# 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 adalah kumpulan instruksi atau langkah-langkah yang dijalankan dalam sebuah program komputer untuk menyelesaikan tugas tertentu secara spesifik. Dalam pembuatan prosedur, terdapat beberapa tahapan yang perlu dilakukan, yaitu :

1. **Analisis Masalah**

   Kita harus bisa mengidentifikasi permasalahan yang perlu diselesaikan. Semisal diberikan tugas untuk menghitung penjumlahan dari 3 pasang variabel berbeda.

2. **Desain Solusi**

   Setelah menganalisis masalah, kita perlu merancang alur instruksi yang akan digunakan untuk menyelesaikan permasalahan. Untuk menangani persoalan tersebut, daripada menulis kode penjumlahan setiap pasang variable sebanyak 3 kali, lebih baik membuat sebuah prosedur untuk menghitung hasil penjumlahan yang bisa dipanggil berulang kali.

3. **Implementasi**

Tahap terakhir adalah mengimplementasikan solusi ke dalam Bahasa pemrograman yang akan digunakan.

```c
void tampilkanPenjumlahan(int x, int y);

int main() {
    int a1 = 2, b1 = 3;
    int a2 = 5, b2 = 7;
    int a3 = 10, b3 = 15;

    // Memanggil prosedur 3 kali
    tampilkanPenjumlahan(a1, b1);
    tampilkanPenjumlahan(a2, b2);
    tampilkanPenjumlahan(a3, b3);

    return 0;
}

void tampilkanPenjumlahan(int x, int y) {
    int hasil = x + y;
    printf("Hasil dari %d + %d = %d\n", x, y, hasil);
}
```

```
Hasil dari 2 + 3 = 5
Hasil dari 5 + 7 = 12
Hasil dari 10 + 15 = 25
```

Dalam contoh kasus ini, mungkin penggunaan prosedur belum sepenuhnya terasa manfaatnya karena kode yang dibuat masih sederhana. Namun, tujuan utama membuat prosedur adalah untuk mengatur kode agar lebih rapi, modular, dan mudah dikelola. Selain itu, prosedur sangat berguna untuk menjalankan bagian program yang membutuhkan pengulangan.

## C. Parameter

1. **Parameter Input**

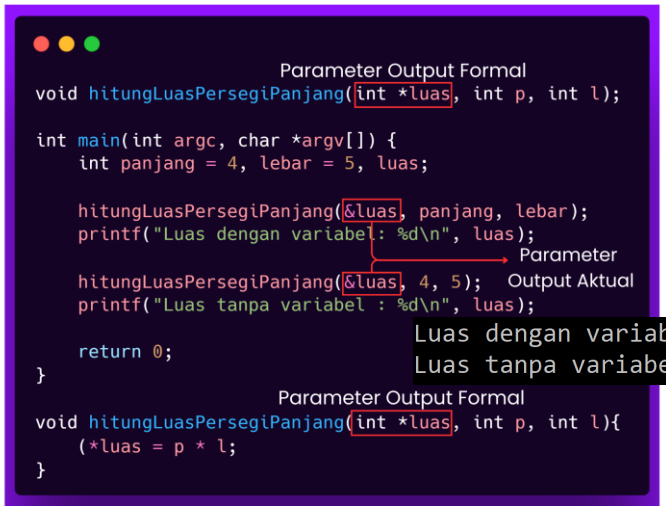
Parameter input adalah parameter yang nilainya sudah pasti atau sudah ditentukan pada saat prosedur dipanggil. Nilai dari parameter ini digunakan sebagai input untuk proses yang ada di dalam prosedur. Sepanjang eksekusi prosedur, nilai parameter input ini tidak boleh diubah selama prosedur berjalan sampai prosedur selesai. Perhatikan contoh berikut :

```c
                                        Parameter Input Formal
void hitungLuasPersegiPanjang(int *luas, int p, int l);

int main(int argc, char *argv[]) {
    int panjang = 4, lebar = 5, luas;

    hitungLuasPersegiPanjang(&luas, panjang, lebar);
    printf("Luas dengan variabel: %d\n", luas);   Parameter
                                                  Input Aktual
    hitungLuasPersegiPanjang(&luas, 4, 5);
    printf("Luas tanpa variabel : %d\n", luas);

    return 0;
}
                                        Parameter Input Formal
void hitungLuasPersegiPanjang(int *luas, int p, int l){
    (*luas = p * l;
}
```

```
Luas dengan variabel: 20
Luas tanpa variabel : 20
```

