
14
15     return 0;
16 }
17
18 void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
19     *luasPersegi = panjang * lebar;
20 }
21
22 void hitungLuasSegitiga(int *luasSegitiga, int alas, int tinggi){
23     *luasSegitiga = (alas * tinggi) / 2;
24 }
25
26 void tampilData(int luasPersegi, int luasSegitiga){
27     printf("\n\t ===[TAMPIL DATA] ===\n");
28     printf("Luas Persegi    : %d", luasPersegi);
29     printf("\nLuas Segitiga : %d", luasSegitiga);
30 }
```

Terletak Pada Bagian  
Margin Bawah Program,  
Tepat Setelah Main  
Program Selesai

Contoh 1 :



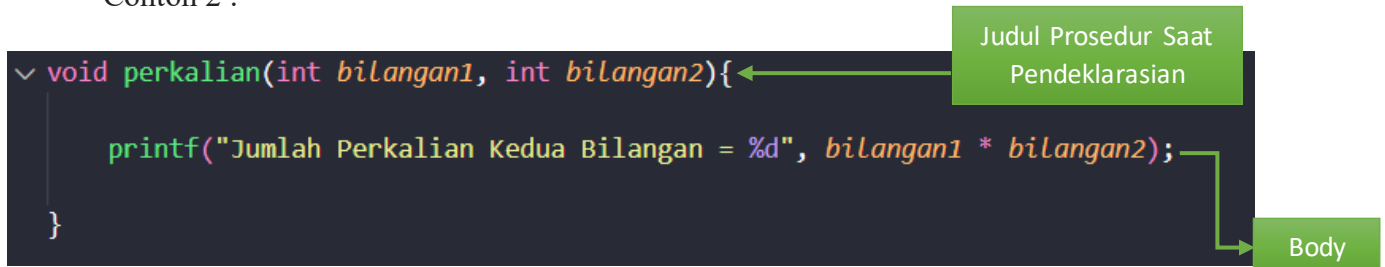
```
void printSegitiga(){
    int i, j;

    for(i = 0; i < 10; i++){
        for(j = 0; j < 10; j++){
            if(j < i+1){
                printf("* ");
            }
        }
        printf("\n");
    }
}
```

The diagram shows the function definition for `printSegitiga`. A green box labeled "Judul Prosedur Saat Pendeklarasian" points to the first line `void printSegitiga(){`. Another green box labeled "Body" points to the nested loop structure starting from `for(i = 0; i < 10; i++){` and ending with `}`.

Merupakan pendefinisian dari judul prosedur `printSegitiga`. Prosedur tersebut didefinisikan dengan alur code perulangan untuk mencetak pola yang berbentuk segitiga.

Contoh 2 :



```
void perkalian(int bilangan1, int bilangan2){
    printf("Jumlah Perkalian Kedua Bilangan = %d", bilangan1 * bilangan2);
}
```

The diagram shows the function definition for `perkalian`. A green box labeled "Judul Prosedur Saat Pendeklarasian" points to the first line `void perkalian(int bilangan1, int bilangan2){`. Another green box labeled "Body" points to the `printf` statement inside the function.

Merupakan pendefinisian dari judul prosedur `perkalian`. Prosedur tersebut didefinisikan untuk menampilkan hasil perkalian dari `bilangan1` dan `bilangan2`.

- **Pemanggilan Prosedur**

```
<namaProsedur>(<parameter tanpa tipe data>);
```

Setelah kita mendeklarasikan dan mendefinisikan prosedur, prosedur tentu tidak akan berguna jika tidak dipanggil pada main program. Tentunya prosedur harus kita panggil ke dalam main program supaya perintah di dalam prosedur tersebut **dapat dieksekusi oleh main program atau oleh prosedur lain**.

**Syarat Pemanggilan Prosedur :**

1. **Nama Prosedur** yang dipanggil **harus sama** dengan nama Prosedur yang telah di deklarasikan dan didefinisikan.
2. **Jumlah Parameternya harus sama** dengan prosedur yang dipanggil.
3. **Tipe data variabel** pada parameter yang ada di main program harus sama dan **sesuai dengan penempatannya** dengan tipe data variabel pada **parameter** pada saat prosedur dideklarasikan dan didefinisikan.
4. **Jika Parameter ditempel asteris / pointer (\*)** saat pendeklarasian & pendefinisian, maka saat pemanggilan di parameter tersebut **harus ditempel dengan "&"**.

Contoh pemanggilan prosedur:

