

PRAKTIKUM DASAR PEMROGRAMAN
UNIVERSITAS ATMA JAYA YOGYAKARTA
2022/2023

BAGIAN I

PENDAHULUAN

Apakah teman-teman pernah mendengar kata “Prosedur”? Yapp mungkin prosedur sudah tidak asing didengar dalam kehidupan kita sehari-hari. Menurut Wikipedia, prosedur merupakan tata cara atau serangkaian aksi yang spesifik.

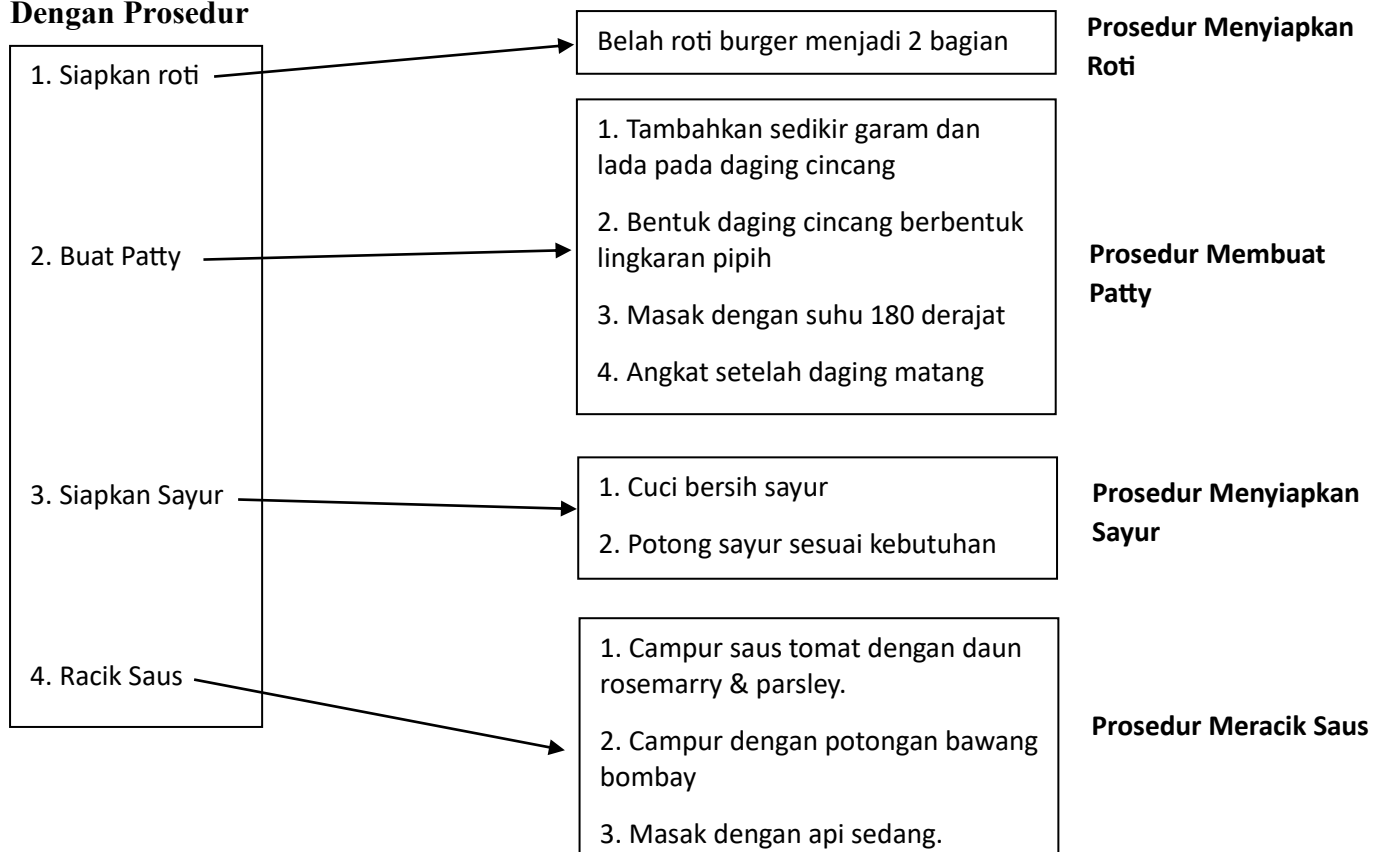
Coba kita lihat ilustrasi bagaimana seorang koki membuat hidangan hamburger di bawah:

Tanpa Prosedur



Belah roti burger menjadi 2 bagian. Patty dibuat dengan daging cincang kemudian tambah sedikit garam, lada dan dibentuk menjadi lingkaran pipih kemudian dimasak dengan suhu 180 derajat. Cuci bersih sayur kemudian dipotong potong sesuai dengan kebutuhan. Setelah daging matang, angkat dari teflon kemudian letakan pada wadah. Campur saus tomat, daun rosemary, daun parsley serta campur bawang bombay yang sudah matang kemudian masak dengan api sedang. Kemudian letakan roti pada piring, letakan sayur, tambahkan patty, tambahkan saos, dan tutup dengan roti kembali.

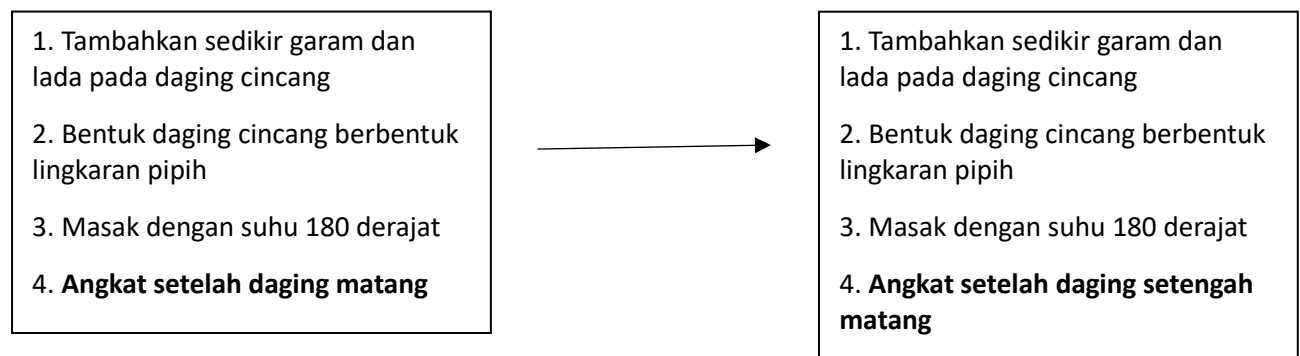
Dengan Prosedur



Kira-kira apa yang bisa disimpulkan dari ilustrasi di atas teman-teman? Ilustrasi seorang koki membuat hidangan hamburger dengan prosedur lebih mudah dipahami dan lebih terstruktur bukan? Dibandingkan dengan ilustrasi tanpa menggunakan prosedur.

Bayangkan apabila seorang *Excecutive Chef* ingin merubah resep hamburger dari daging yang matang menjadi daging setengah matang (medium). *Excecutive Chef* hanya perlu memilih prosedur membuat patty dan merubah sedikit langkah-langkahnya.

Prosedur Membuat Patty



Bandingkan apabila kita ingin merubahnya langkah-langkah membuat hamburger tanpa menggunakan prosedur.

Belah roti burger menjadi 2 bagian. Patty dibuat dengan daging cincang kemudian tambah sedikit garam, lada dan dibentuk menjadi lingkaran pipih kemudian dimasak dengan suhu 180 derajat. Cuci bersih sayur kemudian dipotong potong sesuai dengan kebutuhan. Setelah daging matang, angkat dari teflon kemudian letakan pada wadah. Campur saus tomat, daun rosemarry, daun parsley serta campur bawang bombay yang sudah matang kemudian masak dengan api sedang. Kemudian letakan roti pada piring, letakan sayur, tambahkan patty, tambahkan saos, dan tutup dengan roti kembali.

