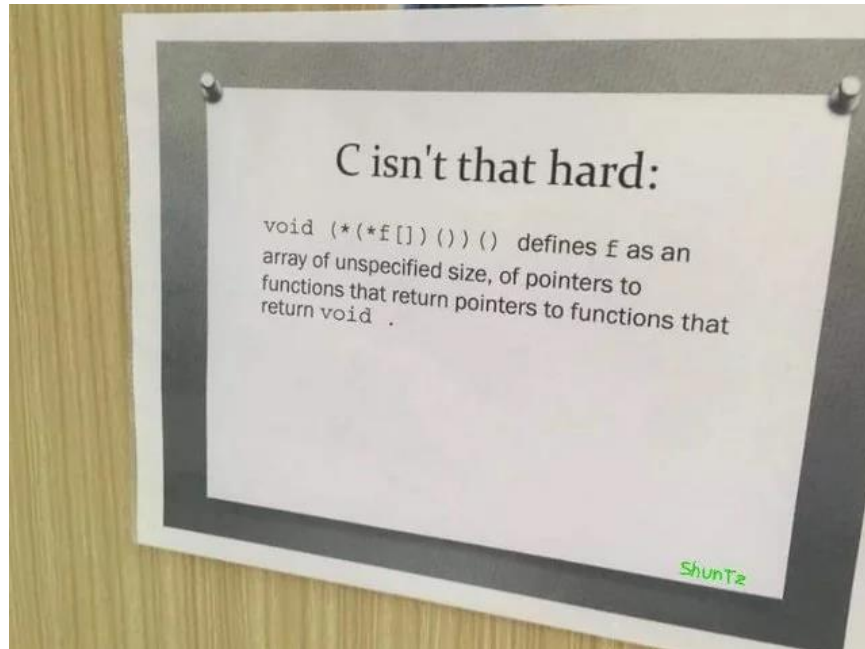


MODUL 8

PROSEDUR 2



A. Tujuan Praktikum

1. Memahami konsep parameter secara lebih mendalam.
2. Memahami konsep parameter input, output, dan input/output.
3. Memahami konsep passing parameter by value dan by reference.
4. Memahami konsep prosedur dalam prosedur.

B. Prosedur

Prosedur merupakan serangkaian instruksi atau tindakan yang dijalankan dalam sebuah program komputer untuk menyelesaikan tugas tertentu (bersifat spesifik). Terdapat beberapa tahap untuk membuat sebuah prosedur, yaitu :

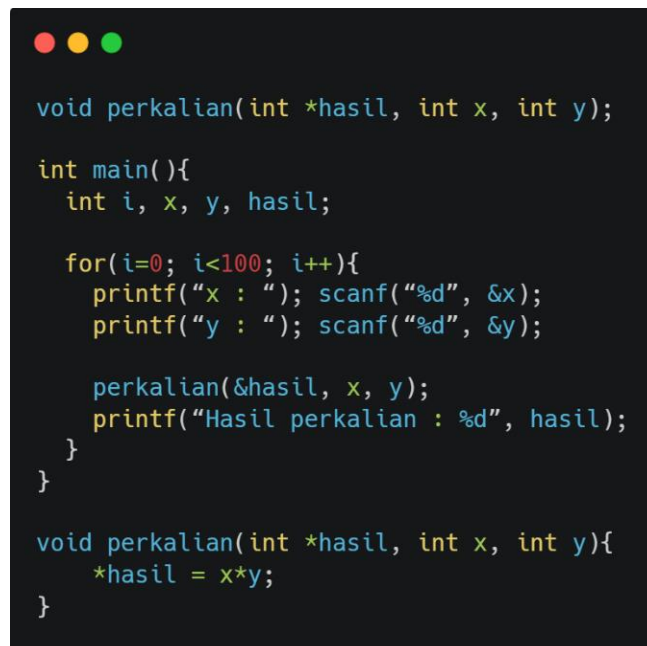
1. Analisis masalah: identifikasikan masalah yang harus dipecahkan
Contoh : Diberikan soal untuk menghitung perkalian dengan meminta inputan pengguna 100 kali.

2. Desain Solusi : merencanakan urutan instruksi untuk menyelesaikan masalah

Contoh : Untuk mengatasi masalah tersebut, daripada menuliskan rumus perkalian variabel sebanyak 100 kali untuk setiap perhitungan, lebih baik membuat sebuah prosedur perkalian yang dapat digunakan berkali kali.

3. Implementasi :

Implementasikan desain solusi tersebut dalam bahasa pemrograman yang digunakan.



```
void perkalian(int *hasil, int x, int y);

int main(){
    int i, x, y, hasil;

    for(i=0; i<100; i++){
        printf("x : "); scanf("%d", &x);
        printf("y : "); scanf("%d", &y);

        perkalian(&hasil, x, y);
        printf("Hasil perkalian : %d", hasil);
    }
}

void perkalian(int *hasil, int x, int y){
    *hasil = x*y;
}
```

Gambar 1. Prosedur perkalian

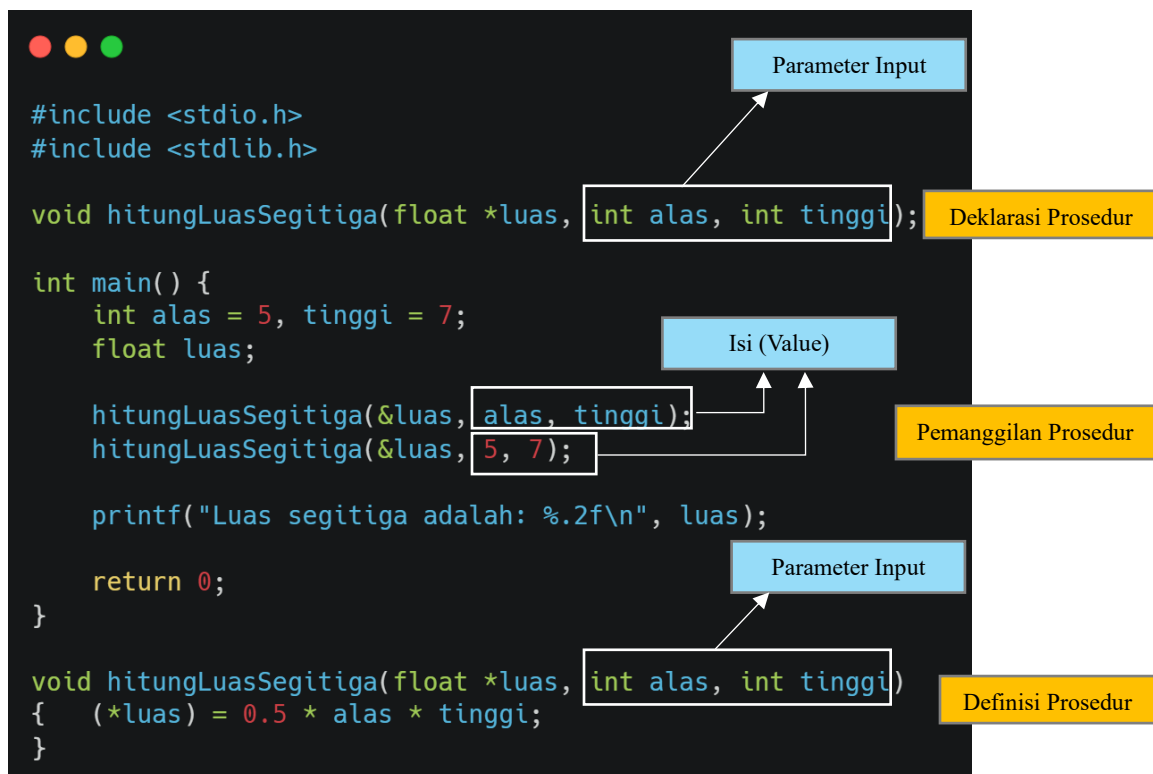
Untuk studi kasus diatas memang tidak terlalu terlihat manfaat dari prosedur karena jumlah baris dalam prosedur `perkalian` tidak terlalu banyak. Tujuan utama dalam pembuatan prosedur yaitu untuk mengorganisir dan membagi kode menjadi bagian yang lebih kecil dan terorganisir. Selain itu prosedur juga dimanfaatkan untuk melaksanakan program yang akan dijalankan berulang kali.