```
int main(int argc, char const *argv[]){
    int luasPersegi;
    int panjang = 10;
    int lebar = 5;

    hitungLuasPersegi(&luasPersegi, panjang, lebar);

    return 0;
}

void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
    *luasPersegi = panjang * lebar;
}
```

Memanggil

Parameter yang ditempel "\*", saat pemanggilan wajib ditempel "&"

**NOTE**

“**NAMA VARIABEL** pada parameter ketika memanggil prosedur dengan nama variabel pada parameter yang melekat pada prosedur **TIDAK HARUS SAMA**, tetapi **TIPE DATA VARIABEL HARUS SAMA**.”



Contoh :

```

4 void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar);
5
6 int main(int argc, char const *argv[]){
7     int hasil;
8     int p = 10;
9     int l = 5;
10
11     hitungLuasPersegi(&hasil, p, l);
12
13     return 0;
14 }
15
16 void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
17     *luasPersegi = panjang * lebar;
18
19     printf("Luas : %d", *luasPersegi);
20 }
21

```

Bisa terlihat bahwa parameter saat pemanggilan dengan pendeklarasian / pendefinisian berbeda. Namun, selama tipe datanya sama, maka prosedur akan tetap berjalan sesuai dengan parameternya.

### • Parameter

Parameter merupakan **informasi / variabel** yang dibutuhkan untuk menjalankan aksi suatu prosedur. Seorang programmer harus bisa menentukan sekiranya informasi / variabel apa yang diperlukan supaya sebuah prosedur bisa berjalan sesuai aksi yang diinginkan. Ketika prosedur yang hendak dieksekusi ternyata **masih kekurangan parameter**, maka **program akan mengalami error dan tidak bisa dijalankan**.

Sekarang ibaratkan kita ingin membuat prosedur untuk menghitung luas segitiga. Maka kita harus paham bagaimana cara kita menghitung luas segitiga di dunia nyata.

$$\text{Luas} = (\text{Alas} \times \text{Tinggi}) / 2$$

Sesuai rumus di atas, maka bisa kita simpulkan bahwa kita punya informasi penting, yaitu **alas dan tinggi** sebagai **input**, serta **luas** sebagai **output**. Selanjutnya, kita tentukan kategori prosedur apa yang hendak kita pakai.

Jika kategori **Semi-Naive Input**, maka parameter hanya akan diisi oleh informasi yang berupa **input**. Jika kategori **Semi-Naive Output**, maka parameter akan diisi oleh informasi yang berupa **output**. Jika kategori **Nett Effect**, maka parameter akan diisi oleh informasi yang berupa **input dan output**.

Di dalam prosedur, bentuk parameter dapat berupa :

- 1) **Parameter input** = merupakan parameter **sebagai masukan** ke dalam prosedur untuk menjalankan suatu aksi
- 2) **Parameter output** = merupakan parameter di dalam prosedur yang **nilainya nanti akan dibawa keluar** ke dalam variabel yang ada di main program (memiliki **ciri khas**, yaitu **terdapat pointer / asteris “\*”** yang menempel pada parameter)

```
void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar);

int main(int argc, char const *argv[]){
    int luasPersegi;
    int panjang = 10;
    int lebar = 5;

    hitungLuasPersegi(&luasPersegi, panjang, lebar);
    printf("Luas      : %d", luasPersegi);