Seorang *Excecutive Chef* harus membaca dari awal dengan teliti hingga ia menemukan catatan mengolah daging. Sangat membuang waktu dan tenaga, serta tidak efisien bukan?

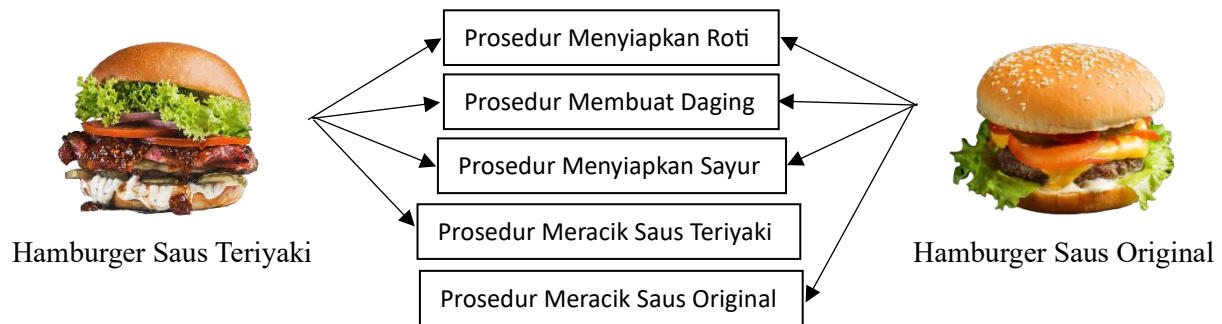
Begitu pula dalam pemrograman, semakin lama kalian mengembangkan sebuah program, maka semakin kompleks pula *code* yang kalian buat (ratusan hingga ribuan atau bahkan puluhan ribu baris code). Apabila kalian tidak menerapkan konsep prosedur dalam pemrograman, betapa sulitnya kalian dalam membuat program tersebut berhasil. Belum lagi apabila terdapat *Bug* pada fungsionalitas tertentu ketika program dijalankan/di-*compile*. Programmer akan sangat kesulitan apabila program yang dibuatnya tidak terstruktur dan tidak menerapkan konsep prosedur.

“Menggunakan prosedur akan membuat seorang programmer dapat merancang code yang terstruktur dan mudah dipahami.”

Balik pada ilustrasi di atas, bagaimana apabila seorang *Executive Chef* ingin menambah menu Hamburger Saus Teriyaki? Apakah harus membuat prosedur baru untuk menyiapkan roti, menyiapkan sayur, dan membuat patty pada menu yang baru?



Jawabannya **tidak perlu**. Hamburger Original & Saus teriyaki hanya berbeda pada jenis sausnya saja. Roti tetap sama, patty tetap sama, dan sayur tetap sama.



Kita dapat menarik kesimpulan, bahwa sebuah prosedur dapat digunakan **berulang kali** pada setiap perilaku yang sama. Hal yang sama terjadi pada pemrograman, prosedur dapat dipanggil berulang kali di dalam main (main merupakan bagian utama yang berasal dari prosedur atau fungsi untuk menjalankan program secara sekuensial).

“Menggunakan prosedur akan membuat code menjadi lebih efisien. Karena tidak perlu mengetik code yang sama berulang kali untuk masalah yang sama.”

Setelah kalian sudah mempunyai gambaran mengenai prosedur mengenai implementasi dalam dunia nyata maupun *programming*, maka selanjutnya kita akan membahas lebih dalam terkait penggunaan prosedur pada pemrograman bahasa C.

TUJUAN PRAKTIKUM

1. Praktikan dapat mengenal prosedur dalam Dasar Pemrograman.
2. Praktikan dapat mengetahui konsep dari prosedur dalam Dasar Pemrograman.
3. Praktikan dapat mengimplementasikan prosedur sesuai dengan kebutuhan dalam membuat program.

BAGIAN II

TEORI

Dalam pemrograman, prosedur adalah suatu **sub program** pada bahasa C untuk membuat program menjadi terstruktur. Prosedur membuat program C dapat **membagi tugas - tugas nya** sesuai apa yang kita perlukan, serta bisa kita panggil ke dalam Main program apabila diperlukan, sehingga program lebih mudah untuk dibaca dan lebih enak untuk dilihat. Dan juga dengan adanya prosedur akan menghemat memori karena memori akan terpakai jika dia dipanggil saja.

Prosedur digunakan untuk membantu programmer agar dapat **mengerjakan suatu hal berulang** maupun tidak berulang dengan **lebih mudah**. Terdapat juga beberapa permasalahan yang hanya dapat diselesaikan dengan penggunaan konsep-konsep dari prosedur.

Keuntungan dalam menggunakan prosedur yaitu:

1. Code menjadi lebih terstruktur
2. Proses Debugging menjadi lebih mudah
3. Efisiensi Code (Tidak perlu mengetik code yang sama berulang kali untuk masalah yang sama)
4. Jika bekerja dalam Tim, Prosedur dapat dengan mudah diatur dan digunakan oleh rekan tim.

Karakteristik dari sebuah prosedur:

1. Prosedur menggunakan keyword (*Syntax*) **void**.
2. Prosedur biasanya dinamai dengan sebuah kata kerja {tampilData, editData, hapusData,...}.
3. Prosedur dapat memiliki sebuah parameter maupun tidak.

4. Prosedur memiliki “**Lingkungan**”-nya sendiri, sehingga terdapat konsep *local* dan *global variabel*.
5. Umumnya, prosedur yang baik tidak memiliki standard *input/output*.

- **Struktur Prosedur**



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

Prosedur memiliki 3 bagian penting, yakni:

Nama, Parameter, dan Body

1. **Nama** → digunakan untuk mengidentifikasi dan memanggil prosedur. (**harus unik**)
2. **Parameter** → digunakan untuk memberikan inputan agar dapat digunakan di dalam prosedur. (bersifat opsional)
3. **Body** → merupakan isi dari prosedur bersangkutan.

- **Contoh Program tanpa Prosedur**



```
int main(){  
    string namaUser1;  
    int beratBadanUser1;  
  
    string namaUser2;  
    int beratBadanUser2;  
  
    strcpy(namaUser1,"Bobon");  
    beratBadanUser1 = 70;  
  
    strcpy(namaUser2,"Agus");  
    beratBadanUser2 = 65;  
  
    printf("Daftar Nama & Berat Badan: ");  
  
    printf("\n\tUsername      : %s",namaUser1);  
    printf("\n\tBerat badan    : %d",beratBadanUser1);  
  
    printf("\n");  
  
    printf("\n\tUsername      : %s",namaUser2);  
    printf("\n\tBerat badan    : %d",beratBadanUser2);  
  
    return 0;  
}
```

Pada contoh disamping, kita hanya memiliki 2 user. Andaikan saja teman-teman memiliki 5 user atau lebih. Maka teman-teman akan terus menerus **menulis baris code yang sama**.

Teman-teman akan menulis code **berulang kali**. Hal ini menyebabkan **pemborosan** dan tentu saja code akan sangat susah untuk di maintance.

- **Contoh Program dengan Prosedur**

```
typedef char string[30];

void inputData(string user, int *berat, string tempUser, int tempBerat);
void tampilData(string username, int berat);

int main(int argc, char *argv[]) {
    string user1;
    int beratBadanUser1;

    string user2;
    int beratBadanUser2;

    inputData(user1, &beratBadanUser1, "Bobon", 70);
    inputData(user2, &beratBadanUser2, "Agus", 65);

    printf("Daftar User : ");
    tampilData(user1, beratBadanUser1);
    tampilData(user2, beratBadanUser2);

    return 0;
}

void inputData(string user, int *berat, string tempUser, int tempBerat){
    strcpy(user, tempUser);
    *berat = tempBerat;
}

void tampilData(string username, int berat){
    printf("\n\tUsername    : %s", username);
    printf("\n\tBerat Badan : %d", berat);
    printf("\n");
}
```

Sedangkan jika teman-teman menggunakan Prosedur, maka teman-teman hanya perlu **memanggil prosedur** yang telah teman-teman buat.