C. Jenis Parameter

1. Parameter Input

Parameter input adalah Parameter yang isi (*value*) nya sudah jelas terdefinisi ketika dilakukan pemanggilan prosedur. Isi (*value*) dari parameter ini digunakan sebagai masukan untuk prosedur. Isi (*value*) dari parameter ini harus **tidak mengalami perubahan** selama pemrosesan prosedur (tidak akan berubah baik saat memasukkan kedalam prosedur maupun mengeluarkan hasil keluar dari prosedur).

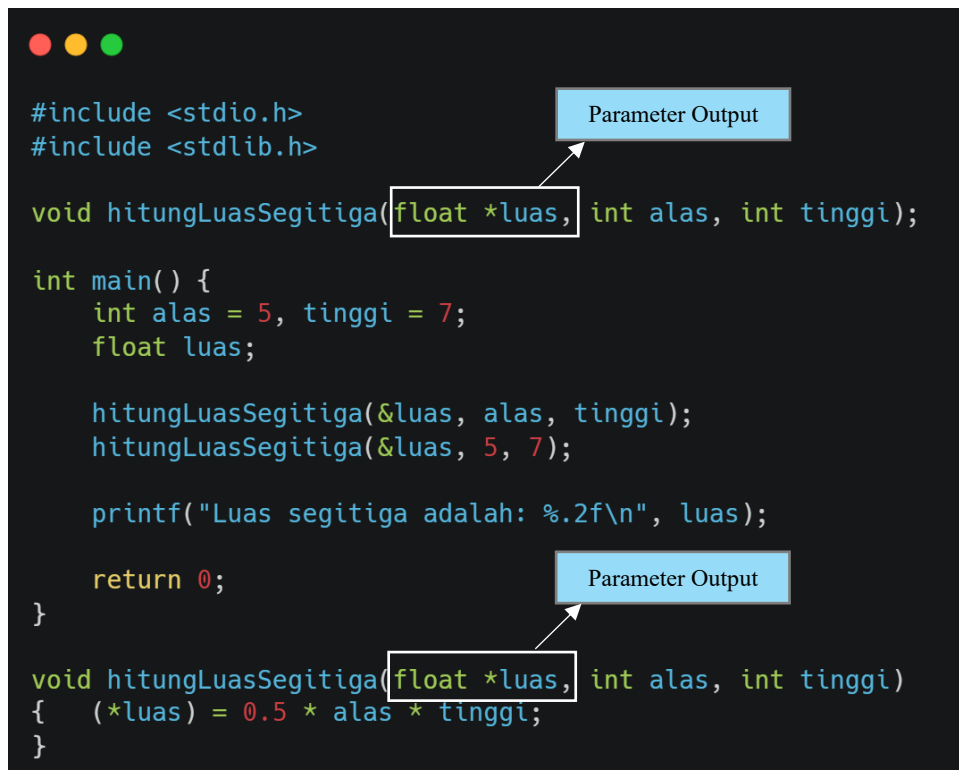
Dalam suatu assignment dalam body prosedur, parameter ini biasanya terletak disebelah parameter output (akan dibahas selanjutnya). Perhatikan contoh berikut.



Gambar 2. Program untuk menghitung luas segitiga

2. Parameter Output

Parameter output adalah Parameter target yang hendak diubah nilainya atau parameter yang menampung hasil keluaran dari prosedur. Parameter ini mengalami perubahan dan efek netto dari prosedur (menjadi tujuan pengubahan). Parameter ini memiliki isi (value) yang sudah jelas ataupun belum jelas, yang pasti parameter ini akan diubah di dalam prosedur. Dalam suatu assignment di dalam body prosedur, parameter ini tertelak disebelah kiri. Salah satu ciri khas dari parameter output dalam prosedur yaitu terdapat *pointer* yang menempel pada nama variabel nya. Perhatikan contoh berikut.



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

void hitungLuasSegitiga(float *luas, int alas, int tinggi);

int main() {
    int alas = 5, tinggi = 7;
    float luas;

    hitungLuasSegitiga(&luas, alas, tinggi);
    hitungLuasSegitiga(&luas, 5, 7);

    printf("Luas segitiga adalah: %.2f\n", luas);

    return 0;
}

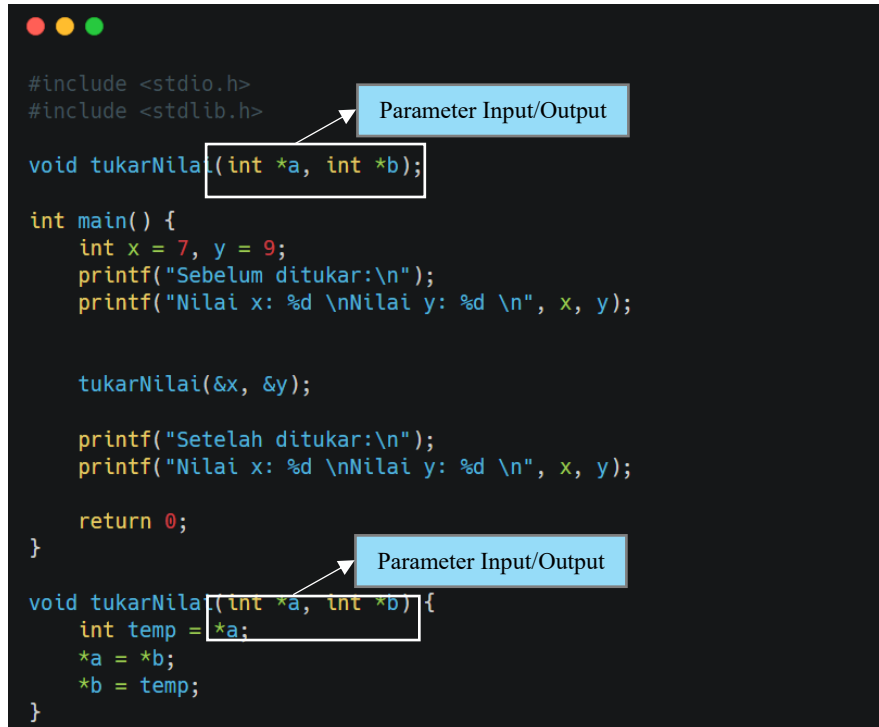
void hitungLuasSegitiga(float *luas, int alas, int tinggi)
{
    (*luas) = 0.5 * alas * tinggi;
}
```

Gambar 3. Program untuk menghitung luas segitiga

Dalam bahasa C, prosedur tidak dapat mengembalikan nilai (return) secara langsung. Namun, dengan menggunakan parameter output, kita dapat mengubah nilai dari variabel-variabel yang diberikan saat memanggil prosedur.

3. Parameter Input/Output

Parameter output yang juga sekaligus menjadi parameter input (sebagai masukan sekaligus keluaran bagi prosedur tersebut). Parameter ini pastilah sudah jelas isinya ketika memasuki prosedur, karena dia juga berperan sebagai parameter input. Untuk lebih memahaminya, perhatikan contoh code dibawah ini.



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

void tukarNilai(int *a, int *b);

int main() {
    int x = 7, y = 9;
    printf("Sebelum ditukar:\n");
    printf("Nilai x: %d \nNilai y: %d \n", x, y);

    tukarNilai(&x, &y);

    printf("Setelah ditukar:\n");
    printf("Nilai x: %d \nNilai y: %d \n", x, y);

    return 0;
}

void tukarNilai(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Gambar 4. Program untuk menukar nilai dua variabel

Dalam contoh prosedur `tukarNilai`, parameter `*a` dan `*b` adalah parameter input yang menerima alamat dari variabel-variabel yang akan ditukar nilainya. Disisi lain, `*a` dan `*b` adalah parameter output yang akan mengubah nilai yang ada pada main.

4. Pendefinisian dan Pemanggilan Prosedur

- **Pendefinisian Prosedur**

Dalam mendefinisikan prosedur, ada struktur yang harus diperhatikan. Adapun struktur dalam pendefinisian prosedur antara lain **nama prosedur**, **parameter**, dan **body prosedur**. Nama prosedur ini bersifat *case sensitive*, artinya nama prosedur ketika

dideklarasikan, didefinisikan, dan dipanggil haruslah sama. Parameter prosedur bersifat opsional, artinya bisa ada dan bisa tidak. Bagian body prosedur sendiri berisi logika program yang dimiliki oleh prosedur itu sendiri.