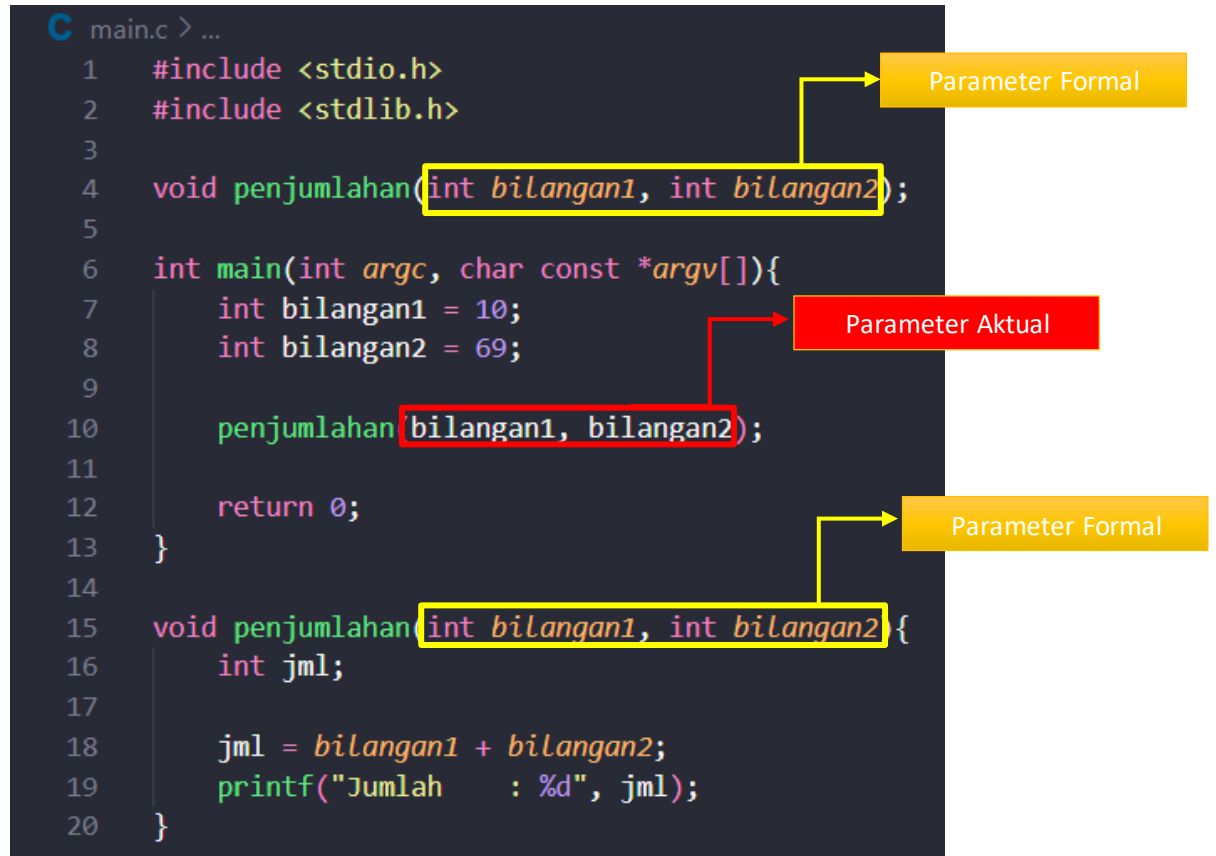
    return 0;
}

void hitungLuasPersegi(int *luasPersegi, int panjang, int lebar){
    *luasPersegi = panjang * lebar;
}
```

Hasil Akan Langsung Dibawa  
Keluar Ke Main program

### Jenis Parameter

Parameter pada prosedur dibedakan menjadi 2 jenis, yaitu **parameter Formal** dan **parameter Aktual**. **Parameter Formal** merupakan jenis parameter yang letaknya berada di luar main program. **Parameter Aktual** merupakan jenis parameter yang ada di dalam main program pada saat pemanggilan dilakukan.



Berikut adalah karakteristik dari Parameter Formal dan Parameter Aktual :

1. Parameter Formal dan Parameter Aktual **tidak harus** memiliki **nama yang sama**.
2. Parameter Formal dan Parameter Aktual harus memiliki **tipe data yang sama**.
3. **Jumlah** Parameter Aktual dan Parameter Formal **harus sama**.
4. **Urutan** Parameter Formal **harus sama** dengan Parameter Aktual.

- **Kategori Prosedur Berdasarkan Parameter**

1. **Naive**

Merupakan prosedur yang **tidak mempunyai parameter** sama sekali

```

void tampilMenu(){
    printf("\n\t ===[MENU UTAMA] ===\n");
    printf("\n[1]. Input");
    printf("\n[2]. Tampil Data");
    printf("\n[3]. Hapus Data");
    printf("\n\n[0]. Exit");
}
  
```

## 2. Semi-Naive Input

Merupakan prosedur yang **hanya memiliki parameter input** dan akan menghasilkan output yang dikeluarkan melalui Standard Input/Output. Pada kategori prosedur ini, **code inputan akan berada di main program**, sedangkan **code output akan berada di dalam prosedur**.

```
void perkaliar(int bilangan1, int bilangan2){
    int jml;

    jml = bilangan1 * bilangan2;
    printf("Jumlah Perkalian : %d", jml);
}
```

Hanya Ada  
Parameter Input

Code Printf Output  
Ada Di Dalam  
Prosedur