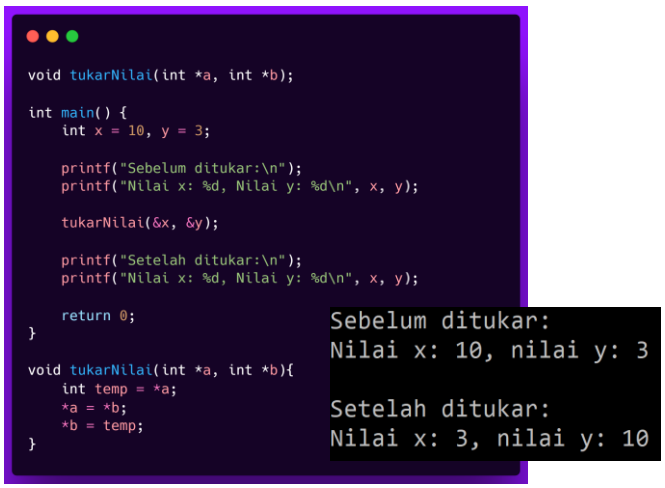
2. **Parameter Output**

Parameter output adalah parameter yang berfungsi sebagai target untuk mengembalikan nilai, atau sebagai penampung hasil keluaran dari prosedur. Parameter ini akan mengalami perubahan selama prosedur berlangsung, karena memang menjadi tujuan dari proses perubahan. Parameter output bisa saja sudah memiliki nilai tertentu sebelum prosedur dipanggil, atau bisa juga belum, namun yang pasti nilainya akan dimodifikasi di dalam prosedur. Ciri khas parameter output formal adalah adanya penanda *pointer* (*) yang melekat pada nama variabelnya, lalu untuk parameter output actual diberi karakter '&' pada nama variabelnya. Dalam bahasa C, biasanya parameter output diletakkan di sebelah kiri, sedangkan parameter input di sebelah kanan. Ini mengikuti bentuk alami dari penugasan (*assignment*) di dalam Bahasa C, yaitu: `output = input1 <operator> input2`. Perhatikan contoh berikut :




3. **Parameter Input/Output**

Parameter output juga bisa sekaligus menjadi parameter input. Parameter ini pastilah sudah jelas isinya ketika memasuki prosedur, karena dia juga berperan sebagai parameter input. Untuk lebih memahaminya, perhatikan contoh berikut :



4. ***Passing Parameter by value & Passing Parameter by Reference***

a) *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.

```c
void hasilKuadrat(int x);

int main() {
    int x = 10;

    printf("Nilai awal      : %d\n", x);
    hasilKuadrat(x);
    printf("Setelah prosedur: %d\n", x);

    return 0;
}

void hasilKuadrat(int x){
    x = x * x;
    printf("Hasil kuadrat   : %d\n", x);
}
```

```
Nilai awal      : 10
Hasil kuadrat   : 100
Setelah prosedur: 10
```

b) *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.

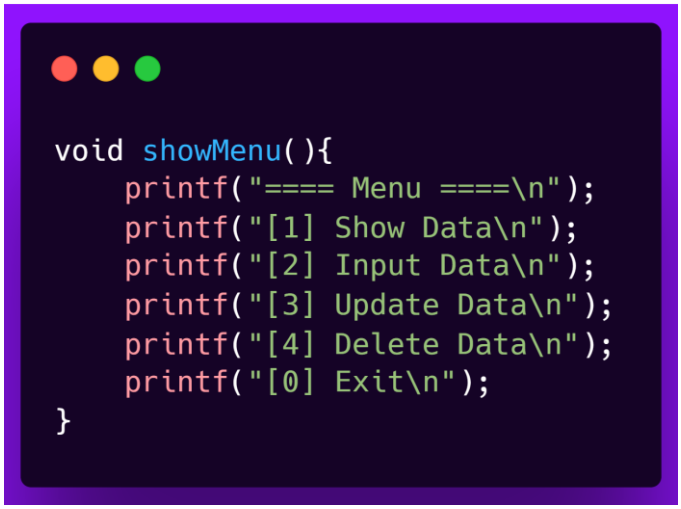
```c
void hasilKuadrat(int *x);

int main() {
    int x = 10;

    printf("Nilai awal      : %d\n", x);
    hasilKuadrat(&x);
    printf("Setelah prosedur: %d\n", x);

    return 0;
}

void hasilKuadrat(int *x){
    (*x) = (*x) * (*x);
    printf("Hasil kuadrat   : %d\n", *x);
}
```

```
Nilai awal      : 10
Hasil Kuadrat   : 100
Setelah prosedur: 100
```

