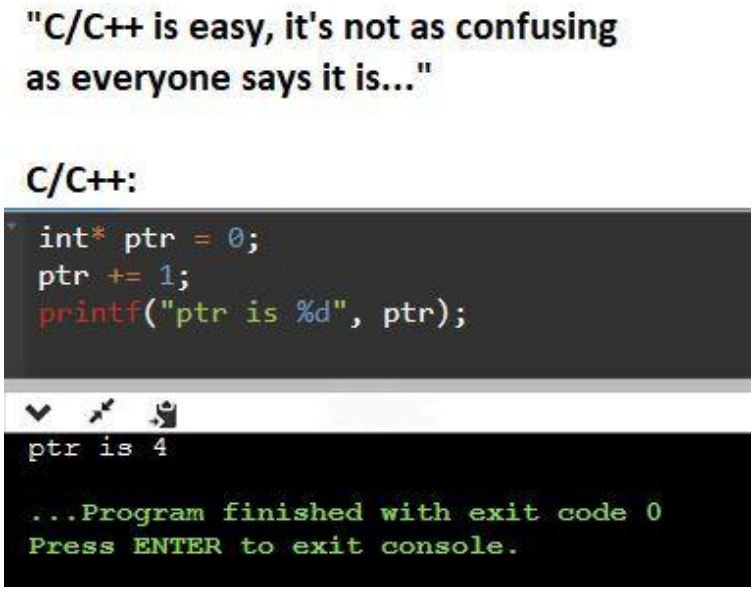
MODUL 7

PROSEDUR 1



A. Tujuan Praktikum

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

B. Pendahuluan

Kata “Prosedur” sendiri tentu sudah tidak asing di telinga teman-teman semua. Berdasarkan dari 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 dalam kehidupan sehari-hari ketika kita membaca prosedur membuat mie di belakang bungkus kemasan, maka prosedur tersebut akan berisi langkah-langkah secara runtut dan lengkap untuk membuat mie dari 0

hingga menjadi produk jadi. Langkah- langkah tersebut sudah tersimpan secara rapi dan terstruktur di dalam judul prosedur “Cara Mambuat Mie”. Sewaktu-waktu ketika ingin membuat mie lagi, kalian tinggal mencari judul prosedur “Cara Membuat Mie” dan di dalamnya kalian tinggal membaca & mengikuti langkah-langkah yang sudah disediakan.

Sama halnya dalam dunia pemrograman, konsep prosedur digunakan agar programmer tidak menuliskan kode secara berulang di dalam main programnya. Terutama jika kode tersebut panjang dan rumit. Programmer hanya perlu memanggil prosedur yang sudah dibuat dan meletakkannya di main program sama halnya dengan contoh sebelumnya saat ingin membuat mie lagi, hanya perlu mencari judul prosedur “Cara Membuat Mie” lagi. Di dalam programming, saat memanggil prosedur yang kalian buat, program akan membaca & menjalankan kode yang telah programmer buat di dalam prosedur tersebut.

C. Pengertian Prosedur

Prosedur adalah sub program yang berisi segala alur dan logika untuk menjalankan sebuah kode tertentu. Prosedur seperti sebuah kode tersendiri yang bisa kita panggil ke dalam main program saaat kita membutuhkannya. Penggunaan prosedur juga akan membuat program menjadi lebih rapi, efisien, dan hemat memori. Prosedur juga berguna apabila programmer ingin menjalankan sebuah code yang berulang maupun tidak karena tidak perlu menulis code tersebut secara berulang dan hanya perlu memanggil prosedur yang telah dibuat.

- **Struktur Prosedur**

  Prosedur terdiri dari 3 bagian penting, yaitu:

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

1. **Nama Prosedur :** Nama digunakan untuk mengidentifikasi dan memanggil prosedur (harus unik)
2. **Parameter :** Untuk mmeberikan inputan supaya bisa digunakan di dalam prosedur (opsional)
3. **Body :** Isi dari prosedur berupa alur dan logika dari suatu code.