```
void namaProsedur(tipeData1 namaVariabel1, tipeData2 namaVariabel2...){ //nama prosedur dan parameter
    body //bagian yang berisi logika code
}
```

Gambar 5. Struktur definisi prosedur

```
void faktorial(int bilangan, int *hasil) {
    int i;
    (*hasil) = bilangan;
    for(i=bilangan-1; i>1; i--){
        (*hasil) *= i;
    }
}
```

Gambar 6. Contoh definisi prosedur

- **Pemanggilan Prosedur**

Pemanggilan prosedur dapat dilakukan pada main program maupun pada prosedur lain. Struktur pemanggilan prosedur sendiri yaitu `namaProsedur(namaVariabel1, namaVariabel2, ...)`; Adapun beberapa hal penting yang harus diperhatikan dalam pemanggilan prosedur antara lain:

1. **Nama Prosedur** yang dipanggil **harus sama** dengan nama Prosedur yang telah dideklarasikan 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. Urutan parameter pada saat pendefinisian dan pemanggilan prosedur harus sama. Jika urutan parameter berbeda maka akan memungkinkan terjadinya error pada saat kompilasi atau program menghasilkan output yang tidak sesuai harapan.

5. Jika Parameter **ditempel asteris / pointer (*)** saat pendeklarasian dan pendefinisian, maka saat pemanggilan di parameter tersebut harus **ditempel dengan "&"**.
6. Nama variabel dalam parameter prosedur pada saat pendefinisian dan pemanggilan pada main program **tidak harus sama / boleh berbeda**.



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


//Deklarasi dan Definisi Prosedur
void pembagian(int pembilang, int penyebut, float *hasil) {
    (*hasil) = pembilang / penyebut;
}

int main(int argc, char *argv[]) {
    int pembilang = 8, penyebut = 4;
    float jawaban;

    //Pemanggilan Prosedur
    pembagian(penyebut, pembilang, &jawaban);
    //Pemanggilan prosedur tersebut tidak tepat karena urutan parameter pada saat
    //pendefinisian dan pemanggilan prosedur berbeda. Jika dijalankan, program akan
    //menghasilkan output yang tidak sesuai harapan
    printf("Hasil pembagian adalah %.1f\n", jawaban);

    return 0;
}
```

Gambar 7. Contoh pemanggilan prosedur yang tidak tepat



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

//Deklarasi dan Definisi Prosedur
void pembagian(int pembilang, int penyebut, float *hasil) {
    (*hasil) = pembilang / penyebut;
}

int main(int argc, char *argv[]) {
    int pembilang = 8, penyebut = 4;
    float jawaban;

    //Pemanggilan Prosedur
    pembagian(pembilang, penyebut, &jawaban);

    printf("Hasil pembagian adalah %.1f\n", jawaban);

    return 0;
}
```

Gambar 8. Contoh pemanggilan prosedur yang benar

```

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

//Deklarasi dan Definisi Prosedur
void pembagian(int base, int pembagi, float *hasil) {
    (*hasil) = base / pembagi;
}

int main(int argc, char *argv[]) {
    int pembilang = 8, penyebut = 4;
    float jawaban;

    //Pemanggilan Prosedur
    pembagian(pembilang, penyebut, &jawaban);

    printf("Hasil pembagian adalah %.1f\n", jawaban);

    return 0;
}

```

Gambar 9. Contoh pemanggilan prosedur yang benar

5. Perbedaan *passing parameter by value* dan *passing parameter by reference*

- **Passing parameter by value**

Pada passing parameter by value, nilai dari parameter akan disalin ke dalam variabel baru yang berada di dalam prosedur. Ini artinya, ketika nilai parameter diubah di dalam prosedur, nilai variabel asli di luar prosedur tidak akan berubah.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void hitungKuadrat(int x) {
5
6 int main() {
7     int nilai = 5;
8
9     printf("Nilai awal          : %d\n", nilai);
10    hitungKuadrat(nilai);
11    printf("Nilai setelah prosedur : %d\n", nilai);
12
13    return 0;
14 }
15
16 void hitungKuadrat(int x) {
17     x = x * x;
18     printf("Hasil kuadrat      : %d\n", x);
19 }

```

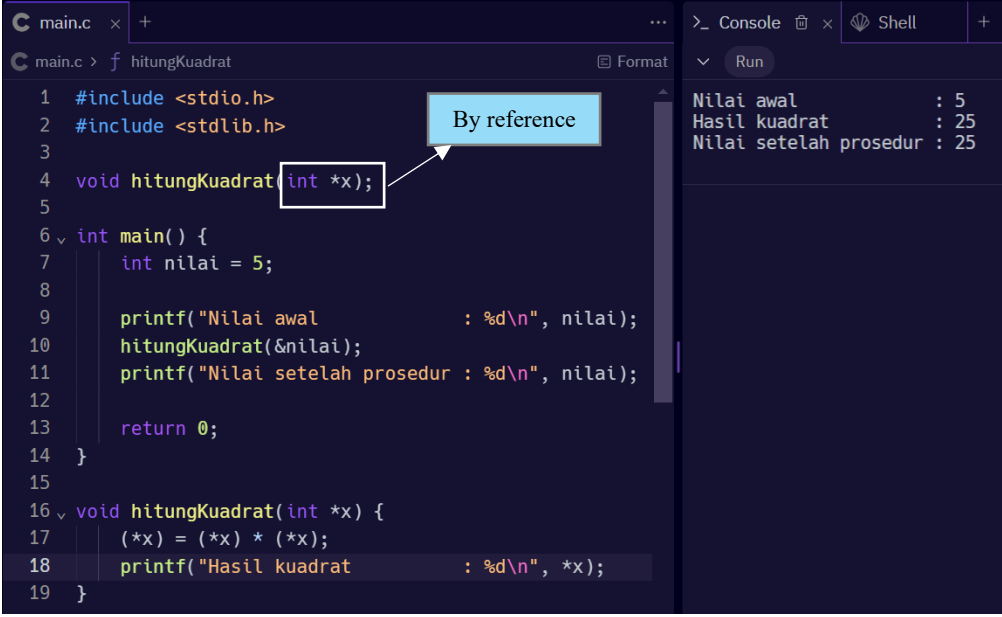
By value

Nilai awal : 5
Hasil kuadrat : 25
Nilai setelah prosedur : 5

Gambar 9. Prosedur dengan *passing parameter by value*

- **Passing parameter by reference**

Pada passing parameter by reference, alamat memori dari variabel akan disalin ke dalam prosedur, bukan nilai dari variabel itu sendiri. Ini artinya, ketika nilai parameter diubah di dalam prosedur, nilai variabel asli di luar prosedur juga akan berubah.



```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void hitungKuadrat(int *x);
5
6 int main() {
7     int nilai = 5;
8
9     printf("Nilai awal          : %d\n", nilai);
10    hitungKuadrat(&nilai);
11    printf("Nilai setelah prosedur : %d\n", nilai);
12
13    return 0;
14 }
15
16 void hitungKuadrat(int *x) {
17     (*x) = (*x) * (*x);
18     printf("Hasil kuadrat      : %d\n", *x);
19 }
  