## D. Jenis Prosedur

1. *Naive*

   Prosedur yang tidak memiliki parameter sama sekali dan biasanya digunakan hanya untuk melakukan print. Perhatikan contoh berikut :



```
void showMenu(){
    printf("==== Menu ====\n");
    printf("[1] Show Data\n");
    printf("[2] Input Data\n");
    printf("[3] Update Data\n");
    printf("[4] Delete Data\n");
    printf("[0] Exit\n");
}
```

2. *Semi Naive Input*

   Prosedur yang hanya memiliki parameter input. Perhatikan contoh berikut :

```
void hitungLuasPersegi(int sisi){
    int luas = sisi * sisi;
    printf("Luas Persegi : %d", sisi);
}
```

3. *Semi Naive Output*

   Prosedur yang hanya memiliki parameter output. Perhatikan contoh berikut :

```
void setSisi(int *sisi){
    int temp;

    printf("Masukkan sisi : "); scanf("%d", &temp);

    *sisi = temp;
}
```

4. *Nett Effect*

   Jenis prosedur yang paling direkomendasikan adalah prosedur dengan efek netto (*nett effect proc*edure). Prosedur ini disebut menghasilkan efek

netto karena tidak menggunakan perangkat standar untuk input dan output, sehingga di dalamnya tidak ada perintah seperti `printf` atau `scanf`.

Prosedur ini sangat dianjurkan karena menggabungkan kelebihan dari dua pendekatan lain, yaitu semi naïve input dan semi naïve output. Dengan menggunakan prosedur net effect, program akan menjadi lebih efisien dalam penggunaan memori dan lebih optimal dalam kecepatan pemrosesan data, terutama saat menangani data yang kompleks. Berikut adalah contoh prosedur dengan efek netto :

```c
void hitungLuasPersegiPanjang(int *luas, int p, int l);

int main(int argc, char *argv[]) {
    int panjang = 4, lebar = 5, luas;

    hitungLuasPersegiPanjang(&luas, panjang, lebar);
    printf("Luas dengan variabel: %d\n", luas);

    hitungLuasPersegiPanjang(&luas, 4, 5);
    printf("Luas tanpa variabel : %d\n", luas);

    return 0;
}

void hitungLuasPersegiPanjang(int *luas, int p, int l){
    (*luas = p * l;
}
```

```
Luas dengan variabel: 20
Luas tanpa variabel : 20
```

## E. Pemanggilan Prosedur Dalam Prosedur Lain

Sebuah prosedur bukanlah program yang dapat berjalan sendiri, sehingga tidak bisa dieksekusi langsung saat program dijalankan, berbeda dengan program utama (*main program*). Oleh karena itu, prosedur membutuhkan pemanggil agar dapat digunakan. Prosedur bisa dipanggil dari mana saja, seperti dari fungsi `main()`, dari fungsi lain, atau dari prosedur lain. Pemanggilan prosedur dari `main()` telah dibahas pada modul sebelumnya, sedangkan materi tentang pemanggilan dari fungsi akan dipelajari pada modul berikutnya. Pada bagian ini, fokus pembahasan adalah pemanggilan prosedur oleh prosedur lain. Perhatikan prosedur `luasPersegi` dan `luasKubus` berikut :

```c
void luasPersegi(int *luasP, int sisi);
void luasKubus(int *luasK, int luasP);

int main() {
    int s = 4, luasP, luasK;

    luasPersegi(&luasP, s);
    luasKubus(&luasK, s);

    printf("Luas persegi : %d\n", luasP);
    printf("Luas Kubus   : %d", luasK);

    return 0;
}

void luasPersegi(int *luasP, int sisi){
    *luasP = sisi * sisi;
}

void luasKubus(int *luasK, int sisi){
    int luasP;

    luasP = sisi * sisi;

    (*luasK) = 6 * luasP;
}
```

Apakah kalian melihat ada struktur yang sama pada isi prosedur `luasPersegi` dan `luasKubus`? Jika kalian cermati, `luasP = sisi * sisi` pada prosedur `luasKubus` sama denga nisi prosedur `luasPersegi`. Hal ini menunjukkan bahwa kode diatas kurang efektif karena ada kode yang ditulis berulang-ulang. Untuk mengatasi hal tersebut kita bisa memanggil prosedur `luasPersegi` di dalam prosedur `luasKubus`. Maka kode akan menjadi seperti berikut :

```c
void luasPersegi(int *luasP, int sisi);
void luasKubus(int *luasK, int luasP);

int main() {
    int s = 4, luasK;

    luasKubus(&luasK, s);

    printf("%d", luasK);

    return 0;
}

void luasPersegi(int *luasP, int sisi){
    (*luasP) = sisi * sisi;
}

void luasKubus(int *luasK, int sisi){
    int luasP;

    luasPersegi(&luasP, sisi);

    (*luasK) = 6 * luasP;
}
```

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. Pada contoh diatas kita perlu membuat variable `luasP` agar bisa memanggil prosedur `luasPersegi`, untuk variabel `sisi` kita tidak perlu membuat variable baru lagi karena di parameter prosedur `luasKubus` sudah ada variabel `sisi`.