- **Contoh Penggunaan Prosedur**

```c
// Pendeklarasian Prosedur
void luasPersegi(int *luas, int sisi);

int main(int argc, char const *argv[])
{
    // Pemanggilan Prosedur
    luasPersegi(&luas, sisi);

    return 0;
}

// Pendefinisian Prosedur
void luasPersegi(int *luas, int sisi)
{
    *luas = sisi * sisi;
}
```

- **Deklarasi Prosedur**

```
void <namaProsedur> (<prosedur1, prosedur2, ...>);
```

```
void initialisasi(string nama, int umut, string npm);
```

Pendeklarasian prosedur pada dasar pemrograman dilakukan di bagian margin atas pada program atau pada bagian sebelum main program. Pada semester 3 dalam konsep ADT, pendeklarasian prosedur akan dilakukan pada file header.

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

// Pendeklarasian Prosedur
void bil(int *bil1, int *bil2) // <--- Terletak Pada Margin Atas Program

int main(int argc, char *argv[]) {
    int bil1 = 3, bil2 = 5;

    printf("\tSebelum Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    bil(&bil1, &bil2); // Pemanggilan Prosedur

    printf("\n\tSesudah Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    return 0;
}

// Pendefinisian Prosedur
void bil(int *bil1, int *bil2)
{
  int temp;

    temp = (*bil1);
    (*bil1) = (*bil2);
    (*bil2) = temp;
}
```

Prosedur akan tetap berjalan meski tidak di deklrasikan. Namun, akan memicu warning conflicting types karena program tidak menemukan body dari prosedur terkait saat dieksekusi.

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

// Tidak Ada Pendeklarasian Prosedur

int main(int argc, char *argv[]) {
    int bil1 = 3, bil2 = 5;

    printf("\tSebelum Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    bil(&bil1, &bil2); // Pemanggilan Prosedur

    printf("\n\tSesudah Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    return 0;
}

// Pendefinisian Prosedur
void bil(int *bil1, int *bil2)
{
  int temp;

    temp = (*bil1);
    (*bil1) = (*bil2);
    (*bil2) = temp;
}
```

Bisa dilihat pada gambar disamping, pada bagian atas tidak terdapat pendeklarasian prosedur. Program tetap berjalan dengan baik dan benar tanpa ada error. Namun, akan muncul tulisan berwarna orange yang berisi warning pada bagian bawah. Hal tersebut dapat dihilangkan dengan melakukan pendeklarasian terlebih dahulu.

Compiler (2) | Resources | Compile Log | Debug | Find Results | Close

Line	Col	File	Message
17	6	D:\Document\1Tentor Abi\Private\Latihan Pros 2\Latiha...	[Warning] conflicting types for 'bil'
12	2	D:\Document\1Tentor Abi\Private\Latihan Pros 2\Latiha...	[Note] previous implicit declaration of 'bil' was here

- **Pendefinisian Prosedur**

```c
void <namaProsedur> (<Parameter1, Paramter2, ...>)
{
  <body> /*Berisi code yang menjalankan suatu aksi
        ketika prosedur ini dipanggil */
}
```

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

Pendefinisian prosedur biasa dilakukan pada margin bawah program atau dibagian bawah main program. Pada saat menerapkan konsep ADT di semester 3 nanti, pendefinisian prosedur akan dilakukan di file source.

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

// Pendeklarasian Prosedur
void bil(int *bil1, int *bil2);

int main(int argc, char *argv[]) {
    int bil1 = 3, bil2 = 5;

    printf("\tSebelum Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    bil(&bil1, &bil2); // Pemanggilan Prosedur

    printf("\n\tSesudah Penukaran");
    printf("\n\tBilangan 1 : %d", bil1);
    printf("\n\tBilangan 2 : %d", bil2);

    return 0;
}

// Pendefinisian Prosedur
void bil(int *bil1, int *bil2) // <--- Terletak Pada Margin Bawah Program
{
  int temp;