```

By reference

Nilai awal : 5
Hasil kuadrat : 25
Nilai setelah prosedur : 25

Gambar 10. Prosedur dengan *passing parameter by reference*

D. Jenis Prosedur

Pada dasarnya, prosedur membutuhkan input dan output. Input yang berasal dari input standar (standard input device), yaitu keyboard. Output yang standar berasal dari output standar (standard output device), yaitu monitor. Kedua piranti I/O (Input/Output) dijumpai penggunaannya dalam C oleh library `stdio.h` yang didalamnya terdapat `printf` untuk keperluan input dan `scanf` untuk keperluan output. Berdasarkan asal datangnya input dan output, prosedur dikategorikan ke dalam 4 jenis :

1. Naïve : Prosedur yang tidak memiliki parameter sama sekali dan biasanya digunakan hanya untuk melakukan print.

Contoh:

```
void showMenu() {  
    printf("\n===== MENU =====");  
    printf("\n1. Menu 1");  
    printf("\n2. Menu 2");  
    printf("\n3. Menu 3");  
    printf("\n4. Menu 4");  
    printf("\n5. Keluar");  
    printf("\n=====");  
}
```

Gambar 11. Prosedur *Naïve*

2. Semi Naïve Input : Prosedur yang hanya memiliki parameter input.

Contoh:

```
void volumeKubus(float sisi) {  
    float volume = sisi * sisi * sisi;  
    printf("\nVolume kubus adalah %.2f", volume);  
}
```

Gambar 12. Prosedur *Semi Naïve Input*

3. Semi Naïve Output : Prosedur yang hanya memiliki parameter output

Contoh:

```
void setNilai(int *variabel) {  
    int nilaiBaru;  
  
    printf("Masukkan nilai baru: "); scanf("%d", &nilaiBaru);  
  
    *variabel = nilaiBaru;  
}
```

Gambar 13. Prosedur *Semi Naïve Output*

4. Net Effect : Hanya ada satu jenis prosedur yang **direkomendasikan**, yaitu prosedur yang menghasilkan efek netto (*nett effect procedure*). Prosedur ini menghasilkan efek

netto yang berarti tidak menggunakan standard device untuk input maupun output (tidak terdapat `printf` maupun `scanf` dalam prosedur).

Prosedur ini paling direkomendasikan karena menggabungkan keuntungan dari 2 pendekatan lainnya yaitu semi naïve input dan semi naïve output. Dengan prosedur nett effect, program akan semakin hemat penggunaan memori, mengoptimalkan kecepatan pemrosesan data, terutama ketika data yang diolah sangat kompleks. Contoh :



```
void hitungPangkat(int base, int pangkat, int *hasil);

int main() {
    int base = 2, pangkat = 3, hasil;

    hitungPangkat(base, pangkat, &hasil);

    printf("%d pangkat %d adalah %d\n", base, pangkat, hasil);

    return 0;
}

void hitungPangkat(int base, int pangkat, int *hasil) {
    *hasil = 1;
    for (int i = 0; i < pangkat; i++) {
        (*hasil) = (*hasil) * base;
    }
}
```

Gambar 14. Prosedur *Nett Effect*

E. Pemanggilan Prosedur Oleh Prosedur Lain

Suatu prosedur bukan merupakan program yang berdiri sendiri sehingga tidak dapat dieksekusi secara langsung ketika program dijalankan (berbeda dengan main program). Untuk itu suatu prosedur memerlukan sebuah pemanggil. Prosedur dapat dipanggil darimana saja, dari `main()` function, dari fungsi lain, maupun dari prosedur lain. Pemanggilan dari `main()` function telah dipelajari pada modul sebelumnya. Materi fungsi akan didapatkan pada modul selanjutnya. Kali ini, pembahasan akan berfokus pada pemanggilan prosedur oleh prosedur lain. Perhatikan contoh kasus berikut, diberikan prosedur akumulasi dan kenaikan gaji:

```

void akumulasi(double *hasil, double delta);
void kenaikanGaji(double *gaji, double persenKenaikan);

int main(int argc, char *argv[]) {
    int gaji = 10000;
    int persen_kenaikan = 10;

    printf("Gaji sebelum naik: %d\n", gaji);
    kenaikanGaji(&gaji, persen_kenaikan);
    printf("Gaji sekarang: %d", gaji);

    return 0;
}

void akumulasi(double *hasil, double delta) {
    *hasil = *hasil + delta;
}

void kenaikanGaji(double *gaji, double persenKenaikan) {
    double kenaikan;
    kenaikan = (persenKenaikan / 100) * (*gaji);
    *gaji = (*gaji) + kenaikan;
}

```

Gambar 15. Program untuk menghitung kenaikan gaji

Pada prosedur `akumulasi`, `hasil` sudah ada isi/nilainya, hendak diakumulasikan dengan `delta`. `delta` sudah terdefinisi dan ada isi/nilainya juga. Kemudian, `hasil` akan ketambahan sebesar `delta`. Isi/nilai dari `delta` tetap. Pada prosedur `kenaikanGaji`, `gaji` sudah ada isinya, hendak diakumulasikan dengan `persenKenaikan` dikalikan `gaji`. `persenKenaikan` sudah terdefinisi dan ada isi/nilainya juga. Kemudian, `gaji` akan ketambahan sebesar `persenKenaikan` dikalikan `gaji`. Isi/nilai dari `persenKenaikan` tetap.

Apakah kalian melihat adanya struktur yang sama pada isi prosedur `akumulasi` dan `kenaikanGaji`? Jika kalian melihat dan mencermati, `*gaji = (*gaji) + kenaikan;` pada isi prosedur `kenaikanGaji` memiliki struktur yang sama dengan isi pada prosedur `akumulasi`.

Ini menunjukkan adanya ketidakefektifan karena adanya code yang ditulis berulang - ulang. Lalu bagaimana mengatasinya? Adalah dengan memanggil prosedur `akumulasi` yang sudah dibuat sebelumnya. Sehingga, prosedur berubah menjadi:

```
void akumulasi(double *hasil, double delta) {  
    *hasil = *hasil + delta;  
}  
  
void kenaikanGaji(double *gaji, double persenKenaikan) {  
    double kenaikan;  
    kenaikan = (persenKenaikan / 100) * (*gaji);  
    akumulasi(&(*gaji), kenaikan);  
}
```

Gambar 16. Prosedur kenaikan gaji dan prosedur akumulasi

```
Gaji sebelum naik: 10000  
Gaji sekarang: 10000
```

Gambar 17. Contoh *output* dari program kenaikan gaji

Remainder!!!

Perlu diingat bahwa ketika pemanggilan prosedur didalam prosedur lain, aturan pemanggilan prosedur tetaplah berlaku. Jangan lupa untuk mendeklarasikan setiap variabel yang akan digunakan didalam prosedur yang akan dipanggil nantinya.

Pointer (*)

Jika kalian perhatikan pada potongan code diatas, ketika prosedur `akumulasi` dipanggil didalam prosedur `kenaikanGaji`, syntax nya yaitu `akumulasi(&(*gaji), kenaikan);`.

Q : “Itu kok ada karakter ‘&’ kak? Buat apaan tu?”