## 3. Semi-Naive Output

Merupakan prosedur yang **hanya memiliki parameter output** dan menggunakan input yang didapat melalui Standard Input/Output. Pada kategori prosedur ini, **code inputan akan berada di dalam prosedur**, sedangkan **code output berada di main program**.

```
void pengurangan(int *jml){
    int bil1, bil2;

    printf("\nInput Bilangan Pertama : "); scanf("%d", &bil1);
    printf("\nInput Bilangan Kedua : "); scanf("%d", &bil2);
    *jml = bil2 - bil1;
}
```

Hanya Ada  
Parameter Output

Code Printf &  
Scanf Input  
Ada Di Dalam  
Prosedur

## 4. Nett Effect

Merupakan prosedur yang tidak menggunakan Standard Input/Output. Kategori prosedur ini merupakan kategori prosedur Input/Output **yang paling direkomendasikan**. Nett Effect **tidak memperbolehkan** adanya **inputan dan output** berbentuk printf atau pun scanf **di dalam prosedur** (Segala jenis code input & output berada di main program).

```
void hitLuas(int *luas, int panjang, int lebar){
    *luas = panjang * lebar;
}
```

Terdiri Dari  
Parameter  
Input dan  
Output

Tidak Ada Code Printf &  
Scanf Input Atau Output Karena Sudah  
Dilakukan Di Main Program

- **Pointer / Asteris (\*)**

Seperti yang sudah disebutkan pada halaman sebelumnya, salah satu ciri khas dari parameter output adalah adanya **bintang atau asteris / pointer yang menempel pada parameter formal**. Penggunaan asteris / pointer akan membuat **variabel pada parameter aktual nilainya ikut berubah** menyesuaikan nilai dari variabel pada parameter formal.

Berbeda cerita ketika teman-teman **tidak menempelkan asteris / pointer** pada parameter formal, maka segala perubahan nilai pada parameter formal selama prosedur dieksekusi **tidak akan mengakibatkan perubahan pada parameter aktual**. Oleh karena itu, ketika teman-teman ingin **mengubah value suatu parameter** pada prosedur, **diwajibkan untuk menggunakan asteris / pointer** pada parameter formal terkait yang ingin datanya diubah.