    temp = (*bil1);
    (*bil1) = (*bil2);
    (*bil2) = temp;
}
```

Merupakan pendefinisian dari prosedur bil yang dimana program akan melakukan pertukaran nilai dari bil1 ke bil2 dan menampilkan perubahannya.

- **Pemanggilan Prosedur**

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

Setelah kita melakukan Pendeklarasian dan pendefinisian, prosedur tentu tidak akan berguna jika tidak kita panggil ke main program. Tentu 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 sesuai dengan nama prosedur yang sudah di deklarasikan dan definisikan.
2. **Jumlah Parameternya harus sama** dengan prosedur yang dipanggil.
3. **Tipe data variable** pada parameter yang ada di main program harus sama dan sesuai dengan penempatannya dengan tipe data variable pada parameter pada saat prosedur dideklarasikan dan didefinisikan.
4. **Jika parameter di tempel asteris / pointer(*)** saat pendeklarasian & pendefinisian, maka saat pemanggilan di parameter tersebut harus ditempel dengan "&".

Contoh Pemanggilan Prosedur:

```
7  int main(int argc, char *argv[]) {
8      int bilangan1 = 3, bilangan2 = 5, tampung;
9
10     printf("\tSebelum Penukaran\n");
11     printf("\nBilangan 1 : %d", bilangan1);
12     printf("\nBilangan 2 : %d", bilangan2);
13
14     tukarAngka(&bilangan1, &bilangan2, tampung); //Pemanggilan Prosedur
15
16     printf("\n\n\tSetelah Penukaran\n");
17     printf("\nBilangan 1 : %d", bilangan1);
18     printf("\nBilangan 2 : %d", bilangan2);
19     return 0;
20 }
21
22 //Pendefinisian Prosedur
23 void tukarAngka(int *bil1, int *bil2, int temp)
24 {
25     temp = *bil1;
26     *bil1 = *bil2;
27     *bil2 = temp;
28 }
29
```

Pada contoh diatas, dapat dilihat bahwa parameter yang ditempel dengan asteris(*), pada saat pemanggilan prosedur wajib ditempel "&". Perlu diingat bahwa **Nama Variabel** pada parameter yang melekat pada prosedur **tidak harus sama,** tetapi **tipe data balikan variable harus sama.**

- **Parameter**

Parameter adalah informasi atau variable yang dibutuhkan untuk menjalankan aksi dari suatu prosedur. Sebagai programmer kita harus bisa menentukan sendiri variabel apa saja yang dibutuhkan di dalam parameter agar prosedur dapat berjalan sesuai dengan aksi yang diinginkan. Ketika prosedur di jalankan dan ternyata masih ada kekurangan parameter, maka

program akan mengalami error dan tidak bisa dijalankan. Di dalam prosedur, terdapat dua bentuk parameter:

1) **Parameter Input** = parameter sebagai masukan ke dalam prosedur untuk menjalankan suatu aksi.
2) **Parameter Output** = merupakan parameter di dalam prosedur yang nilainya akan dibawa keluar ke dalam variable yang ada di main program (Memiliki ciri khas, yaitu terdapat pointer / asteris "*" yang menempel pada parameter)

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

//Pendeklarasian Prosedur
                    //  Parameter       Parameter       Parameter
                    //  Output          Input           Input
void luasPersegiPanjang(float *luas, float panjang, float lebar);

int main(int argc, char *argv[]) {
    float luasPP;
    float pjg = 5;
    float lbr = 3;

    //nilai panjang dan lebar di terima dari variabel pjg dan lbr yang ada pada main program.
    luasPersegiPanjang(&luasPP, pjg, lbr); //pada main, tanda asteris diganti menjadi tanda ampersand " & ".

                //Pada pemanggilan prosedur, nama pada parameter tidak harus sama
                //  dengan yang ada pada pendeklarasian dan pendefinisian prosedur.
    return 0;
}

//Pendefinisian Prosedur
                    //  Parameter       Parameter       Parameter
                    //  Output          Input           Input
void luasPersegiPanjang(float *luas, float panjang, float lebar)
{
    *luas = panjang * lebar; //pada body prosedur apabila variabel yang memiliki tanda asteris " * "
}                           // tetap di tulis menggunakan tanda asterisnya jika tidak akan mengakibatkan error.

```

Jenis Parameter

Parameter pada prosedur dibedakan menjadi 2 jenis yaitu **Parameter Formal** dan **Parameter Aktual. Parameter Formal** biasanya terletak di luar main program atau bisa dikatakan parameter pada saat pendeklarasian dan pendefinisian prosedur sedangkan **Parameter Aktual** biasanya yang terdapat di dalam main program atau pada saat pemanggilan prosedur.