## F. *Pointer* (*)

Sama seperti modul Prosedur 1 sebelumnya, jika kita ingin mengubah *value* yang dimiliki oleh suatu variabel melalui sebuah prosedur kita harus menggunakan *pointer*. Dan ketika kita ingin menarik atau mengambil *value* tersebut kita menggunakan karakter '&' pada variabel kita. Konsep tersebut tetap berlaku ketika kalian memanggil suatu prosedur didalam prosedur lainnya. Perhatikan kode pengisian tangki air berikut :

```c
void akumulasi(double *volumeTangki, double tambahan);
void isiTangki(double *volumeTangki, double debit, double jam);

int main() {
    double volumeSaatIni = 1000.0;
    double debitAir = 250.0;
    double waktuPengisian = 2.0;

    isiTangki(&volumeSaatIni, debitAir, waktuPengisian);

    printf("Volume tangki setelah pengisian: %.2f liter\n", volumeSaatIni);

    return 0;
}

void akumulasi(double *volumeTangki, double tambahan){
    *volumeTangki = *volumeTangki + tambahan;
}

void isiTangki(double *volumeTangki, double debit, double jam){
    double volumeMasuk;

    volumeMasuk = debit * jam;
    akumulasi(&(*volumeTangki), volumeMasuk);
}
```

Pada prosedur `isiTangki` terdapat kode pemanggilan prosedur `akumulasi` dengan *syntax* `akumulasi(&(*volumeTangki), volumeMasuk)`. Lalu, untuk apa karakter '&' di dalam *syntax* tersebut?

Pada prosedur `akumulasi`, terdapat parameter `volumeTangki` yang menggunakan *pointer*. Maka pada saat pemanggilan prosedur `akumulasi` *syntax*-nya adalah `akumulasi(&volumeTangki, tambahan)`.

Pada prosedur `isiTangki`, kita ingin mengubah nilai dari variabel `volumeTangki` dengan menambahkan volume air tambahan ke volume awal. Karena kita ingin nilai `volumeTangki` benar-benar berubah (bukan hanya salinan sementara), maka kita mendeklarasikannya sebagai *pointer* dalam parameter prosedur `isiTangki`. Ketika memanggil prosedur `akumulasi`, kita menuliskan `akumulasi(&(*volumeTangki), volumeMasuk)`. Di sini, `*volumeTangki` digunakan untuk mengakses nilai yang ditunjuk oleh *pointer*, lalu kita gunakan

‘&’untuk mengambil alamat dari nilai tersebut agar bisa dikirim kembali sebagai *pointer* ke prosedur `akumulasi`.

```c
void akumulasi(int *hasil, int tambahan);

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

    akumulasi(&x, 5);

    return 0;
}

void akumulasi(int *hasil, int tambahan){
    *hasil = *hasil + tambahan;
}
```

*Pointer* sendiri sebenarnya berfungsi untuk mengambil *value* dari alamat yang ditunjuk, dan karakter ‘&’ berfungsi untuk mengambil alamat dari suatu variabel. Perhatikan kode diatas, saat di dalam `main()` ditulis `akumulasi(&x, 5)` artinya alamat variabel `x` akan dikirim ke prosedur `akumulasi` dan *value*-nya akan diambil oleh `*hasil`. Karena objeknya sama, saat *value*-nya dirubah didalam prosedur *value* itu akan tetap disimpan oleh alamatnya yaitu `&x`.

Masih bingung tentang *pointer*? Tenang... 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.

## *GUIDED*

Pada guided ini kita akan lebih berfokus pada penerapan *nett effect procedure* dan mengimplementasikan prosedur di dalam prosedur lain.