```

C main.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void penjumlahanAsteris(int *jmlAsteris, int bil1, int bil2);
5  void penjumlahanBiasa(int jmlBiasa, int bil1, int bil2);
6
7
8  int main(int argc, char const *argv[]){
9      int jmlAsteris = 0;
10     int jmlBiasa = 0;
11     int bil1 = 10;
12     int bil2 = 5;
13
14     penjumlahanAsteris(&jmlAsteris, bil1, bil2);
15     penjumlahanBiasa(jmlBiasa, bil1, bil2);
16
17     printf("Penjumlahan Asteris : %d", jmlAsteris);
18     printf("\n\nPenjumlahan Biasa : %d", jmlBiasa);
19
20     return 0;
21 }
22
23 void penjumlahanAsteris(int *jmlAsteris, int bil1, int bil2){
24     *jmlAsteris = bil1 + bil2;
25 }
26
27 void penjumlahanBiasa(int jmlBiasa, int bil1, int bil2){
28     jmlBiasa = bil1 + bil2;
29 }

```

```

Penjumlahan Asteris : 15

```

```

Penjumlahan Biasa : 0

```

```

-----
Process exited after 0.0424 seconds with return value 0
Press any key to continue . . .

```

Supaya bisa lebih memahami terkait konsep pointer / asteris, teman-teman saya sarankan untuk mencoba code disamping dan melakukan modifikasi secara mandiri terkait pointer / asteris.

Pada hasil code tersebut, terlihat bahwa prosedur dengan asteris berhasil mengubah value dari parameter jmlAsteris. Hal ini disebabkan karena parameter tersebut ditempel oleh asteris, sehingga segala perubahan yang terjadi saat eksekusi prosedur terjadi akan mempengaruhi parameter aktual pada main program

Berbeda dengan prosedur tanpa asteris yang value dari parameter jmlBiasa tertampil tetap 0 karena tidak ditempel oleh asteris pada bagian parameter formalnya

- **Prosedur Rekrusif**

Ketika kalian sudah masuk ke modul prosedur, tentunya kalian sudah mengerti konsep perulangan di dalam bahasa pemrograman. Namun, tahukan kalian bahwa dengan prosedur kalian **bisa melakukan perulangan tanpa menerapkan konsep “do-while”, “while”, atau pun “nested loop”** ?

Seperti yang sudah disebutkan di atas, bahwa sebuah **prosedur bisa memanggil prosedur lain di dalam bodynya** saat kita melakukan pendefinisian. Nah, dengan memanfaatkan hal tersebut, kita bisa melakukan perulangan dengan memanggil kembali prosedur itu sendiri dan mengubah sedikit parameternya saat pemanggilan. Konsep tersebut disebut dengan konsep **prosedur rekrusif**.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void visualisasi(int alas, int tinggi, int i, int j);
5
6 int main(int argc, char *argv[]) {
7     visualisasi(10, 10, 0, 0);
8     return 0;
9 }
10
11 void visualisasi(int alas, int tinggi, int i, int j){
12
13     if(i > tinggi){ //kondisi berhenti
14         return;
15     }
16     else if(j > alas){ //kondisi ketika value j sudah lebih besar dari alas
17         printf("\n");
18         visualisasi(alas, tinggi, i+1, 0); //Memanggil kembali prosedur visualisasi
19                                     //hanya saja kita lakukan penambahan 1 pada parameter i
20                                     //dan ubah value dari parameter j menjadi 0
21                                     //modifikasi prosedur ini konsepnya sama dengan perulangan for(i = 0; i < tinggi; i++)
22     }
23     else{
24         if(j < i){
25             printf("* ");
26         }
27         visualisasi(alas, tinggi, i, j+1); //Memanggil kembali prosedur visualisasi
28                                     //hanya saja kita lakukan penambahan 1 pada parameter j
29                                     //modifikasi prosedur ini konsepnya sama dengan perulangan for(j = 0; j < alas; j++)
30     }
31 }

```

Code di atas merupakan contoh dari implementasi prosedur rekrusif untuk visualisasi segitiga. Bisa dilihat bahwa tidak ada code perulangan sama sekali di code tersebut. Namun, code tersebut akan tetap berjalan dengan semestinya dan berhasil melakukan visualisasi segitiga. Konsep rekrusif memanfaatkan “if-else” sebagai pemberhenti rekrusif dan pengkondisian suatu alur code yang diinginkan.