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

//                      |           Parameter Formal            |
void luasPersegiPanjang(float *luas, float panjang, float lebar);

int main(int argc, char *argv[]) {
    float luasPP;
    float pjg = 5;
    float lbr = 3;

    //                | Parameter Aktual |
    luasPersegiPanjang(&luasPP, pjg, lbr);
    return 0;
}

//                      |           Parameter Formal            |
void luasPersegiPanjang(float *luas, float panjang, float lebar)
{
    *luas = panjang * lebar;
}
```

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 uamg sama.
3. Jumlah Parameter Aktuan dan Parameter Formal harus sama.
4. Urutan Parameter Formal harus sama dengan Parameter Aktual.

Kategori Prosedur Berdasarkan Parameter

1. **Naive**

   Prosedur ini adalah prosedur yang tidak memiliki parameter sama sekali.

```c
void tampilMenu()
{
    printf("\t---=== [ MENU BELANJA ] ===---");
    printf("\n[1]. Login");
    printf("\n[2]. Input Belanja");
    printf("\n[3]. Hapus Belanja");
    printf("\n[4]. Bayar Belanja");
    printf("\n[0]. EXIT");
}
```

2. **Semi-Naive Input**

Prosedur ini hanya memiliki parameter input dan menghasilkan outpur yang dikeluarkan melalui Standard Input/Output. Pada kategori ini, inputan akan berada pada main program, sedangkan code output akan berada di dalam prosedur. Hasil operasi di dalam prosedur ini juga tidak akan merubah apa-apa di main program.

```c
void modulus(int bilangan, int pembagi)
{
  int hasil = bilangan % pembagi;

  printf("\nHasil Modulus adalah %d", hasil);
}
```

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 berasa di dalam prosedur, sedangkan code output berada di main program.

```c
void modulus(int *hasil) // isi parameter hanya ada parameter output
{
  int angka1, angka2;

  printf("Input Angka Pertamna  : "); scanf("%d", &angka1);
  //printf dan scanf di dalam prosedur
  printf("Input Angka Kedua     : "); scanf("%d", &angka2);
  //printf dan scanf di ddalam prosedur

  *hasil = angka1 % angka2; // nilai akan berubah di dalam main program
}
```

4. **Nett Effect**

Prosedur Nett Effect tidak menggunakan Standard Input/Output. Prosedur ini adalah kategori prosedur Input/Output yang paling direkomendasikan. Nett Effect tidak memperbolehkan adanya inputan dan output berbentuk printf ataupun scanf di dalam prosedur. (Input dan Output berada pada main program).

```c
void hitungLuasSegitiga(int *luas, int alas, int tinggi)
  //parameter berisi parameter input dan output. alas dan tinggi akan
  //diperoleh dari main program dan masuk ke dalam prosedur sedangkan
  //hasil akan dikeluarkan dari prosedur dan dikemablikan ke main.
{
  *luas = (alas * tinggi) / 2;
}
```

- **Pointer / Asteris (*)**

Seperti yang sudah di bahas sebelumnya, ciri dari parameter output adalah adanya binta atau asteris / pointer yang menempel pada parameter formal. Penggunaan asteris/pointer akan membuat variable pada parameter aktual nilainya ikut berubah menyesuaikan nilai dari variabel pada parameter formal.

Berbeda Ketika kalian tidak menempelkan asteris / pointer pada parameter formal, maka segala perubahan nilai pada parameter formal selama prosedur dieksekusi tidak akan mengakibatkan perubahan pada parameter actual. Oleh karena itu, Ketika teman-teman ingin mengubah value suatu parameter pada prosedur, diwajbkan untuk menggunakan asteris/pointer pada parameter formal terkait yang ingin datanya diubah.

Agar kalian lebih memahami terkait konsep pointer/asteris, kalian saya sarankan untuk mencoba code disamping dan melakukan modifikasi mandiri terkait pointer / asteris.

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

void penjumlahanPakaiPointer(int *hasil, int angka1, int angka2);
void penjumlahanTanpaPointer(int hasil, int angka1, int angka2);

int main(int argc, char *argv[]) {
    int hasil = 0, a1, a2;

    printf("Masukkan Angka Pertama  : "); scanf("%d", &a1);
    printf("Masukkan Angka Kedua    : "); scanf("%d", &a2);

    penjumlahanTanpaPointer(hasil, a1, a2);
    printf("\n\n\tHasil Dari Prosedur Tanpa Pointer : %d\n", hasil);
    penjumlahanPakaiPointer(&hasil, a1, a2);
    printf("\n\n\tHasil Dari Prosedur Pakai Pointer : %d\n", hasil);
    return 0;
}

void penjumlahanPakaiPointer(int *hasil, int angka1, int angka2)
{
    *hasil = angka1 + angka2;
    printf("\nHasil Dalam Prosedur Dengan Pointer : %d", *hasil);
}

void penjumlahanTanpaPointer(int hasil, int angka1, int angka2)
{
    hasil = angka1 + angka2;
    printf("\nHasil Dalam Prosedur Tanpa Pointer : %d", hasil);
}
```

```
Masukkan Angka Pertama  : 3
Masukkan Angka Kedua    : 2

Hasil Dalam Prosedur Tanpa Pointer : 5

        Hasil Dari Prosedur Tanpa Pointer       : 0

Hasil Dalam Prosedur Dengan Pointer : 5

        Hasil Dari Prosedur Pakai Pointer       : 5
```