Jawaban Sederhana :

Masih ingatkah kalian pada modul Prosedur 1 sebelumnya kita mengubah value yang dimiliki oleh suatu variabel melalui sebuah prosedur dengan menggunakan **pointer**? Dan ketika kita ingin menarik / mengambil value tersebut kita menggunakan karakter ‘&’ pada variabel kita? Ya, konsep tersebut tetap berlaku ketika kalian memanggil suatu prosedur didalam prosedur lainnya.

Dalam prosedur `akumulasi(double *hasil, double delta)`, variabel pertama didalam parameternya (`*hasil`) menggunakan pointer. Ketika kita melakukan pemanggilan prosedur akumulasi, maka syntaxnya menjadi `akumulasi(&hasil, delta)`.

Sekarang perhatikan prosedur `kenaikanGaji(double *gaji, double persenKenaikan)`. Kita menggunakan pointer pada variabel gaji yang ingin diubah value nya dengan cara mengakumulasikan **gaji** dengan **kenaikan gajinya**. Maka kita memanggil prosedur akumulasi dengan cara `akumulasi(&(*gaji), kenaikan);`. Karakter **&** muncul karena kita sebelumnya menggunakan variabel ber pointer pada prosedur akumulasi, tujuannya adalah untuk menarik / mengambil value yang telah di olah didalam prosedur akumulasi tadi. Variabel `*gaji` tetap menggunakan pointer, sebab kita mendeklarasikan `*gaji` didalam prosedur `kenaikanGaji` menggunakan pointer agar valuenya bisa berubah didalam main program nantinya.

Masih bingung dengan konsep pointer? Tidak apa-apa, yang penting kalian ingat saja kalau sebelumnya ada variabel yang **menggunakan pointer** dalam parameternya, maka pada saat pemanggilan **gunakan karakter ‘&’** pada variabel.

Pointer sendiri sebenarnya berfungsi untuk menunjuk alamat pada memory, dan karakter ‘&’ berfungsi untuk memanggil value dari alamat yang ditunjuk oleh pointer. Namun hal ini tidak akan dibahas lebih dalam pada modul ini, karena sebenarnya konsep pointer sendiri akan diulas lebih dalam pada mata kuliah Informasi dan Struktur Data di semester 3 nanti.

F. Beberapa Hal Yang Harus Diperhatikan Pada Prosedur

1. Pemanggilan prosedur harus berdiri sendiri sebagai sebuah statement merdeka, tidak boleh ditempelkan pada statement lain.

```

void hitungLuasSegitiga(float alas, float tinggi, float *luas);

int main() {
    float alas = 5.0, tinggi = 3.0, luas;
    hitungLuasSegitiga(alas, tinggi, &luas);
    printf("Luas segitiga adalah %.2f\n", luas);
    printf("Luas segitiga adalah %.2f\n", hitungLuasSegitiga(alas, tinggi, &luas));
    return 0;
}

void hitungLuasSegitiga(float alas, float tinggi, float *luas) {
    *luas = (alas * tinggi) / 2;
}

```

Benar (berdiri sendiri sebagai statement terpisah)

Salah (tidak berdiri sendiri)

Gambar 18. Pemanggilan prosedur yang benar dan salah pada program

2. Nama prosedur baik saat pemanggilan, pendeklarasian, maupun saat pendefinisian merupakan case sensitive

```

//Deklarasi
void hitungLuasSegitiga(float alas, float tinggi, float *luas);

int main() {
    float alas = 5.0, tinggi = 3.0, luas;

    //Pemanggilan
    hitungLuasSegitiga(alas, tinggi, &luas);

    printf("Luas segitiga adalah %.2f\n", luas);

    return 0;
}

//Definisi Prosedur
void hitungLuasSegitiga(float alas, float tinggi, float *luas) {
    *luas = (alas * tinggi) / 2;
}

```

Benar (case sensitive)

Gambar 19. Pemanggilan nama prosedur yang benar pada program luas segitiga

```

//Deklarasi
void hitungLuasSegitiga(float alas, float tinggi, float *luas);

int main() {
    float alas = 5.0, tinggi = 3.0, luas;

    //Pemanggilan
    hitungluassegitiga(alas, tinggi, &luas);

    printf("Luas segitiga adalah %.2f\n", luas);

    return 0;
}

//Definisi Prosedur
void hitungLuasSegitiga(float alas, float tinggi, float *luas) {
    *luas = (alas * tinggi) / 2;
}

```

Salah (case sensitive)

Gambar 20. Pemanggilan nama prosedur yang salah pada program luas segitiga

3. Nama variabel di parameter actual dan parameter formal **dapat berbeda maupun sama**.
Namun untuk **tipe datanya harus sama**.

```

void hitungLuasSegitiga(float alas, float tinggi, float *luas);

int main() {
    float a, t, luas;

    printf("Input alas : "); scanf("%f", &a);
    printf("Input tinggi : "); scanf("%f", &t);

    hitungLuasSegitiga(a, t, &luas);

    printf("Luas segitiga adalah %.2f\n", luas);

    return 0;
}

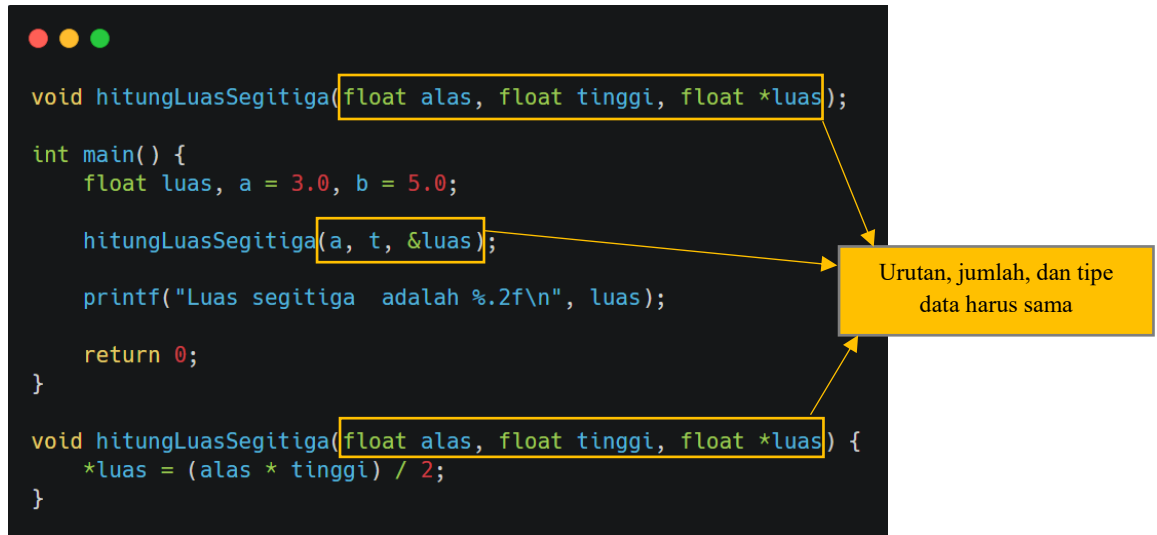
void hitungLuasSegitiga(float alas, float tinggi, float *luas) {
    *luas = (alas * tinggi) / 2;
}

```

a dan t merupakan parameter variabel local, sedangkan alas dan tinggi merupakan variabel local yang ada dalam prosedur

Gambar 21. Pemanggilan prosedur dengan nama variabel yang berbeda

4. Jumlah parameter, tipe data, dan urutan parameter **harus sama** baik di parameter aktual maupun parameter formal.



Gambar 22. Pemanggilan prosedur dengan urutan, jumlah, dan tipe data yang sama

GUIDED

Pada Guided Prosedur 2, kita akan lebih berfokus pada penerapan Nett Effect Procedure dan bagaimana mengimplementasikan pemanggilan prosedur didalam prosedur lain.