\*Note :

Q1 : “Kak, kok kayaknya rumit banget. Bingung nih belum paham.”

A1 : “Ya memang, ini sebenarnya materi semester 3. Namun, saya berikan pengantar disini karena di tahun kakak kemarin **konsep prosedur rekrusif ini dikeluarkan di UAS.**”

Q2 : “Kak, bukannya lebih ringkas dan simpel pakai perulangan biasa ya buat code di atas ?”

A3 : “Yups benar. Itu hanya contoh paling relate saat ini untuk penggunaan prosedur rekrusif. Konsep rekrusif ini sebenarnya lebih efektif digunakan untuk konsep Array of Record dan Struktur Data.”

Q3 : “Kak, ini keluar di UGD gak ya nanti ?”

A3 : “YNTKS”



## **GUIDED 1**

1. Pada guided nomor 1 ini, akan disediakan soal studi kasus yang menerapkan **semua 4 kategori prosedur** yang sudah dijelaskan di modul.
2. Buatlah file baru dengan penamaan **GD7\_X\_YYYYY\_1** (X = kelas; Y = 5 digit akhir NPM)
3. Berikut detail soal studi kasusnya :

Kak Pasha adalah seorang programmer pemula yang sedang belajar bahasa C. Saat ini, Kak Pasha sudah mulai masuk ke modul Prosedur 1. Di modul Prosedur 1, Kak Pasha paham bahwa ada 4 kategori prosedur dengan ciri khasnya masing-masing, antara lain **Naive**, **Semi-Naive Input**, **Semi-Naive Output**, dan **Nett Effect**. Dikarenakan Kak Pasha merasa belum cukup paham, ia kemudian membuat program yang menerapkan masing-masing dari kategori tersebut. Menu yang dibuat oleh Kak Pasha ada 5 (**Naive**) antara lain:

### **MENU 1 (HITUNG LUAS SEGITIGA)**

Menu untuk menghitung luas segitiga. Program akan meminta inputan alas dan tinggi. Program juga akan menampilkan error handling dan meminta inputan ulang jika inputan user kurang dari 0. Program akan langsung menampilkan hasil dari perhitungan luas setelah semua inputan user benar. (**Gunakan Prosedur Semi-Naive Input**)

### **MENU 2 (BELI BARANG)**

Menu untuk membeli barang antara beras dan jagung. Beras memiliki harga Rp10.000,00 dan jagung memiliki harga Rp7.000,00. Program akan meminta inputan nama barang dan jumlah barang. Ketika inputan tidak sesuai, maka akan menampilkan error handling dan meminta inputan ulang.

Setelah semua sesuai, harga akan secara otomatis terinput oleh user sesuai dengan nama barang yang diinput oleh user. Setelah semua sesuai, program akan menampilkan konfirmasi pembelian berhasil yang diikuti dengan nama barang, jumlah, dan nominal total harganya. (**Gunakan Prosedur Semi-Naive Output**)

### MENU 3 (TAMPIL KERANJANG)

Menu untuk menampilkan detail dari nama barang, jumlah, dan nominal total harga dari pembelian pengguna. Menu ini tidak akan bisa diakses jika keranjang masih kosong dan menampilkan error handling. **(Gunakan Prosedur Semi-Naive Input)**

### MENU 4 (HITUNG VOLUME BALOK)

Menu untuk menghitung volume balok. Program akan meminta inputan panjang, lebar, dan tinggi. Program akan menampilkan error handling & meminta inputan ulang jika inputan panjang, lebar, dan tinggi kurang dari 0. Setelah semua terinput dengan benar, program akan menghitung volumenya dan langsung menampilkan hasil dari penghitungan volume balok. **(Gunakan Prosedur Nett Effect)**