## Penyewaan Drone AtmaTech

AtmaTech sedang mengembangkan sistem penyewaan drone berbasis terminal menggunakan bahasa pemrograman C. Sistem ini terdiri dari tiga unit drone, yang masing-masing memiliki status (apakah sedang dipinjam atau tidak) dan kapasitas baterai dalam bentuk persentase. Program ini memiliki sistem *login* untuk menjaga keamanan akses, di mana pengguna harus memasukkan *username* dan *password* yang benar dalam tiga kali kesempatan. Bila pengguna gagal *login* sebanyak tiga kali, program akan keluar secara otomatis. Namun jika berhasil *login*, pengguna akan diarahkan ke menu utama dengan empat pilihan sebagai berikut :

1. **Isi Baterai Drone**

   Pada menu ini, pengguna diminta untuk memilih salah satu dari tiga unit drone. Sistem kemudian akan mengecek apakah unit yang dipilih sedang tersedia (tidak sedang dipinjam). Jika unit tersedia, maka pengguna diminta untuk memasukkan durasi pengisian baterai dalam satuan menit. Program akan mengonversi waktu tersebut ke dalam persentase, berdasarkan asumsi bahwa 60 menit akan mengisi baterai sebesar 100%. Hasil konversi ini akan ditambahkan ke kapasitas baterai drone yang bersangkutan. Namun, jika setelah penambahan baterai menjadi lebih dari 100%, maka sistem akan membatasi nilainya tetap maksimal 100%.

2. **Sewa Drone**

   Jika pengguna memilih menu kedua, sistem akan meminta pengguna untuk memilih unit drone yang akan disewa. Sistem kemudian akan memeriksa dua hal: apakah unit tersedia (tidak sedang dipinjam) dan apakah

baterai pada unit tersebut minimal 30%. Jika kedua kondisi terpenuhi, pengguna akan diminta memasukkan durasi peminjaman dalam satuan menit. Sistem kemudian akan mengubah status unit menjadi “dipinjam” dan menyimpan durasi peminjaman tersebut untuk nantinya digunakan saat pengembalian.

3. **Kembalikan Drone**

Pada menu ini, pengguna kembali diminta memilih unit drone yang ingin dikembalikan. Sistem akan mengecek apakah unit tersebut memang sedang dalam status dipinjam. Jika benar, maka program akan mengubah status unit menjadi “tersedia” kembali dan mengurangi kapasitas baterai berdasarkan durasi peminjaman sebelumnya. Pengurangan baterai dihitung dengan rumus yang sama seperti saat pengisian, yaitu bahwa 60 menit pemakaian akan menghabiskan 100% baterai. Jika setelah dikurangi nilainya menjadi kurang dari 0%, maka baterai akan dianggap 0%.

4. **Keluar**

Jika pengguna memilih menu ini, sistem akan menanyakan konfirmasi logout. Bila pengguna mengetikkan "yes", maka program akan kembali ke halaman login, dan semua data unit drone akan di-reset ke kondisi awal. Jika tidak, maka sistem akan membatalkan proses logout dan kembali ke menu utama.

# Kode

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

typedef char string[50];

void init(int *kesempatan, float *batUnit1, float *batUnit2, float *batUnit3, bool *statusUnit1, bool *statusUnit2, bool *statusUnit3);
void cekLogin(bool *cek, string username, string password);
void printMenu();

void isSewa(bool *isValid, bool statusUnit1, bool statusUnit2, bool statusUnit3, int pilihUnit);
void batasBaterai(float *batUnit);
void tambahBaterai(float *batUnit, float menit);
void ubahMenit(float *persentase, float menit);
void tambahBateraiUnit(float *batUnit1, float *batUnit2, float *batUnit3, float menit, int pilihUnit);

void cekKondisi(bool *isValid, float batUnit, bool statusUnit);
void cekKondisiUnit(bool *isValid, float batUnit1, float batUnit2, float batUnit3, bool statusUnit1, bool statusUnit2,
                    bool statusUnit3, int pilihUnit);
void ubahStatus(bool *statusUnit, float *durasiUnit, float durasi);
void ubahStatusUnit(bool *statusUnit1, bool *statusUnit2, bool *statusUnit3, float *durasiUnit1, float *durasiUnit2, float *durasiUnit3,
                    float durasi, int pilihUnit);

void kurangBaterai(float *batUnit, float menit);
void pengembalian(bool *statusUnit, float *batUnit, float durasiUnit);
void pengembalianUnit(bool *statusUnit1, bool *statusUnit2, bool *statusUnit3, float *batUnit1, float *batUnit2, float *batUnit3,
                      float durasiUnit1, float durasiUnit2, float durasiUnit3, int pilihUnit);
```