RENTAL MOTOR WINDAH BERSAUDARA



Bang Windah adalah seorang pengusaha rental motor yang menjalankan usahanya dengan nama "Rental Motor Windah Bersaudara". Setiap hari, dia mengelola penyewaan motor dengan berbagai merk dan tipe untuk memenuhi kebutuhan transportasi masyarakat di sekitar kota. Namun, karna jumlah pelanggan yang merental motor semakin banyak, bang Windah pun kewalahan dalam manajemen penyewaan motornya dan butuh bantuan agar ia dan stafnya dapat memanajemen sistem perentalan dengan mudah. Oleh karena itu, bang Windah menyewa kalian sebagai programmer handal untuk membuatnya program manajemen rental motor dari sudut pandang tempat penyewaan (bukan penyewa) dengan ketentuan sebagai berikut.

Login page

Wajib menggunakan nett effect

Ketika pertama kali memasuki program, pengguna diminta menginputkan username dan password, dengan ketentuan username “Brando” dan password 5 digit terakhir NPM praktikan. Program akan menampilkan error/peringatan apabila username dan password yang dimasukan salah. Toleransi kesalahan yang diberikan sebanyak 3 kali, jika user salah memasukkan username dan password sampai 3 kali, maka program akan keluar. Tampilkan nama dan NPM saat program keluar dengan format: “[NAMA – NPM – KELAS]”. Apabila user berhasil login, maka program akan masuk ke pilihan menu.

1. Menu Input Data Penyewa

Menu ini hanya bisa diakses satu kali. Pada menu ini, pengguna diminta menginputkan nama penyewa, nomor telepon, alamat, dan jaminan. Adapun ketentuan program : nama, alamat, dan jaminan tidak boleh kosong, sedangkan nomor telepon harus berada di antara 11 sampai 13 digit. Program akan terus meminta inputan ulang selama data yang dimasukkan tidak sesuai dengan ketentuan.

2. Menu Input Pemesanan

Wajib mengimplementasikan prosedur nett effect

Menu ini hanya bisa di akses jika pengguna sudah menginput data penyewa. Selain itu, menu ini juga hanya bisa di akses satu kali untuk satu penyewa. Ada 3 tipe motor yang disewakan oleh bang Windah, yaitu

Merk Motor	Stok	Harga / jam
Ronda	11	2500
Yamahal	9	3500
Sijuki	7	2000

Pada menu ini, pengguna diminta menginputkan jenis motor antara Ronda, Yamahal, atau Sijuki. Program akan meminta ulang inputan selama jenis motor yang di input tidak sesuai. Jika

sisa stok motor sudah habis, maka motor tersebut tidak bisa disewa. Kemudian program juga akan meminta pengguna untuk menginput nomor plat, dengan ketentuan nomor plat tidak boleh kosong.

3. Menu Show Rincian Pesanan

Menu ini hanya bisa di akses jika pengguna telah melakukan pemesanan. Menu ini berfungsi untuk menampilkan rincian pesanan motor yang akan di sewa berupa nama penyewa, alamat, nomor telepon, jaminan, merk motor yang di sewa, plat, dan biaya sewa / jam.

4. Menu Pembayaran

Wajib mengimplementasikan prosedur nett effect

Menu ini berfungsi untuk melakukan pembayaran, dan dapat diakses jika pengguna telah input pemesanan. Pada menu ini akan ditampilkan rincian pesanan seperti pada menu 3. Kemudian program akan meminta inputan berupa durasi peminjaman motor. Durasi yang di input tidak boleh kurang dari 1 jam, program akan meminta ulang inputan durasi jika tidak sesuai ketentuan. Tampilkan total harga dimana total harga diperoleh dari biaya motor per jam di kali durasi. Program kemudian meminta pengguna untuk melakukan pembayaran. Selama nominal uang kurang dari total harga, maka program akan terus meminta inputan dari pengguna. Jika nominal uang yang di input lebih dari sama dengan total harga, maka pembayaran berhasil dilakukan. Tampilkan kembalian jika nominal uang lebih dari total biaya. Ketika pembayaran berhasil dilakukan, stok motor yang tersedia juga harus berkurang dan program sudah boleh mengakses menu 1 untuk menginput data penyewa yang baru.

5. Menu Edit Stok [TUGAS]

Wajib mengimplementasikan prosedur nett effect (tanpa nett effect GD -10)

Menu ini hanya bisa di akses jika transaksi kosong. Artinya tidak ada data penyewa yang terinput, dan transaksi pembayaran sudah selesai dilakukan. Menu ini akan meminta inputan berupa merk motor yang ingin di update stok nya. Error handling cukup mengikuti menu sebelumnya. Kemudian program akan meminta inputan berupa jumlah stok baru, stok yang di input tidak boleh kurang dari 1 unit motor, program akan terus meminta inputan ulang jika inputan kurang dari 1. Kemudian program juga akan meminta inputan harga sewa baru untuk motor tersebut dengan ketentuan harga minimum untuk semua jenis motor adalah Rp 2000. Program akan

meminta ulang inputan jika tidak sesuai ketentuan. Setelah user selesai menginput, stok akan di update.

Contoh Output:

```
===== [ Rental Motor Windah Bersaudara ] =====
[Jumlah Ronda : 11]      [Jumlah Yamahal : 9]      [Jumlah Sijuki : 7]

[1] Input Data Penyewa
[2] Input Pemesanan
[3] Show Rincian Pesanan
[4] Pembayaran
[5] Edit Stok & Harga

[0] Exit
>>> 5

===== [ Edit Stok Motor ] =====

Merk Motor [ Ronda | Yamahal | Sijuki ]: ronda
Update Stok Motor ronda : 0

[!] Jumlah Motor Minimum : 1 Unit [!]

Update Stok Motor ronda : 20
Update Biaya Motor ronda : 1000

[!] Harga Minimum Penyewaan Rp 2000 per Unit [!]

Update Biaya Motor ronda : 5000

[*] Berhasil Mengedit Stok Dan Harga ronda [*]
```

Stok Ronda Berubah dari 11 menjadi 20

```
===== [ Rental Motor Windah Bersaudara ] =====
[Jumlah Ronda : 20]      [Jumlah Yamahal : 9]      [Jumlah Sijuki : 7]

[1] Input Data Penyewa
[2] Input Pemesanan
[3] Show Rincian Pesanan
[4] Pembayaran
[5] Edit Stok & Harga

[0] Exit
>>> |
```

0. Logout

Sebelum logout dari program, program akan meminta konfirmasi pengguna apakah ingin logout atau tidak. Jika terkonfirmasi “Yes” maka pengguna akan kembali ke menu login awal dan seluruh data akan di reset. Jika konfirmasi pengguna bukan “Yes”, maka pengguna batal logout dari program.

Code

1. Deklarasi Prosedur

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

typedef char string[50];

void inisialisasi(int *jam, int *input_perental, int *input_pesanan, int *kesempatan, int *jml_ronda, int *jml_yamahal,
                 int *jml_sijuki, int *sum_ronda, int *sum_yamahal, int *sum_sijuki, float *harga_ronda, float *harga_yamahal,
                 float *harga_sijuki);

void cekLogin(bool *cek, string username, int password);
void mainMenu();
void cekBiaya(float *biaya, string merk, float ronda, float yamahal, float sijuki);
void showNota(string nama, string alamat, string no_telpon, string jaminan, string merk, string plat, float biaya);
void perkalian(int pengali, float *base);
void pengurangan(int pengurang, int *base);
void penjumlahan(int penjumlah, int *base);
void decrementStock(string jenisMotor, int *jml_ronda, int *jml_yamahal, int *jml_sijuki, int pengurang);
void cekSisaStock(int *ketersediaan, string jenisMotor, int jml_ronda, int jml_yamahal, int jml_sijuki);
```

2. Main Program