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

Berbeda dengan prosedur tanpa asteris yang value dari hasil tertampil 0 di main sedangkan hanya berubah di dalam prosedur.

- **Prosedur Rekursif**

Ketika kalian sudah masuk ke modul prosedur, tentunya kalian sudah mengerti konsep perulangan di dalam bahasa pemrograman. Namun, tahukah 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 memanfaarkan hal tersebut, kitab isa melakukan perulangan dengan memanggil Kembali prosedur itu sendiri dan mengubah sedikit parameternya saat pemanggilan. Konsep tersebut disebut dengan konsep prosedur rekursif. Untuk rekursif, terangkan prosedur rekursif harus mengandung 2 komponen, yaitu:

- Basis (yang bertugas menghentikan rekursif)
- Rekurens (bagian yang memanggil diri sendiri, oleh sebab itu dinamakan rekursif)

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

void visualisasi(int alas, int tinggi, int i, int j);

int main(int argc, char *argv[]) {

    visualisasi(10, 10, 0, 0);

    return 0;
}

void visualisasi(int alas, int tinggi, int i, int j){

    if(i > tinggi){ //kondisi berhenti
        return;
    }else if(j > alas){ //Kondisi ketika value j sudah lebih besar dari alas
        printf("\n");
        visualisasi(alas, tinggi, i+=1, 0); //Memanggil kembali prosedur visualisasi
                                            //hanya saja kita lakukan penambahan 1 pada parameter i
                                            //dan ubah value dari parameter j menjadi 0
                                            //modifikasi prosedur ini konsepnya sama dengan perulangan for(i = 0; i < tinggi; i++)
    }else{
        if(j < i){
            printf("* ");
        }
        visualisasi(alas, tinggi, i, j+=1); //Memanggil kembali prosedur visualisasi
                                            //hanya saja kita lakukan penambahan 1 pada parameter j
                                            //modifikasi prosedur ini konsepnya sama dengan perulangan for(j = 0; j < alas; j++)
    }
}
```

Code di atas merupakan contoh dari implementasi prosedur rekursif 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 rekursif memanfaatkan “If-else” sebagai pemberhenti rekursif dan pengkondisian suatu alur code yang diinginkan.

***Note:**

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

A!: “Ya memang, ini sebenernya materi semester 3. Namun, saya berikan pengantar disini karena di tahun kakak kemarin konsep ini dikeluarkan di UAS

Q2 : ”Kak, ini keluar di UGD ga ya nanti?”

A2 : ”Gatau yahh, liat nanti aja tergantung mood.”

GUIDED

1. Buatlah project baru dengan penamaan **GD7_X_YYYYY (X = Kelas; Y = 5 Digit Akhir NPM)**
2. Berikut Detail Studi Kasusnya;