Sehingga meskipun teman-teman memiliki 5 user atau lebih, teman-teman tidak perlu repot repot untuk menuliskan baris code yang sama.

TENANG TEMAN-TEMAN, jangan pusing terlebih dahulu, kita pelajari secara perlahan di halaman berikutnya.

- **Deklarasi, Definisi & Pemanggilan**

Implementasi prosedur dalam program dibagi menjadi 3, yaitu Deklarasi, Definisi, & Pemanggilan.

The image shows a C program with three parts highlighted and labeled:

- Deklarasi Prosedur** (Procedure Declaration):

```
typedef char string[30];  
void inputData(string user, int *berat, string tempUser, int tempBerat);  
void tampilData(string username, int berat);
```
- Pemanggilan Prosedur** (Procedure Calling):

```
int main(int argc, char *argv[]) {  
    string user1;  
    int beratBadanUser1;  
  
    string user2;  
    int beratBadanUser2;  
  
    inputData(user1, &beratBadanUser1, "Bobon", 70);  
    inputData(user2, &beratBadanUser2, "Agus", 65);  
  
    printf("Daftar User : ");  
    tampilData(user1, beratBadanUser1);  
    tampilData(user2, beratBadanUser2);  
  
    return 0;  
}
```
- Pendefinisian Prosedur** (Procedure Definition):

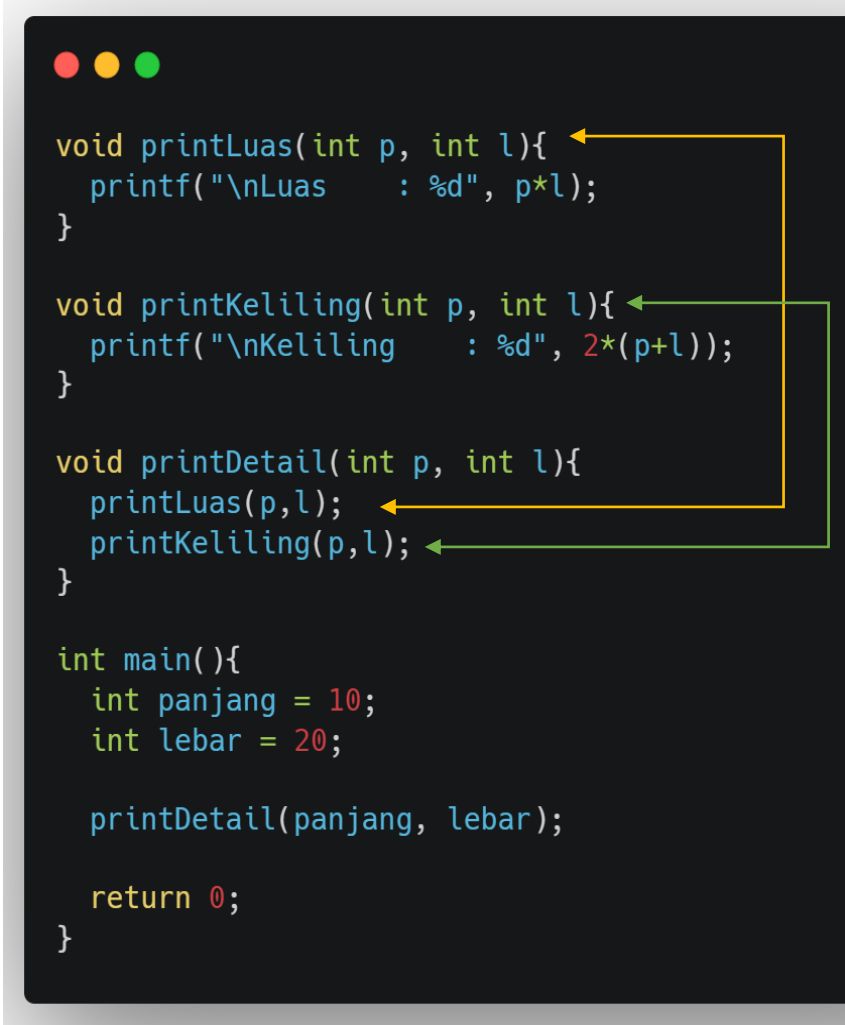
```
void inputData(string user, int *berat, string tempUser, int tempBerat){  
    strcpy(user, tempUser);  
    *berat = tempBerat;  
}  
void tampilData(string username, int berat){  
    printf("\n\tUsername      : %s", username);  
    printf("\n\tBerat Badan : %d", berat);  
    printf("\n");  
}
```

1. Deklarasi Prosedur

Pada hakekatnya, Prosedur dapat berjalan tanpa perlu di deklarasi terlebih dahulu pada beberapa kasus sederhana.

Mengapa Prosedur perlu atau disarankan dideklarasikan terlebih dahulu?

Yukk kita simak contoh **code tanpa deklarasi** berikut:



```
void printLuas(int p, int l){  
    printf("\nLuas      : %d", p*l);  
}  
  
void printKeliling(int p, int l){  
    printf("\nKeliling    : %d", 2*(p+l));  
}  
  
void printDetail(int p, int l){  
    printLuas(p,l);  
    printKeliling(p,l);  
}  
  
int main(){  
    int panjang = 10;  
    int lebar = 20;  
  
    printDetail(panjang, lebar);  
  
    return 0;  
}
```

The image shows a code editor with three colored window buttons (red, yellow, green) at the top left. The code is written in C. A yellow arrow points from the `printLuas` call inside `printDetail` to its definition above. A green arrow points from the `printKeliling` call inside `printDetail` to its definition above. This illustrates that the functions are called before they are defined, which is allowed in C for simple cases.

Fact Note :

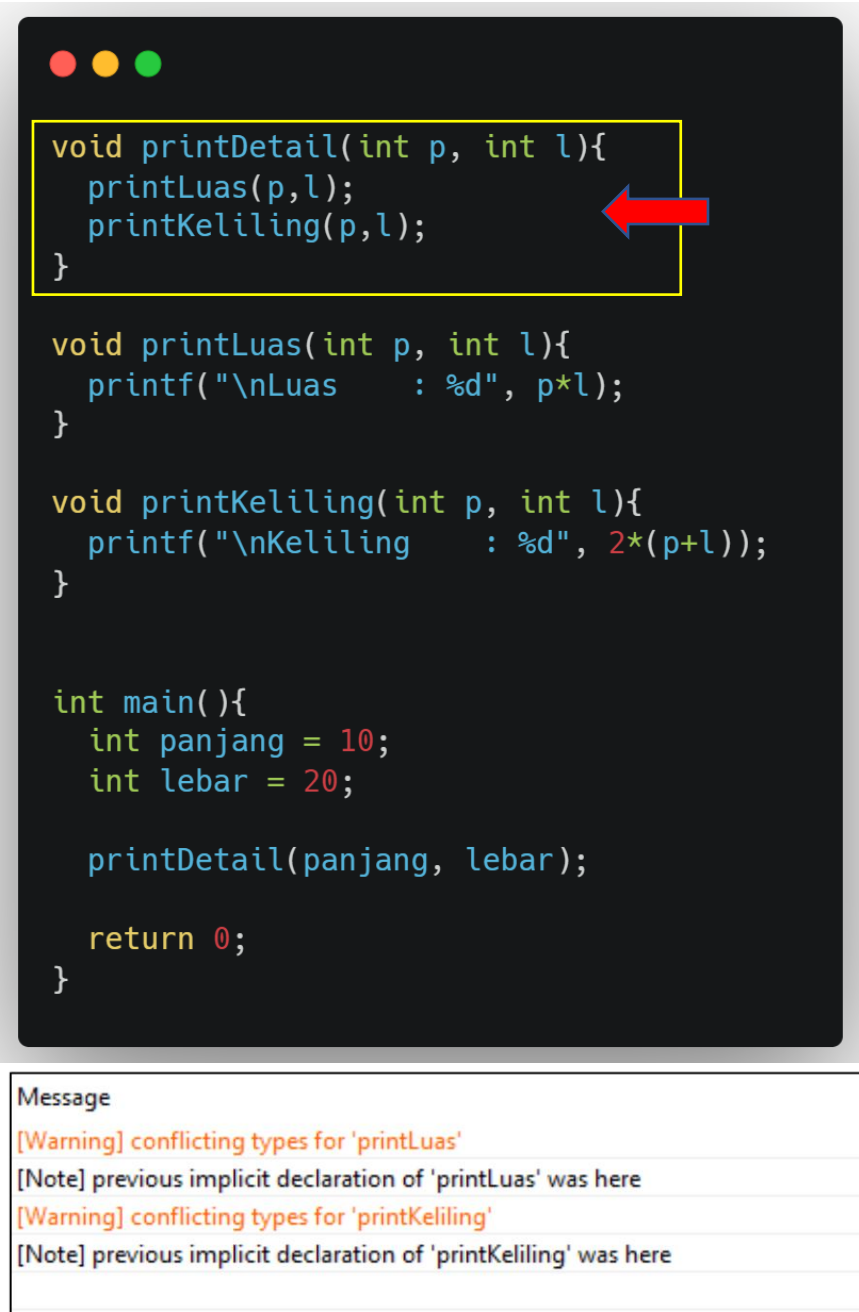
Sebuah Prosedur dapat dipanggil oleh prosedur lainnya. (**bukan hanya** bisa dipanggil di **main program** saja).