### MENU 0 (EXIT)

Tuliskan **nama lengkap, NPM, dan kelas praktikan.**

## Code

### Deklarasi Prosedur

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

typedef char string[100];

void tampilMenu(); //Pendeclarasian prosedur tampilMenu (Naive)
void hitungLuas(float alas, float tinggi); //Pedeclarasian prosedur hitungLuas (Semi-Naive Input)
void beliBarang(string *nama, int *jml, float *harga); //Pendeclarasian prosedur beliBarang (Semi-Naive Output)
void tampilKeranjang(string nama, int jml, float harga); //Pendeclarasian prosedur tampilKeranjang (Semi-Naive Input)
void hitVolumeBalok(int *hasil, int panjang, int lebar, int tinggi); //Pendeclarasian prosedur hitVolumeBalok (Nett Effect)
```



## Main Program

```

int main(int argc, char *argv[]){
    string nama = "-";
    int menu, p, l, tg, hasil;
    int jml = 0;
    float a, t;
    float harga = 0;

    do{
        tampilMenu();
        printf("\n\t>>> "); scanf("%d", &menu);

        switch(menu){
            case 1 :
                printf("\n\t\t=== [HITUNG LUAS] ===\n");
                printf("\n\tMasukkan Alas   : "); scanf("%f", &a);
                while(a <= 0){
                    printf("\n\t[!] Inputan Tidak Boleh Kurang dari 0");
                    printf("\n\tMasukkan Alas   : "); scanf("%f", &a);
                }
                printf("\n\tMasukkan Tinggi : "); scanf("%f", &t);
                while(t <= 0){
                    printf("\n\t[!] Inputan Tidak Boleh Kurang dari 0");
                    printf("\n\tMasukkan Tinggi : "); scanf("%f", &t);
                }
                hitungLuas(a, t); //Pemanggilan prosedur hitungLuas
                break;

            case 2 :
                printf("\n\t\t=== [BELI BARANG] ===\n");
                beliBarang(&nama, &jml, &harga); //pemanggilan prosedur beliBarang
                printf("\n\t[+] Berhasil Membeli Barang %s Dengan Jumlah %d Seharga Rp%.2f", nama, jml, harga);
                break;

            case 3 :
                if(jml !=0){
                    printf("\n\t\t=== [TAMPIL KERANJANG] ===\n");
                    tampilKeranjang(nama, jml, harga); //pemanggilan prosedur tampilKeranjang
                }else{
                    printf("\n\t[!] Belum Ada Barang Yang Dibeli");
                }
                break;

            case 4 :
                printf("\n\t\t=== [HITUNG VOLUME BALOK] ===\n");
                printf("\n\tMasukkan Panjang   : "); scanf("%d", &p);
                while(p <= 0){
                    printf("\n\t[!] Inputan Tidak Boleh Kurang dari 0");
                    printf("\n\tMasukkan Panjang   : "); scanf("%d", &p);
                }
                printf("\n\tMasukkan Lebar      : "); scanf("%d", &l);
                while(l <= 0){
                    printf("\n\t[!] Inputan Tidak Boleh Kurang dari 0");
                    printf("\n\tMasukkan Lebar      : "); scanf("%d", &l);
                }
                printf("\n\tMasukkan Tinggi     : "); scanf("%d", &tg);
                while(tg <= 0){
                    printf("\n\t[!] Inputan Tidak Boleh Kurang dari 0");
                    printf("\n\tMasukkan Tinggi     : "); scanf("%d", &tg);
                }
                hitVolumeBalok(&hasil, p, l, tg); //pemanggilan prosedur hitVolumeBalok
                printf("\n\t-----");
                printf("\n\tHasil Volume Balok : %d Cm^3", hasil);
                break;

            case 0 :
                printf("\n\t[NAMA LENGKAP - NPM - KELAS]");
                printf("\n\t[*] Prosedur Itu Mudah :) [*]");
                break;