Shael diminta oleh Kak Shiel untukmembuat program dalam bahasa C untuk tokonya. Kak Shiel meminta Shael untuk hanya sampai prosedur 1 dalam pembuatan program tersebut. Karena Shael adalah programmer yang handal, dia mengetahui bahwa terdapat 4 kategori prosedur dengan ciri khasnya masing-masing, antara lain **Naive, Semi-Naive Input, Semi-Naive Output,** dan **Nett Effect.** Kak Shiel meminta Shael untuk membuatkan program dengan menerapkan semua prosedur tersebut. Berikut adalah menu yang diminta oleh Kak Shiel:

- **TAMPIL MENU (GUNAKAN PROSEDUR NAIVE)**

Pada bagian ini, Kak Shiel meminta Shael untuk emmbuat tampilan menu yang tersedia di dalam program yang diinginkan oleh Kak Shiel. Ada 8 menu yang diinginkan oleh Kak Shiel antara lain:

1. **MENU 1 (INPUT BARANG)**

   Menu ini untuk menginputkan nama barang, jumlah, dan harga yang didapatkan oleh pembeli. Program juga akan memberikan error handling dan meminta inputan ulang jika inputan user tidak sesuai. Program akan menampilkan inputan berhasil apabila semua inputan user benar. **(GUNAKAN PROSEDUR SEMI-NAIVE OUTPUT)**

2. **MENU 2 (TAMPIL BARANG)**

   Pada menu ini, program akan menampilkan nama, jumlah, total harga yang didapatkan setelah melakukan input barang. **(GUNAKAN PROSEDUR SEMI-NAIVE INPUT).**

3. **MENU 3 (UBAH NAMA BARANG)**

   Pada menu ini, mirip dengan menu 1 yang meminta inputan nama barang, jumlah. Pada menu ini program hanya akan menghitung harga yang didaparkan. **(GUNAKAN PROSEDUR NETT EFFECT).**

4. **MENU 4 (PEMBAYARAN)**

   Pada menu ini, program akan melakukan penghitungan berdasarkan barang yang sudah diinputkan dan akan melakukan pembayara. Pada menu ini program hanya akan menghitung hasil kembalian yang didapatkan dari Inputan uang yang dimasukkan user. **(GUNAKAN PROSEDUR NETT EFFECT).**

NOTE : Untuk Menu 5 – 8, kalian coba ber eksperimen untuk membuat menu sendiri menggunakan 4 jenis prosedur, tema bebas boleh melanjutkan dari yang sudah ada atau membuat sendiri. Buat 4 menu tersebut di project baru dan diberi nama **GD7_X_YYYY_4MENU (X = KELAS, Y = 5 DIGIT AKHIR NPM)** yang nanti di zip dalam satu file dengan yang sudah dibuat sebelumnya. **TIDAK MEMBUAT 4 MENU INI = TIDAK MENGERJAKAN GD.**

5. **MENU 0 (EXIT)**

   Tulisakan Nama Lengkap, Kelas, dan NPM kalian.

Code

Deklarasi Prosedur

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

typedef char string[100];

void tampilMenu();
//Pendeklarasian Prosedur Naive
void inputBarang(string *nama, int *jumlah, float *harga);
//Pendeklarasian Prosedur Semi Naive-Output
void tampilBarang(string nama, int jumlah, float harga);
//Pendeklarasian Prosedur Semi Naive-Input
void hitungTotalHarga(float *harga, string nama, int jumlah);
//Pendeklarasian Prosedur Nett Effect
void hitungKembalian(float *kembalian, float price, float uang);
//Pendeklarasian Prosedur Nett Effect
```