Bisa dilihat pada contoh, ketika teman-teman menjalankan program disamping. Maka program akan **berjalan seperti biasa**.

Dikarenakan `printLuas` dan `printKeliling` **telah didefinisi terlebih dahulu di atas `printDetail`**. Maka `printDetail` tidak akan mengalami kesulitan dalam memanggil prosedur yang dibutuhkan karena sebuah program akan mengeksekusi secara **sekuensial** (berurutan sesuai dengan kebutuhan dan alurnya).

Bagaimana apabila dibalik, prosedur *printDetail* terletak di atas *printDetail* & *printKeliling*?

Yukk kita simak contoh program di bawah:



```
void printDetail(int p, int l){
    printLuas(p,l);
    printKeliling(p,l);
}

void printLuas(int p, int l){
    printf("\nLuas      : %d", p*l);
}

void printKeliling(int p, int l){
    printf("\nKeliling    : %d", 2*(p+l));
}

int main(){
    int panjang = 10;
    int lebar = 20;

    printDetail(panjang, lebar);

    return 0;
}
```

Message

[Warning] conflicting types for 'printLuas'

[Note] previous implicit declaration of 'printLuas' was here

[Warning] conflicting types for 'printKeliling'

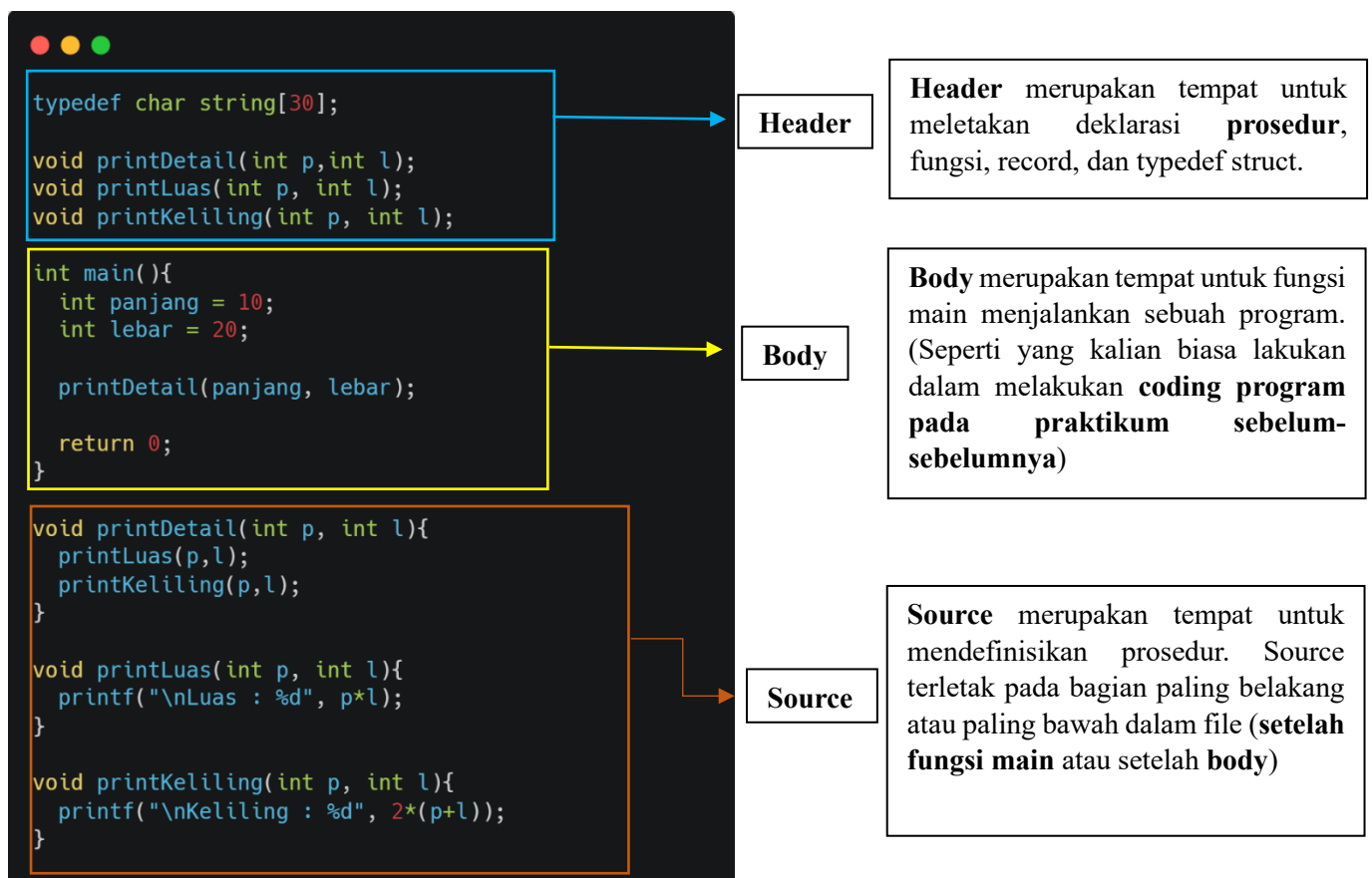
[Note] previous implicit declaration of 'printKeliling' was here

Pada contoh ini, program akan mengeluarkan **Warning**. Hal ini dikarenakan pada saat program mengeksekusi prosedur *printDetail*, program tidak menemukan Body dari *printLuas* dan *printKeliling* (Karena prosedur *printDetail* didefinisi terlebih dahulu sebelum program mengeksekusi definisi prosedur *printLuas* dan *printKeliling*) . sehingga program akan mengeluarkan **Warning**.

Hal ini dapat dicegah dengan cara **mendeklarasikan** terlebih dahulu prosedur-prosedur yang akan kita gunakan atau definisikan.

Bagaimana sih cara mendeklarasikan Prosedur?

Nah untuk mendeklarasikan prosedur sebenarnya sangat amat mudah. Deklarasi prosedur dilakukan pada **header program** (header sendiri artinya bagian dari program yang berada pada **margin atas**). Prosedur dideklarasikan **setelah import library** dan **sebelum masuk pada main program**.