```

int main(int argc, char *argv[]) {
    //Deklarasi variabel variabel
    bool login_status, exit_status;
    string username, confirm, plat, merk;
    int kesempatan, password, input_perental, menu, input_pesanan, jam, stock;
    float biaya, nominal;

    int jml_ronda, jml_yamahal, jml_sijuki, sum_ronda, sum_yamahal, sum_sijuki;
    float harga_ronda, harga_yamahal, harga_sijuki;
    string nama, alamat, jenis_motor, jaminan, no_telpon;

    //pemanggilan prosedur initalisasi, berfungsi untuk memberi nilai awal pada variabel
    initalisasi(&jam, &input_perental, &input_pesanan, &kesempatan, &jml_ronda, &jml_yamahal, &jml_sijuki, &sum_ronda,
        &sum_yamahal, &sum_sijuki, &harga_ronda, &harga_yamahal, &harga_sijuki);

    do{
        system("cls");
        system("color 8f"); //mengubah warna terminal, silahkan eksplor sendiri

        login_status = false;
        printf("\n\t==== [ Login Page ] =====");
        printf("\n\t[ Sisa Kesempatan : %d kali ]\n", kesempatan);
        printf("\nUsername : "); fflush(stdin); gets(username);
        printf("Password : "); scanf("%d", &password);

        cekLogin(&login_status, username, password); //prosedur cek login. Jika berhasil maka value login_status menjadi true

        if(login_status){ //jika login_status == true, maka pengguna bisa masuk ke main menu
            system("color 3f");
            printf("\n\t[*] Berhasil Login [*]\n\n");
            exit_status = false; //kita akan melakukan perulangan selama exit status ini false
            system("pause");

            do{
                system("cls");
                system("color 70");
                printf("\n\t\t==== [ Rental Motor Windah Bersaudara ] =====\n");
                printf("\n\tStok Motor : ");
                printf("\n\t[ Ronda : %d]\t[ Yamahal : %d]\t[ Sijuki : %d]\n", jml_ronda, jml_yamahal, jml_sijuki);
                mainMenu(); //prosedur untuk menampilkan main menu
                printf("\n\t>>> "); scanf("%d", &menu);
                switch(menu){
                    case 1:
                        if(input_perental == 1){
                            printf("\n\t\t!! Anda Sudah Menginput Data Penyewa !!");
                        }else{
                            printf("\n\t==== [ Input Data Penyewa ] =====\n");
                            printf("\tNama      : "); fflush(stdin); gets(nama);
                            while(strlen(nama)==0){
                                printf("\n\t\t!! Nama Tidak Boleh Kosong !!\n");
                                printf("\tNama      : "); fflush(stdin); gets(nama);
                            }

                            printf("\tNo Telpon : "); fflush(stdin); gets(no_telpon);
                            while(strlen(no_telpon)>13 || strlen(no_telpon)<11){
                                printf("\n\t\t!! Nomor Telpon Harus Diantara 11 Sampai 13 Digit !!\n");
                                printf("\tNo Telpon : "); fflush(stdin); gets(no_telpon);
                            }

                            printf("\tAlamat    : "); fflush(stdin); gets(alamat);
                            while(strlen(alamat)==0){
                                printf("\n\t\t!! Alamat Tidak Boleh Kosong !!\n");
                                printf("\tAlamat    : "); fflush(stdin); gets(alamat);
                            }

                            printf("\tJaminan   : "); fflush(stdin); gets(jaminan);
                            while(strlen(jaminan)==0){
                                printf("\n\t\t!! Jaminan Tidak Boleh Kosong !!\n");
                                printf("\tJaminan   : "); fflush(stdin); gets(jaminan);
                            }

                            printf("\n\t\t[*] Berhasil Input Data Perental [*]\n");
                            input_perental = 1;
                        }
                    }
                }
            } while(exit_status == false);
            break;
        }
    } while(login_status == false);
}

```

```

case 2:
    if(input_perental == 0){
        printf("\n\t\t[!] Data Penyewa Masih Kosong [!]\n");
    }else if(input_pesanan==1){
        printf("\n\t\t[!] %s Sudah Melakukan Pemesanan [!]\n", nama);
    }else{
        printf("\n\t\t==== [ Input Pemesanan ] =====\n");

        printf("\tMerk Motor [ Ronda | Yamaha | Sijuki ]: "; fflush(stdin); gets(merk);
        while(strcmp(merk, "ronda")!=0 && strcmp(merk, "yamaha")!=0 && strcmp(merk, "sijuki")!=0){
            printf("\n\t\t[!] Merk Tidak Tersedia [!]\n");
            printf("\tMerk Motor [ Ronda | Yamaha | Sijuki ]: "; fflush(stdin); gets(merk);
        }
        //pemanggilan prosedur cekSisaStok. prosedur ini akan mengubah value stock sesuai jumlah stok merk tertentu
        cekSisaStock(&stock, merk, jml_ronda, jml_yamaha, jml_sijuki);
        if(stock==0){
            printf("\n\t\t[!] Stock Motor %s Habis, Silahkan Rental Merk Lain [!]\n", merk);
            break;
        }

        printf("\tNomor Plat   : "); fflush(stdin); gets(plat);
        while(strlen(plat)==0){
            printf("\n\t\t[!] Nomor Plat Tidak Boleh Kosong [!]\n");
            printf("\tNomor Plat   : "); fflush(stdin); gets(plat);
        }

        //pemanggilan prosedur cekBiaya. prosedur ini akan mengambil harga sewa motor berdasarkan merk nya,
        //dan disimpan di variabel biaya
        cekBiaya(&biaya, merk, harga_ronda, harga_yamaha, harga_sijuki);
        printf("\n\t\t[*] Berhasil Merental Motor [*]\n");
        input_pesanan = 1;
    }
    break;

case 3:
    if(input_pesanan==1){
        printf("\n\t\t==== [ Rincian Pesanan ] =====\n");
        //pemanggilan prosedur showNota
        showNota(nama, alamat, no_telpon, jaminan, merk, plat, biaya);
        printf("\n\t\t===== \n");
    }else{
        printf("\n\t\t[!] belum Ada Pesanan [!]\n");
    }
    break;

case 4:
    if(input_pesanan==1){
        printf("\n\t\t==== [ Pembayaran ] =====\n");
        //pemanggilan prosedur showNota
        showNota(nama, alamat, no_telpon, jaminan, merk, plat, biaya);

        printf("\n\t\t>> Durasi Peminjaman Motor [Jam]: "); scanf("%d", &jam);
        while(jam<1){
            printf("\n\t\t[!] Minimum Durasi Adalah 1 Jam [!]\n");
            printf("\n\t\tDurasi Peminjaman Motor [Jam]: "); scanf("%d", &jam);
        }
        //prosedur perkalian, untuk mengubah value biaya menjadi biaya dikali jam
        perkalian(jam, &biaya);
        printf("\n\t\tTotal Harga = %.2f \n", biaya);
        printf("\t\tMasukkan Nominal Uang : "); scanf("%f", &nominal);
        while(nominal < biaya){
            printf("\n\t\t[!] Nominal Uang Kurang [!]\n");
            printf("\t\tMasukkan Nominal Uang : "); scanf("%f", &nominal);
        }
        printf("\n\t\t[*] Berhasil Melakukan Pembayaran [!]\n");
        if(nominal > biaya){
            nominal = nominal - biaya;
            printf("\n\t\tKembalian = %.2f ", nominal);
        }

        //prosedur decrementStock berfungsi untuk mengurangi stok motor jika penyewa telah melakukan pembayaran
        decrementStock(merk, &jml_ronda, &jml_yamaha, &jml_sijuki, 1);
        input_pesanan = 0;
        input_perental = 0;
    }else{
        printf("\n\t\t[!] belum Ada Pesanan [!]\n");
    }
    break;

```



```

        case 5:
            if(input_perental==0){
                printf("\n\t\t===== [ Edit Stok Motor ] =====\n\n");
                //Isi jawaban pada bagian ini