Main Program

```
int main(int argc, char *argv[]) {
    string barang;
    int jml;
    float price = 0, uang, kembalian;
    int menu;

    do
    {
        tampilMenu(); //Pemanggilan Prosedur Naive
        scanf("%d", &menu);

        switch(menu)
        {
            case 1:
                printf("\n\t------ [ INPUT BARANG ] ===---\n");
                inputBarang(&barang, &jml, &price); //Pemanggilan Prosedur Semi-Naive Output
                printf("\n\t[+] Berhasil Input Barang [+]\n");
            break;

            case 2:
                printf("\n\t------ [ TAMPIL BARANG ] ===---\n");
                tampilBarang(barang, jml, price); // Pemanggilan Prosedur Semi-Naive Input
                printf("\n\t[~] Semua Barang Sudah Ditampilkan [~]\n");
            break;

            case 3:
                printf("\n\t------ [ UBAH BARANG ] ===---\n");
                do
                {
                    printf("\nMasukkan Nama Barang Baru : "); fflush(stdin); gets(barang);
                    if(strcmpi(barang, "Gems") != 0 && strcmpi(barang, "Monthly Pass") != 0 &&
strcmpi(barang, "Double Reward Pass") != 0)
                    {
                        printf("\n\t[!] Nama Barang Hanya Boleh | Gems | Monthly Pass | Double Reward
Pass | [!]\n");
                    }
                }while(strcmpi(barang, "Gems") != 0 && strcmpi(barang, "Monthly Pass") != 0 &&
strcmpi(barang, "Double Reward Pass") != 0);

                do
                {
                    printf("\nMasukkan Jumlah Barang        : "); scanf("%d", &jml);
                    if(jml < 0)
                    {
                        printf("\n\t[!] Jumlah Barang Tidak Boleh Negatif [!]\n");
                    }
                }while(jml < 0);

                hitungTotalHarga(&price, barang, jml); //Pemanggilan Prosedur Nett Effect
                printf("\nHarga Yang Harus Dibayar  : %.2f", price);
            break;

            case 4:
                printf("\n\t------ [ PEMBAYARAN ] ===---\n");
                tampilBarang(barang, jml, price); //Pemanggilan Prosedur Semi-Naive Input
                do
                {
                    printf("\n~ Masukkan Jumlah Uang Yang Ingin Dibayarkan : Rp"); scanf("%f",
&uang);
                    if(uang < 0 || uang < price)
                    {
                        printf("\n\t[!] Uang Tidak Cukup [!]\n");
                    }
                }while(uang < 0 || uang < price);
                hitungKembalian(&kembalian, price, uang); //Pemanggilan Prosedur Nett Effect
                printf("\n\nKembalian Yang didapatkan : Rp%.2f", kembalian);
            break;

            default:
                printf("\n\t[!] Menu Tidak Tersedia [!]\n");
            break;

            case 0:
                printf("\n\t[ ~ NAMA PRAKTIKAN - KELAS - NPM ~ ] Dari Program . . .");
                printf("\n\t=== Prosedur Itu Mudah >.< ===");
            break;
        }getch();
    }while(menu != 0);
    return 0;
}
```