Dengan **mendeklarasikan prosedur terlebih dahulu**, maka semua **warning** tersebut akan hilang. Teman-teman diharapkan untuk mendeklarasikan terlebih dahulu prosedur yang digunakan agar tidak terjadi **Undefined Behavior** pada program teman-teman. Dengan mendeklarasikan prosedur, teman-teman juga dapat dengan mudah melihat apa saja prosedur yang telah teman-teman buat.

Struktur Deklarasi Prosedur:

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

2. Pendefinisian Prosedur

Mendefinisikan Prosedur merupakan hal yang esensial dilakukan dalam menerapkan konsep prosedur pada program. Mendefinisikan prosedur berarti membuat perintah spesifik sesuai dengan konteks prosedur yang dibuat.

Contoh:

```
void Penjumlahan (int bilangan1, int bilangan2){  
    printf("Jumlah kedua bilangan adalah %d", bilangan1 + bilangan2);  
}
```

Pada contoh di atas, terdapat pendefinisian sebuah prosedur untuk menampilkan jumlah bilangan1 dan bilangan2.

Struktur Mendefinisikan Prosedur:

```
void <namaProsedur> (<parameter1, parameter2, ....>){  
    <body> //Memberi perintah pada program untuk melakukan  
           sebuah aksi ketika sebuah prosedur dipanggil  
}
```

3. Pemanggilan Prosedur

Prosedur menjadi tidak berguna apabila tidak dipanggil pada main program. Prosedur **hanya akan mengeksekusi perintah** di dalamnya, apabila prosedur tersebut di panggil oleh sebuah **main program** ataupun prosedur lain (yang juga dipanggil di dalam main program).

Contoh Pemanggilan Prosedur:

Header

```
int main(){  
    int bil1, bil2;  
  
    bil1 = 2;  
    bil2 = 3;  
  
    Penjumlahan(bil1, bil2);  
  
    return 0;  
}
```

Memanggil

Source

```
void Penjumlahan (int bilangan1, int bilangan2){  
    printf("Jumlah kedua bilangan adalah %d", bilangan1 + bilangan2);  
}
```

Struktur Pemanggilan Prosedur:

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

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.

Contoh:

Dalam Main Program:

Tipe Data variabel *bil1* dan *bil2* yang masuk pada parameter prosedur penjumlahan adalah **integer**.

Diluar Main Program:

Tipe Data pada parameter prosedur penjumlahan yaitu *bilangan1* dan *bilangan2* yaitu **integer** pula.

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**.”

(Parameter Formal & Aktual, akan dipelajari lebih dalam di halaman-halaman berikutnya)

Contoh pada kasus di atas, nama variabel pada parameter ketika memanggil prosedur penjumlahan adalah *bil1* dan *bil2* . Akan tetapi, nama variabel atau parameter yang melekat pada prosedur adalah *bilangan1* dan *bilangan2*.

Agar lebih paham mengenai penggunaan Parameter, halaman selanjutnya akan menjelaskan secara lebih mendalam mengenai **Parameter**.

- **Parameter**

Parameter dapat diibaratkan sebagai **informasi (variabel)** apa saja yang yang dibutuhkan untuk menunjang jalannya sebuah aksi pada prosedur.

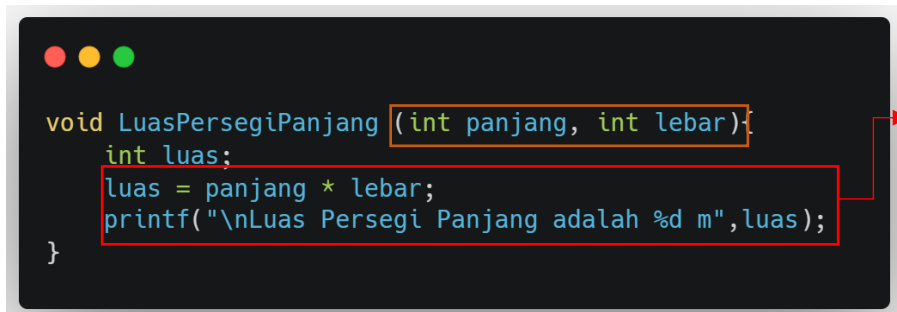
Misal:

Seorang programmer akan membuat sebuah prosedur untuk **menghitung luas persegi panjang**. Maka informasi apa saja yang diperlukan dalam prosedur?

Diketahui bahwa rumus luas persegi panjang adalah demikian:

$$\text{Luas} = \text{Panjang} \times \text{Lebar}$$

Maka untuk mencari luas persegi panjang, seorang programmer membutuhkan informasi **panjang** dan **lebar** dari sebuah persegi panjang untuk kebutuhan menjalankan atau mengeksekusi sebuah prosedur. Maka programmer bisa **menentukan parameter** dari prosedur Luas Persegi Panjang sebagai **masukan (input)** pada sebuah prosedur yaitu **panjang dan lebar** dengan **tipe data integer** (karena value nya berupa angka).

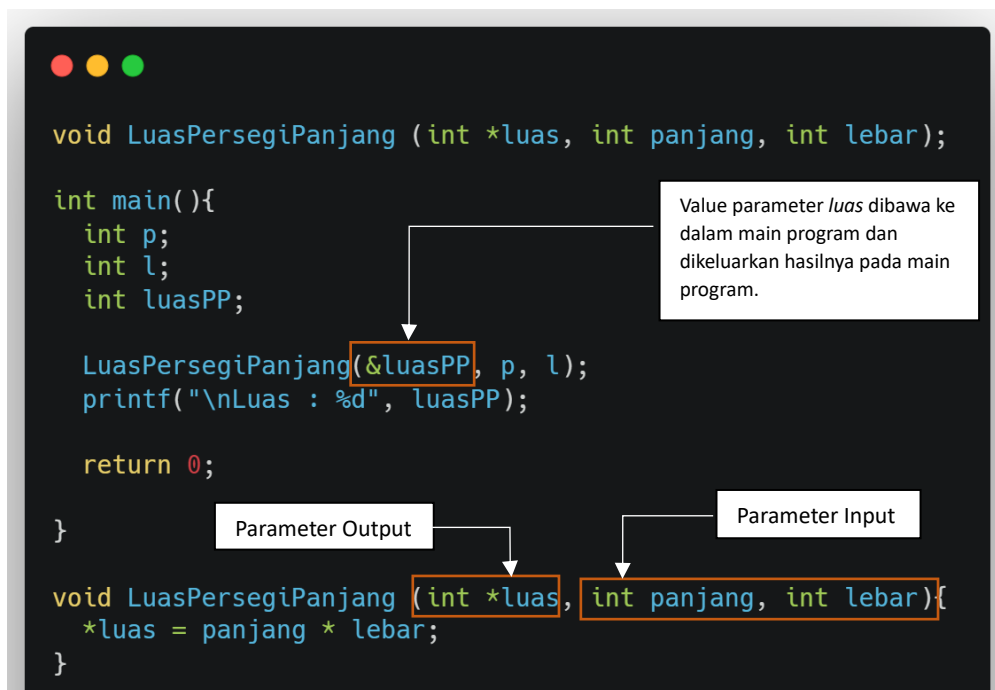


```
void LuasPersegiPanjang (int panjang, int lebar){
    int luas;
    luas = panjang * lebar;
    printf("\nLuas Persegi Panjang adalah %d m",luas);
}
```

Informasi atau parameter langsung diproses dan **dikeluarkan (output)** didalam prosedur.

Contoh prosedur di atas menggunakan **seluruhnya parameter input**.

Terkadang kita membutuhkan variabel dari prosedur untuk **dibawa nilainya kedalam variabel yang ada pada main program** untuk dikeluarkan dalam main program atau diolah lebih lanjut dalam main program (**Parameter Output**).



Contoh prosedur di atas menggunakan **parameter input & output (Nett Effect)**.

(Parameter input, output & input output akan dipelajari lebih dalam di modul Prosedur 2)

Kategori Prosedur

Berdasarkan darimana datangnya input dan output, prosedur dikategorikan menjadi 4 jenis, yakni Naïve, Semi-Naïve Input, Semi-Naïve Output, dan Nett Effect. Perbedaan diantara keempat jenis prosedur tersebut terdapat pada parameter yang digunakan.