            default :
                printf("\n\t[!] Menu Tidak Ada");
                break;
        }
        getch();
    }while(menu!=0);

    return 0;
}

```

## Pendefinisian Prosedur

## GUIDED 2

1. Pada guided nomor 2 ini, kalian akan diminta untuk berkreasi **membuat 1 program baru** yang menerapkan 4 kategori prosedur yang ada.
2. Buatlah file baru dengan penamaan **GD7\_X\_YYYYY\_2** (X = kelas; Y = 5 digit akhir NPM)
3. Buatlah satu program baru secara mandiri dengan ketentuan sebagai berikut :
  - a. Tema program bebas
  - b. Satu prosedur kategori Naive
  - c. Dua prosedur kategori Semi-Naive Input
  - d. Dua prosedur kategori Semi-Naive Output
  - e. Satu prosedur kategori Net Effect
4. Pastikan pada exit menu (menu 0) menampilkan **nama lengkap praktikan, NPM, dan kelas praktikum.**
5. Pastikan program bisa dicompile & berjalan dengan baik sesuai ketentuan
6. Tidak menggunakan prosedur sama sekali (-50)
7. Jumlah prosedur tidak memenuhi ketentuan (-10 tiap prosedur yang kurang)
8. Jika code tidak rapi akan dilakukan pengurangan nilai (-10)
9. **Tidak diperbolehkan** menggunakan materi di modul selanjutnya (fungsi, record, array)
10. **Tema tidak harus berbeda tiap mahasiswanya**, hanya saja yang namanya programming pasti **tiap orang akan memiliki logikanya tersendiri**, sehingga tidak mungkin ada mahasiswa yang codenya sama persis 100% plek ketiplek.
11. Tidak diperbolehkan **memasukkan jawaban dari soal UGD kelas lain** dan **memasukkan ulang jawaban dari soal Guided 1** (-100)
12. Dikarenakan Guided ini deadlinenya lama (3 minggu), **jika ketahuan plagiasi 100% dengan praktikan lain atau pun AI**, tidak akan diberikan toleransi **(AKAN DIANGGAP TIDAK MENGUMPULKAN GUIDED)**.

\*Note :

Soal UGD tidak akan jauh beda formatnya dengan Guided 2 ini.  
Dengan **mengerjakan sendiri Guided 2 ini**, akan sangat membantu kalian saat pengerjaan UGD nanti.



## KETENTUAN DAN FORMAT GUIDED

1. **Wajib** membaca modul sebelum mengerjakan Guided (ingat, saat praktikum jika ada pertanyaan yang jawabannya sudah ada di modul / guided, tidak akan dijawab oleh kakak-kakak asdos)
2. **Comment** tidak perlu ditulis di Guided
3. Wajib menggunakan new project dengan ekstensi **.c bukan .cpp**
4. Kedua folder **GD7\_X\_YYYYY\_1** dan **GD7\_X\_YYYYY\_2** di zip menjadi satu dengan nama **GD7\_X\_YYYYY.zip** (X = Kelas; Y = 5 digit terakhir NPM)
5. Kesalahan format penamaan file akan dilakukan pengurangan nilai **(-10)**
6. Tindak plagiasi tidak akan diberikan toleransi, terutama di guided 2
7. Jika masih ada yang ingin ditanyakan, bisa langsung kontak ke :
  - a. Teams : Eric Daniswara Octa Wijaya ([220711618@students.uajy.ac.id](mailto:220711618@students.uajy.ac.id))
  - b. WA : 085960199033
  - c. Discord : Eric Daniswara (@charlotte\_6969)

### \*Note :

Pelajari & hapalkan penggunaan rand (random) karena akan sangat berguna saat UGD nanti → **karena sudah aku spill disini, jika kalian bertanya saat UGD tidak akan dijawab oleh asisten ya 😊**.

Berlatihlah soal-soal studi kasus mengenai prosedur & berlatihlah mengetik cepat karena **code saat UGD nanti bakal cukup banyak.**