Pendefinisian Prosedur

```
void tampilMenu() //Pendefinisian Prosedur Tampil Menu (NAIVE)
{
    system("cls");
    printf("\t------ [ GUIDED PROSEDUR ] ---------\n");
    printf("\n[1]. Input Barang");
    printf("\n[2]. Tampil Barang");
    printf("\n[3]. Ubah Barang");
    printf("\n[4]. Pembayaran");
    printf("\n[0]. Keluar Program");
    printf("\n>>> ");

    /*
        Pada prosedur tampilMenu menggunakan Naive karena pada prosedur ini tidak perlu
        menampilkan value dari suatu variabel tertentu, sehingga tidak diperlukan adanya parameter
    */
}

void inputBarang(string *nama, int *jumlah, float *harga) //Pendefinisian Prosedur inputBarang
{
    do
    {
        printf("\nMasukkan Nama Barang  : "); fflush(stdin); gets(*nama);
        if(strcmpi(*nama, "Gems") != 0 && strcmpi(*nama, "Monthly Pass") != 0 && strcmpi(*nama,
"Double Reward Pass") != 0)
        {
            printf("\n\t[!] Nama Barang Hanya Boleh | Gems | Monthly Pass | Double Reward Pass |
[!]\n");
        }
    }while(strcmpi(*nama, "Gems") != 0 && strcmpi(*nama, "Monthly Pass") != 0 && strcmpi(*nama,
"Double Reward Pass") != 0);
  //Meminta Inputan terus menerus selama inputan belum benar.

    do
    {
        printf("\nMasukkan Jumlah Barang    : "); scanf("%d", &(*jumlah));
        if(*jumlah < 0)
        {
            printf("\n\t[!] Jumlah Barang Tidak Boleh Negatif [!]\n");
        }
    }while(*jumlah < 0);
  //Meminta Inputan User terus menerus selama inputan belum benar.

    if(strcmpi(*nama, "Gems") == 0) *harga = 10000 * (*jumlah);
    //Jika inputan user adalah "Gems", maka harganya akan menjadi 10000 dikalikan dengan jumlahnya
    else if(strcmpi(*nama, "Monthly Pass") == 0) *harga = 75000 * (*jumlah);
  /*
    Jika inputan user adalah "Monthly Pass", maka harganya akan menjadi 75000 dikalikan
    dengan jumlahnya
  */
    else *harga = 50000 * (*jumlah);
  /*
    Jika Inputan User "Double Reward Pass", maka harga akan menjadi 50000 dikalikan
    dengan jumlahnya. Alasan tidak dibuat else if karena sudah pasti inputan user hanyalah diantara
    "Gems", "Mothly Pass" atau "Double Reward Pass" karena jika diluar itu akan terkena error
    handling pada saat meminta inputan.
  */

  /*
    Bisa dilihat pada prosedur ini menggunakan Semi Naive-Output karena inputan user akan dijalankan
    di dalam prosedur dan dikemablikan nilainya ke main program.
  */
}

void tampilBarang(string nama, int jumlah, float harga)
{
    printf("\n=======================================\n");
    printf("\nNama Barang   : %s", nama);
    printf("\nJumlah Barang : %d", jumlah);
    printf("\nHarga Barang  : %.2f", harga);
    printf("\n\n=======================================\n");
  /*
    Pada menu ini menggunakan prosedur Semi-Naive Input yang didapatkan dari main dan nilai
    sebelumnya dikeluarkan dari Menu 1 ke main program dan di gunakan nilainya di dalam prosedur saat
    ini.
  */
}

void hitungTotalHarga(float *harga, string nama, int jumlah)
{
    if(strcmpi(nama, "Gems") == 0) *harga = 10000 * jumlah;
    else if(strcmpi(nama, "Monthly Pass") == 0) *harga = 75000 * jumlah;
    else *harga = 50000 * jumlah;

  /*
    Pada menu ini mirip dengan yang ada pada Prosedur inputBarang. namun, pada prosedur ini
    menerapkan prinsip nett effect yang dimana tidak ada printf ataupun scanf di dalamnya. Pada
    prosedur ini akan mengembalikan nilai harga ke dalam main program berdasarkan kondisi nama dan
    jumlah yang didapatkan dari main program.
  */
}

void hitungKembalian(float *kembalian, float price, float uang)
{
    *kembalian = uang - price;

  //Menghitung kembalian berdasarkan inputan uang dan harga yang harus dibayarkan dari main program.
}
```

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. Jika membuat project baru untuk 4 menu yang diminta, kedua project di zip menjadi satu dengan nama **GD7_X_YYYYY.zip (X = KELAS, Y = 5 DIGIT TERAKHIR NPM).**
   CONTOH:
   GD7_A_12345_4MENU (FOLDER PROJECT)
   GD7_A_12345 (FOLDER PROJECT)
   KEDUA FOLDER DI ZIP MENJADI GD7_A_12345.zip
5. Kesalahan format penamaan file akan dilakukan pengurangan nilai UGD **(-10)**
6. Tindak Plagiasi tidak akan diberikan toleransi, terutama di 4 Menu yang diminta **(DIANGGAP TIDAK MENGERJAKAN GUIDED)**
7. Jika masih ad ayang ingin ditanyakan, bisa langsung kontak ke:
   a. Teams : Matius Abimanyu Dwi Ariyanto (230712510@students.uajy.ac.id)
   b. WA : https://wa.me/6287881555107 (Recommend)

*Note:
Pelajari & hapalkan penggunaan rand(random) karena sudah aku spill disini, kemungkinan akan muncul saat UGD nanti dan apabila bertanya mengenai random saat UGD, tidak akan di jawab oleh asisten ya.

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