1. Naïve

Merupakan prosedur yang tidak memiliki Parameter sama sekali.

```
void tampilMenu(){  
    printf("\n[1]. LOGIN");  
    printf("\n[2]. ISI DATA");  
}
```

2. Semi-Naïve Input

Merupakan Prosedur yang hanya memiliki parameter Input dan menghasilkan output yang dikeluarkan melalui Standard I/O.

```
void hitungKeliling(int pjg, int lbr){  
    int keliling;  
    keliling = 2*(pjg+lbr);  
    printf("\nKeliling = %d cm",keliling);  
}
```

3. Semi-Naïve Output

Merupakan Prosedur yang hanya memiliki parameter Output dan menggunakan input yang didapat melalui Standard I/O.

```
void hitungKeliling(int *keliling){  
    int pjg, lbr;  
    printf("\nInput Panjang : %d");scanf("%d",&pjg);  
    printf("\nInput Lebar : %d");scanf("%d",&lbr);  
    *keliling = 2*(pjg+lbr);  
}
```

4. Nett Effect

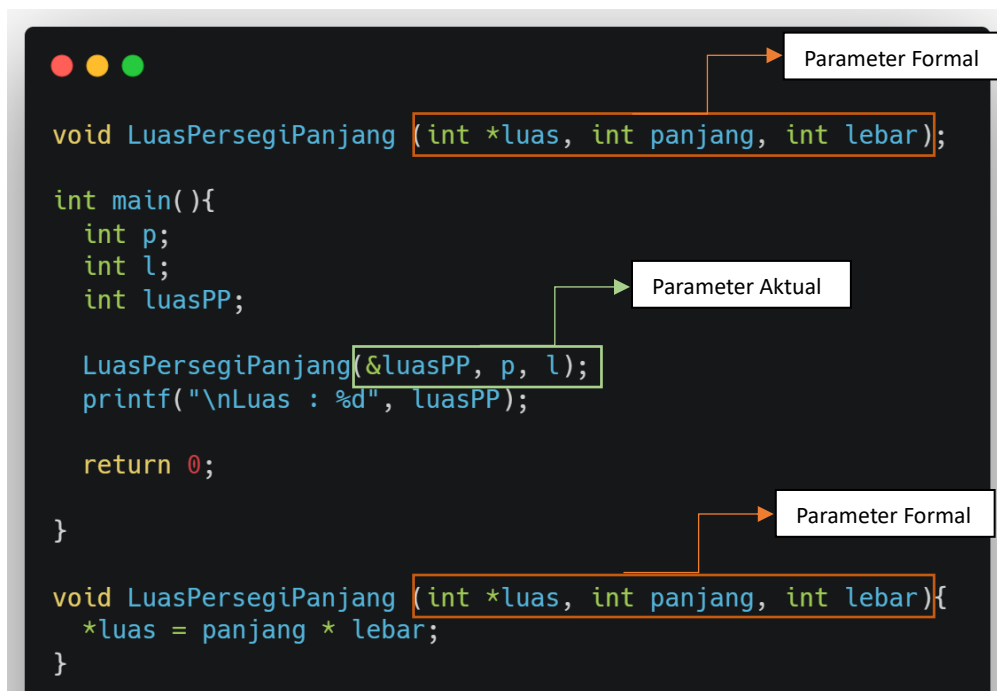
Merupakan Prosedur yang tidak menggunakan Standard I/O (Input/Output).

```
void hitungKeliling(int *keliling, int pjg, int lbr){  
    *keliling = 2*(pjg+lbr);  
}
```

Prosedur **Nett Effect** merupakan jenis prosedur input-output yang paling direkomendasikan. (Menghasilkan efek netto)

Jenis Parameter

Parameter dibedakan menjadi 2 jenis, yaitu parameter **Formal** dan parameter **Aktual**. **Parameter Formal** merupakan parameter yang ada **di luar main program** (parameter yang dideklarasikan pada bagian header prosedur). Sedangkan **Parameter Aktual** merupakan parameter yang disertakan pada saat **pemanggilan**.



```
void LuasPersegiPanjang (int *luas, int panjang, int lebar);

int main(){
    int p;
    int l;
    int luasPP;

    LuasPersegiPanjang(&luasPP, p, l);
    printf("\nLuas : %d", luasPP);

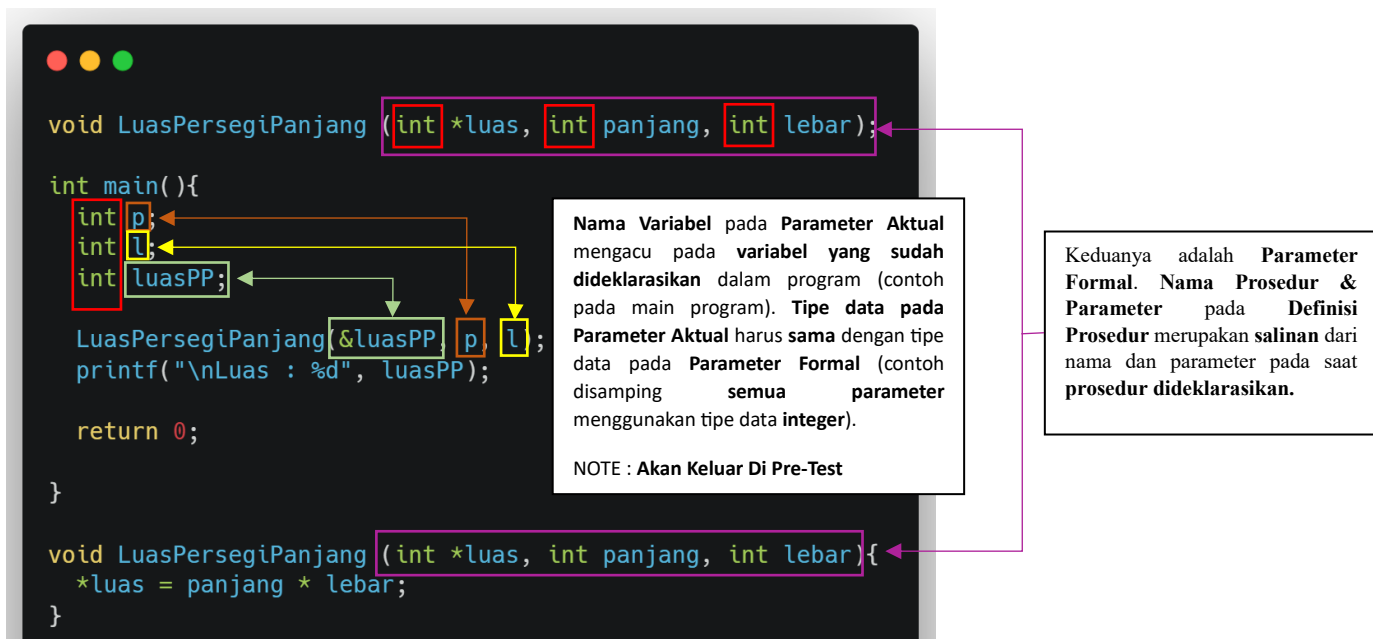
    return 0;
}

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

The image shows a C program with three annotations: 'Parameter Formal' pointing to the parameter list in the function declaration and definition, and 'Parameter Aktual' pointing to the arguments in the function call.

Gimana? Udah paham mengenai jenis parameter di atas belum?

Parameter Formal & Aktual lah yang menjawab pertanyaan, mengapa **nama parameter (variabel) yang dipanggil dapat berbeda** dengan nama parameter yang ada pada saat **prosedur dideklarasikan**.