```c
int main(int argc, char *argv[]) {
    bool loginStatus, exitStatus;
    string username, password, confirm;
    int kesempatan, menu;

    float batUnit1, batUnit2, batUnit3, durasiUnit1, durasiUnit2, durasiUnit3, pengisian, durasi;
    int pilihUnit;
    bool statusUnit1, statusUnit2, statusUnit3, isValid;

    init(&kesempatan, &batUnit1, &batUnit2, &batUnit3, &statusUnit1, &statusUnit2, &statusUnit3);

    do{
        system("cls");
        system("color 07");

        loginStatus = false;
        printf("\n\t\t===== [ Login Page ] =====");
        printf("\n\t\t[ Sisa Kesempatan : %d kali ]\n", kesempatan);
        printf("\n\tUsername : "); fflush(stdin); gets(username);
        printf("\n\tPassword : "); fflush(stdin); gets(password);

        cekLogin(&loginStatus, username, password);

        if(loginStatus){
            system("color a0");
            printf("\n\t\t[*] Berhasil Login [*]\n\n");
            exitStatus = false;
            system("pause");
            do{
                system("cls");
```

```c
                system("color 07");
                printf("\n\t\t\t\t===== [ Penyewaan Drone ] =====");
                printf("\n\tUnit 1: %.2f%% [%s] | Unit 2: %.2f%% [%s] | Unit 3: %.2f%% [%s]\n",
                       batUnit1, statusUnit1 ? "Tersedia" : "Dipinjam",
                       batUnit2, statusUnit2 ? "Tersedia" : "Dipinjam",
                       batUnit3, statusUnit3 ? "Tersedia" : "Dipinjam");
                printMenu();
                printf("\n\t>>> "); scanf("%d", &menu);

                switch(menu){
                    case 1:
                        printf("\n\tPilih unit [1/2/3] : "); scanf("%d", &pilihUnit);
                        if(pilihUnit < 1 || pilihUnit > 3){
                            printf("\n\t\t[!] Unit %d tidak tersedia [!]", pilihUnit);
                        } else {
                            // Mengecek apakah unit bisa disewa (tidak sedang dipinjam)
                            isSewa(&isValid, statusUnit1, statusUnit2, statusUnit3, pilihUnit);

                            if(isValid){
                                printf("\n\tMasukkan lama pengisian (menit): "); scanf("%f", &pengisian);

                                tambahBateraiUnit(&batUnit1, &batUnit2, &batUnit3, pengisian, pilihUnit);

                                printf("\n\t\t[+] Berhasil mengisi baterai Unit %d selama %.2f menit [+]", pilihUnit, pengisian);
                            } else {
                                printf("\n\t\t[!] Unit %d sedang dipinjamkan [!]",  pilihUnit);
                            }
                        }
                    break;
```

```
                case 2:
                    printf("\n\tPilih unit [1/2/3] : "); scanf("%d", &pilihUnit);
                    if(pilihUnit < 1 || pilihUnit > 3){
                        printf("\n\t\t[!] Unit %d tidak tersedia [!]", pilihUnit);
                    } else {
                        // Mengecek apakah kondisi unit memungkinkan untuk disewa
                        cekKondisiUnit(&isValid, batUnit1, batUnit2, batUnit3, statusUnit1, statusUnit2, statusUnit3, pilihUnit);

                        if(isValid){
                            printf("\n\tMasukkan durasi peminjaman (menit): "); scanf("%f", &durasi);
                            ubahStatusUnit(&statusUnit1, &statusUnit2, &statusUnit3, &durasiUnit1, &durasiUnit2, &durasiUnit3,
                                           durasi, pilihUnit);

                            printf("\n\t\t[o] Unit %d telah disewakan [o]", pilihUnit);
                        } else {
                            printf("\n\t\t[!] Kondisi unit %d tidak siap untuk disewakan [!]", pilihUnit);
                        }
                    }
                break;

                case 3:
                    printf("\n\tPilih unit [1/2/3] : "); scanf("%d", &pilihUnit);
                    if(pilihUnit < 1 || pilihUnit > 3){
                        printf("\n\t\t[!] Unit %d tidak tersedia [!]", pilihUnit);
                    } else {
                        // Mengecek apakah unit sedang disewa
                        isSewa(&isValid, statusUnit1, statusUnit2, statusUnit3, pilihUnit);