            }else{
                printf("\n\t[!] Harap Selesaikan Transaksi Terlebih Dahulu [!]\n");
            }

            break;

        case 0:
            printf("\n\t[!] Anda akan logout dari sistem. Konfirmasi Logout [Yes / No] : ");
            printf("\n\t> "); fflush(stdin); gets(confirm);
            if(strncmp(confirm, "yes")==0){
                printf("\n\t\t[*] Berhasil Logout [*]\n");

                //program melakukan inisialisasi lagi jika pengguna logout, dengan begitu data adak ter reset
                inisialisasi(&jam, &input_perental, &input_pesanan, &kesempatan, &jml_ronda, &jml_yamaha, &jml_sjuki,
                    &sum_ronda, &sum_yamaha, &sum_sjuki, &harga_ronda, &harga_yamaha, &harga_sjuki);
                exit_status = true;
            }else{
                printf("\n\t\t[!] Batal Logout [!]\n");
            }

            break;

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

    }while(!exit_status);

}

}else{
    system("color 4F");
    printf("\n\t[!] Username / password salah [!]\n");
    kesempatan--;
}
getch();

}while(kesempatan!=0);

printf("\n\n[!] Username / password salah sebanyak 3 kali [!]\n");
printf("\n[ Nama Praktikan - NPM - Kelas ]\n\n");
system("pause");
return 0;
}

```

3. Definisi Prosedur

```

/*Pada bagian inialisasi, kita memberikan nilai awal kepada variabel variabel, khususnya variabel yang menampung jumlah stok,
harga, dan lain sebagainya. Mengapa kita menggunakan prosedur dan tidak langsung menginisialisasi di main program saja? Karna kita akan
menggunakan prosedur ini lagi untuk mereset data. Dengan demikian, kita tidak perlu menuliskan code inialisasi berulang kali*/
void inialisasi(int *jam, int *input_perental, int *input_pesanan, int *kesempatan, int *jml_ronda, int *jml_yamahal, int *jml_sijuki,
               int *sum_ronda, int *sum_yamahal, int *sum_sijuki, float *harga_ronda, float *harga_yamahal, float *harga_sijuki){
    *jam = 0;
    *input_perental = 0;
    *input_pesanan = 0;
    *kesempatan = 3;
    *jml_ronda = 11;
    *jml_yamahal = 9;
    *jml_sijuki = 7;
    *sum_ronda=0;
    *sum_yamahal = 0;
    *sum_sijuki = 0;
    *harga_ronda = 2500;
    *harga_yamahal = 3500;
    *harga_sijuki = 2000;
}

/*Prosedur cekLogin akan melakukan perbandingan antara username & password inputan pengguna dengan username & password sistem. Jika
cocok, maka variabel *cek akan diubah nilainya menjadi true*/

void cekLogin(bool *cek, string username, int password){
    if(strcmp(username, "Brando")==0 && password==11111){
        *cek=true;
    }
}

//Untuk menampilkan main menu
void mainMenu(){
    printf("\n\t[1] Input Data Penyewa");
    printf("\n\t[2] Input Pemesanan");
    printf("\n\t[3] Show Rincian Pesanan");
    printf("\n\t[4] Pembayaran");
    printf("\n\t[5] Edit Stok & Harga");
    printf("\n\t[0] Exit");
}

/*Prosedur cekBiaya akan melakukan pengecekan biaya terhadap motor tertentu sesuai merk yang di input pengguna. Hasilnya adalah variabel
*biaya akan menyimpan value berupa harga motor yang sesuai merk inputan pengguna*/
void cekBiaya(float *biaya, string merk, float ronda, float yamahal, float sijuki){
    if(strcmp(merk, "ronda")==0){
        *biaya = ronda;
    }else if(strcmp(merk, "yamahal")==0){
        *biaya = yamahal;
    }else{
        *biaya = sijuki;
    }
}

//untuk menampilkan data penyewa dan detail pesanan
void showNota(string nama, string alamat, string no_telpon, string jaminan, string merk, string plat, float biaya){
    printf("\n\t----- Perental -----");
    printf("\n\tNama      : %s", nama);
    printf("\n\tAlamat     : %s ", alamat);
    printf("\n\tNo Telpon  : %s", no_telpon);
    printf("\n\tJaminan    : %s", jaminan);

    printf("\n\t\t\t\t\t Data Motor -----");
    printf("\n\tMerk       : %s", merk);
    printf("\n\tNo Plat    : %s", plat);
    printf("\n\tBiaya      : %.2f per jam", biaya);
    printf("\n");
}

```

```

void perkalian(int pengali, float *base){
    (*base) = (*base) * pengali;
}

void pengurangan(int pengurang, int *base){
    (*base) = (*base) - pengurang;
}

void penjumlahan(int penjumlah, int *base){
    (*base) = (*base) + penjumlah ;
}

/*prosedur yang akan mengurangi jumlah stok motor sesuai jenis yang di input pengguna. dalam hal ini kita menerapkan pemanggilan prosedur
dalam prosedur lain*/
void decrementStock(string jenisMotor, int *jml_ronda, int *jml_yamaha, int *jml_sijuki, int pengurang){
    if(strcmp(jenisMotor, "ronda")==0){
        pengurangan(pengurang, &(*jml_ronda));
    }else if(strcmp(jenisMotor, "yamaha")==0){
        pengurangan(pengurang, &(*jml_yamaha));
    }else{
        pengurangan(pengurang, &(*jml_sijuki));
    }
}

/*Prosedur yang berguna untuk mengecek sisa stok pada jenis motor tertentu sesuai merk yang di input pengguna*/
void cekSisaStock(int *ketersediaan, string jenisMotor, int jml_ronda, int jml_yamaha, int jml_sijuki){
    if(strcmp(jenisMotor, "ronda")==0){
        (*ketersediaan) = jml_ronda;
    }else if(strcmp(jenisMotor, "yamaha")==0){
        (*ketersediaan) = jml_yamaha;
    }else{
        (*ketersediaan) = jml_sijuki;
    }
}

```

KETENTUAN DAN FORMAT GUIDED

1. **Wajib** mempelajari modul dan guided! Soal Unguided tidak akan jauh jauh dari guided kali ini. Pelajari baik baik logic program. Jika ada pertanyaan yang jawabannya sudah jelas ada di modul maka tidak akan dijawab oleh asdos.
2. Comment **tidak perlu** ditulis di guided.
3. Wajib menggunakan new project dengan extensi **.c**
4. File di zip dengan format penamaan **GD8_X_YYYYY.zip** (X = Kelas; Y = 5 digit terakhir NPM)
5. Kesalahan format penamaan file **-10**
6. Masih bingung dengan materi prosedur? Ada pertanyaan atau mau tau lebih jauh terkait prosedur? Langsung kontak aja ke Teams : Dina Oktavia Pangaribuan (220711928@studets.uajy.ac.id) atau WA ke 082248317801

***Hint**

Apakah kamu adalah seorang bonus hunter? Kabar baik buat kamu karna sudah scroll sampai bawah. Pada soal bonus kali ini kita akan banyak bermain dengan rumus random. Pelajari lagi rumus random, khususnya bagaimana cara menentukan **range** angkanya, baik itu bilangan **bulat** maupun **decimal**. Karna range adalah kunci :)

***Hint part 2**

Apakah kalian memperhatikan pada prosedur **inisialisasi** terdapat variabel yang **tidak terpakai** sama sekali didalam program Guided? Variabel tersebut akan menjadi juru kunci UGD nanti :D

Dalam UGD nanti kalian akan seru-seruan, kalian bisa saja menjadi pemilik toko es krim, warmino, bioskop, bahkan seorang trader dan investor. Oleh karena itu, pelajari guided sebaik mungkin. Selamat belajar!