```
void LuasPersegiPanjang (int *luas, int panjang, int lebar);

int main(){
    int p;
    int l;
    int luasPP;

    LuasPersegiPanjang(&luasPP, p, l);
    printf("\nLuas : %d", luasPP);

    return 0;
}

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

Nama Variabel pada Parameter Aktual mengacu pada variabel yang sudah dideklarasikan dalam program (contoh pada main program). Tipe data pada Parameter Aktual harus sama dengan tipe data pada Parameter Formal (contoh disamping semua parameter menggunakan tipe data integer).

NOTE : Akan Keluar Di Pre-Test

Keduanya adalah Parameter Formal. Nama Prosedur & Parameter pada Definisi Prosedur merupakan salinan dari nama dan parameter pada saat prosedur dideklarasikan.

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.

```
void cariSelisih (string nama, float nilai);

int main(){
    string namaUser; strcpy(namaUser,"Keju");
    float nilaiUser = 3.9;
    cariSelisih(namaUser,nilaiUser);

    return 0;
}

void cariSelisih (string nama, float nilai){
    printf("Nama User : %s", nama);
    printf("Nilai User : %.2f",nilai);
}
```

Tipe data dan urutan dari Parameter Aktual harus sama atau mengikuti dengan **Parameter Formal**. Contoh pada gambar disamping, **urutan pertama** pada Parameter Formal adalah **string**. Sedangkan, **urutan kedua** pada Parameter Formal adalah **float**.

Pemanggilan pada prosedur harus diisi dengan urutan dan tipe data yang sama pula. **Parameter Aktual** harus diisi **urutan pertama** dengan tipe data **string & urutan kedua** dengan tipe data **float**.

Default Parameter (Naïve)

Parameter prosedur bersifat optional, maka **prosedur dapat untuk tidak memiliki parameter**. Sebagai contohnya adalah sebuah prosedur untuk menampilkan menu dari program yang kita buat. Untuk memanggil prosedurnya, maka teman-teman tidak memerlukan menginputkan parameter apapun karena pada dasarnya prosedur tersebut **tidak membutuhkan informasi atau variabel** untuk ditampilkan atau diolah dalam prosedur tersebut.

Contoh:

```
void tampilMenu(){
    printf("\n[1] Login");
    printf("\n[2] Isi Data");
    printf("\n[3] Hapus Data");
}
```

POINTER

Teman-teman kita coba balik ke **halaman 16** (contoh prosedur parameter input & output). Teman-teman pasti melihat perbedaan implementasi diantara **parameter input & parameter output**. Apabila teman-teman melihat ada **bintang** atau **asteris** (**pointer**) yang menempel pada **variabel** atau **parameter formal**, artinya **value** atau nilai dari variabel atau **parameter aktual** akan ikut berubah. Sebaliknya, ketika teman-teman menggunakan **parameter biasa**, maka **perubahan value** atau nilai pada **Parameter Formal** tidak akan mengakibatkan perubahan pada **Parameter Aktual**.

```
void prosedurPointer (int *temp);
void prosedurBiasa (int temp);

int main(){
    int tempA = 1;
    int tempB = 2;

    printf("\nTemp A Sebelum : %d", tempA); //Output 1
    printf("\nTemp B Sebelum : %d\n", tempB); //Output 2
    prosedurPointer(&tempA);
    prosedurBiasa(tempB);
    printf("\nTemp A Sesudah : %d", tempA); //Output 100
    printf("\nTemp B Sesudah : %d", tempB); //Output 2

    return 0;
}

void prosedurPointer (int *temp){
    *temp = 100;
}
void prosedurBiasa (int temp){
    temp = 50;
}
```

Untuk lebih memahami perbedaan Parameter menggunakan Pointer (Parameter Output) dan Parameter Biasa, teman-teman dapat mengcopy-paste code di samping ini dan memperhatikan perubahan nilai yang terjadi di dev C++ kalian.

NOTE: Perhatikan baik-baik perbedaannya teman-teman, supaya kalian tidak kena jebakan pada saat Pre-Test.

Parameter Pointer dan Parameter Biasa akan dibahas lebih lanjut di Modul Selanjutnya (Prosedur 2).

PENDALAMAN PRAKTEK

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

Variabel atau parameter dengan pointer (Parameter Output) akan selalu membawa pointernya pada aksi dalam prosedur. Pada contoh di atas, terdapat variabel atau parameter dengan pointer yaitu **luas*. Ketika variabel tersebut digunakan untuk memenuhi kebutuhan prosedur seperti aksi **luas = panjang * lebar*. Maka pointer tetap dibawa melekat pada variabel luas seperti pada contoh di atas.

```
int main(){  
    int p;  
    int l;  
    int luasPP;  
  
    LuasPersegiPanjang(&luasPP, p, l);  
    printf("\nLuas : %d", luasPP);  
  
    return 0;  
}
```

Apabila variabel luas terdapat pointer yang melekat pada **Parameter Formal**-nya, maka pada saat pemanggilan atau pada **Parameter Aktual**-nya menggunakan simbol “&” yang melekat pada variabel atau parameter (**bukan** justru pointer atau “*” yang melekat).

Agar lebih terbiasa dengan penggunaan prosedur, teman-teman dapat membuat **GUIDED** di bawah dengan seksama dan memperhatikan catatan-catatan penting yang ada pada **GUIDED**.

GUIDED

Buatlah sebuah program untuk membantu pelajar matematika dalam menghitung luas persegi panjang dan keliling persegi panjang. Pelajar matematika akan menginputkan data panjang & lebar, tetapi pelajar matematika juga perlu meminta bantuan program untuk mengisi data panjang & lebar secara acak agar pelajar matematika dapat berlatih berhitung. Program memiliki 4 menu didalamnya yaitu:

Menu 1 (Input Data)

Pada menu ini, program akan meminta inputan user berupa panjang dan lebar. Panjang dan lebar tidak boleh kurang dari sama dengan 0. Apabila panjang dan lebar kurang dari sama dengan 0, maka muncul error handling dan program akan meminta inputan ulang. Menu ini hanya dapat diakses 1 kali.

```
[1]. Input Panjang & Lebar
[2]. Tampil Luas Persegi Panjang
[3]. Tampil Keliling Persegi Panjang
[4]. Generate Panjang & Lebar Random

>>> 1

Masukkan Panjang : 0

Panjang tidak boleh kurang dari sama dengan 0!

Masukkan Panjang : 3

Masukkan Lebar : 4

Berhasil input
```

Menu 2 (Tampil Luas Persegi Panjang)

Menu ini hanya bisa diakses apabila user sudah mengakses menu 1 atau 4. Program akan menampilkan informasi panjang dan lebar (saat ini) yang berasal dari inputan user atau angka random kemudian menampilkan hasil luas persegi panjang yg berasal dari data yang ada saat ini.

```
[1]. Input Panjang & Lebar
[2]. Tampil Luas Persegi Panjang
[3]. Tampil Keliling Persegi Panjang
[4]. Generate Panjang & Lebar Random

>>> 2

-----Data Panjang & Lebar Persegi Panjang-----
Panjang Persegi Panjang (Saat ini) = 3
Lebar Persegi Panjang (Saat ini) = 4

Luas Persegi Panjang adalah 12 cm
```

Menu 3 (Tampil Keliling Persegi Panjang)

Menu ini hanya bisa diakses apabila user sudah mengakses menu 1 atau 4. Program akan menampilkan informasi panjang dan lebar (saat ini) yang berasal dari inputan user atau angka random kemudian menampilkan hasil keliling persegi panjang yg berasal dari data yang ada saat ini.

```
[1]. Input Panjang & Lebar
[2]. Tampil Luas Persegi Panjang
[3]. Tampil Keliling Persegi Panjang
[4]. Generate Panjang & Lebar Random

>>> 3

-----Data Panjang & Lebar Persegi Panjang-----
Panjang Persegi Panjang (Saat ini) = 3
Lebar Persegi Panjang (Saat ini) = 4

Keliling Persegi Panjang adalah 14 cm
```

Menu 4 (Generate Panjang & Lebar Random)