                        if(!isValid){
                            // Jika sedang disewa, lakukan proses pengembalian
                            pengembalianUnit(&statusUnit1, &statusUnit2, &statusUnit3, &batUnit1, &batUnit2, &batUnit3, durasiUnit1,
                                         durasiUnit2, durasiUnit3, pilihUnit);
```

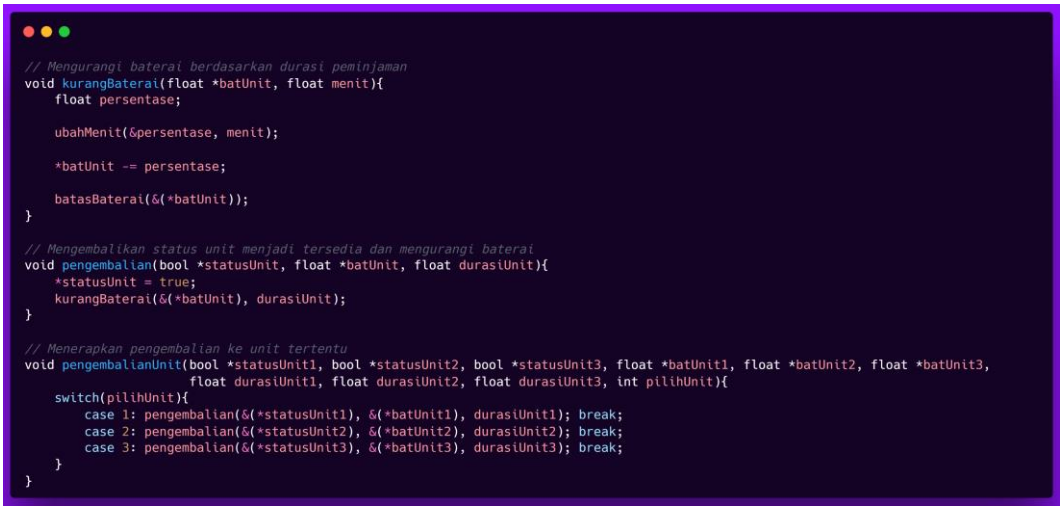
```
                            printf("\n\t\t[+] Berhasil mengembalikan Unit %d [+]", pilihUnit);
                        } else {
                            printf("\n\t\t[!] Unit %d tidak sedang dipinjamkan [!]",  pilihUnit);
                        }
                    }
                break;

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

                        // Reset ke kondisi awal saat logout
                        init(&kesempatan, &batUnit1, &batUnit2, &batUnit3, &statusUnit1, &statusUnit2, &statusUnit3);
                        exitStatus = true;
                    } else {
                        printf("\n\t\t[!] Batal Logout [!]\n");
                    }
                break;
            }

            getch();
        }while(!exitStatus);

    } else {
        system("color 4f");
        printf("\n\t\t[!] Username/Password salah [!]");
```

```
            kesempatan--;
        }
        getch();
    }while(kesempatan != 0);

    printf("\n\n\t[!] Username/Password salah sebanyak 3x [!]");
    printf("\n\t[ Nama | NPM | Kelas ]\n\n");
    system("pause");
    return 0;
}

// Inisialisasi awal baterai dan status unit
void init(int *kesempatan, float *batUnit1, float *batUnit2, float *batUnit3, bool *statusUnit1, bool *statusUnit2,
          bool *statusUnit3){
    *kesempatan = 3;
    *batUnit1 = 20.0;
    *batUnit2 = 80.0;
    *batUnit3 = 75.5;
    *statusUnit1 = *statusUnit2 = *statusUnit3 = true;
}

// Login sederhana: username = nama praktikan, password = 5 digit npm praktikan
void cekLogin(bool *cek, string username, string password){
    if(strcmpi(username, "nama")==0 && strcmpi(password, "npm")==0){
        *cek=true;
    }
}
```

```c
// Menampilkan menu utama
void printMenu(){
    printf("\n\t[1] Isi Baterai Drone");
    printf("\n\t[2] Sewa Drone");
    printf("\n\t[3] Kembalikan Drone");
    printf("\n\t[0] Keluar");
}

// Mengecek apakah unit bisa disewa
void isSewa(bool *isValid, bool statusUnit1, bool statusUnit2, bool statusUnit3, int pilihUnit){
    switch(pilihUnit){
        case 1: *isValid = statusUnit1; break;
        case 2: *isValid = statusUnit2; break;
        case 3: *isValid = statusUnit3; break;
        default: *isValid = false;
    }
}

// Membatasi kapasitas baterai
void batasBaterai(float *batUnit){
    if(*batUnit > 100){
        *batUnit = 100;
    } else if(*batUnit < 0){
        *batUnit = 0;
    }
}
```

```c
// Mengecek apakah kondisi unit layak (tersedia dan baterai >= 30%)
void cekKondisi(bool *isValid, float batUnit, bool statusUnit){
    *isValid = (statusUnit && batUnit >= 30);
}

// Mengecek kondisi kelayakan unit tertentu
void cekKondisiUnit(bool *isValid, float batUnit1, float batUnit2, float batUnit3, bool statusUnit1, bool statusUnit2,
                    bool statusUnit3, int pilihUnit){
    switch(pilihUnit){
        case 1: cekKondisi(&(*isValid), batUnit1, statusUnit1); break;
        case 2: cekKondisi(&(*isValid), batUnit2, statusUnit2); break;
        case 3: cekKondisi(&(*isValid), batUnit3, statusUnit3); break;
        default: *isValid = false;
    }
}

// Mengubah status unit menjadi "dipinjam" dan menyimpan durasi
void ubahStatus(bool *statusUnit, float *durasiUnit, float durasi){
    *statusUnit = false;
    *durasiUnit = durasi;
}

// Menerapkan ubahStatus ke unit tertentu
void ubahStatusUnit(bool *statusUnit1, bool *statusUnit2, bool *statusUnit3, float *durasiUnit1, float *durasiUnit2, float *durasiUnit3,
                    float durasi, int pilihUnit){
    switch(pilihUnit){
        case 1: ubahStatus(&(*statusUnit1), &(*durasiUnit1), durasi); break;
        case 2: ubahStatus(&(*statusUnit2), &(*durasiUnit2), durasi); break;
        case 3: ubahStatus(&(*statusUnit3), &(*durasiUnit3), durasi); break;
    }
}
```

```c
// Konversi durasi ke persentase
void ubahMenit(float *persentase, float menit){
    *persentase = (menit / 60) * 100;
}

// Menambahkan baterai berdasarkan durasi pengisian
void tambahBaterai(float *batUnit, float menit){
    float persentase;

    ubahMenit(&persentase, menit); // Mengkonversi durasi ke persentase

    *batUnit += persentase;

    batasBaterai(&(*batUnit));
}

// Menambahkan baterai ke unit yang dipilih
void tambahBateraiUnit(float *batUnit1, float *batUnit2, float *batUnit3, float menit, int pilihUnit){
    switch(pilihUnit){
        case 1: tambahBaterai(&(*batUnit1), menit); break;
        case 2: tambahBaterai(&(*batUnit2), menit); break;
        case 3: tambahBaterai(&(*batUnit3), menit); break;
    }
}
```



```c
// Mengurangi baterai berdasarkan durasi peminjaman
void kurangBaterai(float *batUnit, float menit){
    float persentase;

    ubahMenit(&persentase, menit);

    *batUnit -= persentase;

    batasBaterai(&(*batUnit));
}

// Mengembalikan status unit menjadi tersedia dan mengurangi baterai
void pengembalian(bool *statusUnit, float *batUnit, float durasiUnit){
    *statusUnit = true;
    kurangBaterai(&(*batUnit), durasiUnit);
}

// Menerapkan pengembalian ke unit tertentu
void pengembalianUnit(bool *statusUnit1, bool *statusUnit2, bool *statusUnit3, float *batUnit1, float *batUnit2, float *batUnit3,
                      float durasiUnit1, float durasiUnit2, float durasiUnit3, int pilihUnit){
    switch(pilihUnit){
        case 1: pengembalian(&(*statusUnit1), &(*batUnit1), durasiUnit1); break;
        case 2: pengembalian(&(*statusUnit2), &(*batUnit2), durasiUnit2); break;
        case 3: pengembalian(&(*statusUnit3), &(*batUnit3), durasiUnit3); break;
    }
}
```

## Ketentuan & Format Pengumpulan Guided

1. Wajib mempelajari modul dan *guided*!
2. *Comment* tidak perlu ditulis di *guided*.
3. Wajib menggunakan *new project* dengan ekstensi **.c** bukan **.cpp**
4. File di zip dengan format penamaan **GD8_X_YYYYY.zip** (X = Kelas; Y = 5 digit terakhir NPM).
5. Kesalahan format penamaan *file* akan dilakukan pengurangan nilai UGD **(-10)**
6. Tindak Plagiasi tidak akan diberikan toleransi **(DIANGGAP TIDAK MENGERJAKAN *GUIDED*)**
7. Kalau kamu masih bingung soal materi prosedur atau ingin tahu lebih dalam lagi, jangan ragu buat hubungi aku langsung.

   Teams : **Muhammad Tafarrel Dhafa Athaya** (230712595@students.uajy.ac.id)

   Atau

   WhatsApp : 081256456103