Menu ini dapat diakses kapanpun. Program akan diminta untuk menentukan angka acak 1-5 untuk mengisi variabel panjang dan 6-10 untuk mengisi variabel lebar secara otomatis (tanpa meminta inputan dari user).

```
[1]. Input Panjang & Lebar
[2]. Tampil Luas Persegi Panjang
[3]. Tampil Keliling Persegi Panjang
[4]. Generate Panjang & Lebar Random

>>> 4

-----Data Panjang & Lebar Persegi Panjang-----
Panjang Persegi Panjang (Saat ini) = 1
Lebar Persegi Panjang (Saat ini) = 8
```

Apabila teman-teman ingin berlatih mengerjakan tanpa melihat kunci di bawah, akan sangat diperbolehkan dan dianjurkan agar pemahaman dan praktek teman-teman dalam materi Prosedur 1 ini lebih matang. Apabila sudah selesai mengerjakan, silahkan cocokan dengan kunci Guided di bawah ini yaa. Pastikan output dari program teman-teman sesuai dengan permintaan soal. Semangatt temen2 😊


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <conio.h>
#include <time.h>

typedef char string[50];

void tampilMenu();
void inputData(int *dataAsli, int dataTampung);
void tampilData(int panjang, int lebar);
void hitungLuas(int panjang, int lebar);
void hitungKeliling(int *keliling, int panjang, int lebar);
void gatchaAngka(int *angka, int min, int max);

int main(int argc, char *argv[]) {
    srand(time(NULL)); //deklarasi untuk mendapatkan angka atau karakter random
    int menu, temp;
    int panjang, lebar;
    int keliling; //Mengapa Luas tidak di deklarasi di dalam sini? Coba kalian cari tahu yaa..
    int angkaRandom;
    bool cekInput = false;

    do{
        tampilMenu();
        printf("\n>>> ");scanf("%d",&menu);
        switch(menu){
            case 1:
                if(!cekInput ){ //if(!cekInput) merupakan bentuk lain dari if(cekInput == false)
                    do{
                        printf("\nMasukkan Panjang : ");scanf("%d",&temp);
                        if(temp<=0){
                            printf("\nPanjang tidak boleh kurang dari sama dengan 0!\n");
                        }else{
                            inputData(&panjang, temp);
                            break;
                        }
                    }
                    }while(true); //while(true) merupakan bentuk lain dari while(temp<=0)
                        //dengan syarat wajib meletakkan break; pada kondisi else

                    do{
                        printf("\nMasukkan Lebar : ");scanf("%d",&temp);
                        if(temp<=0){
                            printf("\nLebar tidak boleh kurang dari sama dengan 0!\n");
                        }else{
                            inputData(&lebar, temp);
                            break;
                        }
                    }
                    }while(true); //while(true) merupakan bentuk lain dari while(temp<=0)
                        //dengan syarat wajib meletakkan break; pada kondisi else

                    printf("\nBerhasil input");
                    cekInput = true;
                }else{
                    printf("\nAnda sudah input!");
                }
            }
        break;
    }
```

```
case 2:
    if(cekInput){ //if(cekInput) merupakan bentuk lain dari if(cekInput == true)
        tampilData(panjang, lebar);
        hitungLuas(panjang, lebar);
    }else{
        printf("\nAnda belum input!");
    }
    break;

case 3:
    if(cekInput){ //if(cekInput) merupakan bentuk lain dari if(cekInput == true)
        tampilData(panjang, lebar);
        hitungKeliling(&keliling, panjang, lebar);
        //varibael keliling yg sudah di deklarasi pada main, nilainya berubah
        //sesuai perhitungan pada prosedur dan dibawa dalam parameter output

        printf("\n\n\tKeliling Persegi Panjang adalah %d cm", keliling);
    }else{
        printf("\nAnda belum input!");
    }
    break;

case 4:
    gatchaAngka(&angkaRandom, 1, 5);
    //dalam pemanggilan parameter bisa diisi langsung dengan value
    //contoh diatas parameter min diisi dengan 1 dan max diisi dengan 5
    //mengenerate angka random dari 1 sampai 5;

    inputData(&panjang, angkaRandom);
    //mengisi Panjang dg angka random yg telah di generate

    gatchaAngka(&angkaRandom, 6, 10);
    ;

    inputData(&lebar, angkaRandom);
    //mengisi Lebar dg angka random yg telah di generate

    tampilData(panjang, lebar);
    //menampilkan kembali informasi panjang dan lebarnya.
    cekInput = true;
    break;

case 0:
    printf("\n\n\t <isi nama kalian> - <kelas> - <NPM kalian>");
    //contoh printf("\n\n\t Kevin Julian Rahadinata - A - 210711024");
    break;

default:
    printf("\nMenu tidak ada!");
    break;

}getch();

}while(menu!=0);

return 0;
}
```

```
void tampilMenu(){
    system("cls"); // berfungsi untuk clear tampilan layar sebelumnya
    puts("[1]. Input Panjang & Lebar");
    puts("[2]. Tampil Luas Persegi Panjang");
    puts("[3]. Tampil Keliling Persegi Panjang");
    puts("[4]. Generate Panjang & Lebar Random");
}

void inputData(int *dataAsli, int dataTampung){
    *dataAsli = dataTampung;
    //perhatikan bahwa "*" atau pointer selalu menempel pada variabel
    //yang pada parameternya menggunakan pointer
    //tidak hanya di parameter saja
}

void tampilData(int panjang, int lebar){
    printf("\n\t-----Data Panjang & Lebar Persegi Panjang-----");
    printf("\n\tPanjang Persegi Panjang (Saat ini) = %d", panjang);
    printf("\n\tLebar Persegi Panjang (Saat ini) = %d", lebar);
}

void hitungLuas(int panjang, int lebar){
    int luas;
    luas = panjang * lebar;
    printf("\n\n\tLuas Persegi Panjang adalah %d cm",luas);
    //karena variabel luas tidak di bawa atau di passing ke dalam main
    //maka dicetak luasnya didalam prosedur.
}

void hitungKeliling(int *keliling, int panjang, int lebar){
    *keliling = 2*(panjang+lebar);
}

void gachaAngka(int *angka, int min, int max){
    *angka = (rand() % (max - min+1) + min);
    //artinya didapatkan angka random dari rentang min sampai dengan max
    //misal min = 3 & max = 5, maka didapatkan rentang angka 3 sampai dengan 5
}
```

Ketentuan & Format GUIDED

1. Untuk comment tidak menjadi masalah jika tidak ditulis di Guided
2. Teman-teman boleh berkreasi dengan Guided seperti mengganti nama variabel, membuat prosedur lain, mengganti logika program, dll. Dengan ketentuan
 - Program bisa di-compile tentunya
 - Fungsionalitas Program tetap terjaga
 - Memiliki minimal 6 prosedur
3. Tidak diperkenankan untuk menggunakan modul-modul selanjutnya yaaa... [Fungsi, Array, Record]
4. Semua kode dimasukkan ke dalam sebuah folder dengan format penamaan: GD7_X_YYYY
5. Folder tadi kemudian di Zip dengan format penamaan: GD7_X_YYYY.zip
6. Keterangan:
 - X = Kelas
 - YYYY = 4 digit terakhir NPM Praktikkan

Apabila dari teman-teman masih belum paham dan ingin bertanya terkait dengan Modul & Guided Prosedur 1 dapat menghubungi CP di bawah yaa. Mohon maaf apabila ada kesalahan, baik itu penjelasan kurang bisa dimengerti ataupun kesalahan ketik dalam modul di atas. Sebagai bahan evaluasi, kakak meminta bantuan kalian agar bisa mengisi form di bawah yaa :) See u in Class 😊

Link Evaluasi:

<https://forms.gle/7QJzgoVUwUYFUVj18> (anonym)

Contact Person:

Teams : Kevin Julian Rahadinata (210711024@students.uajy.ac.id)

WA : 0897 5804 210 (apabila butuh respon cepat)