LAPORAN PRAKTIKUM

POSTTEST 2

ALGORITMA PEMROGRAMAN DASAR

Disusun oleh:

Yoga Ananda Prasetya (2509106017)

Kelas (A1‘25)

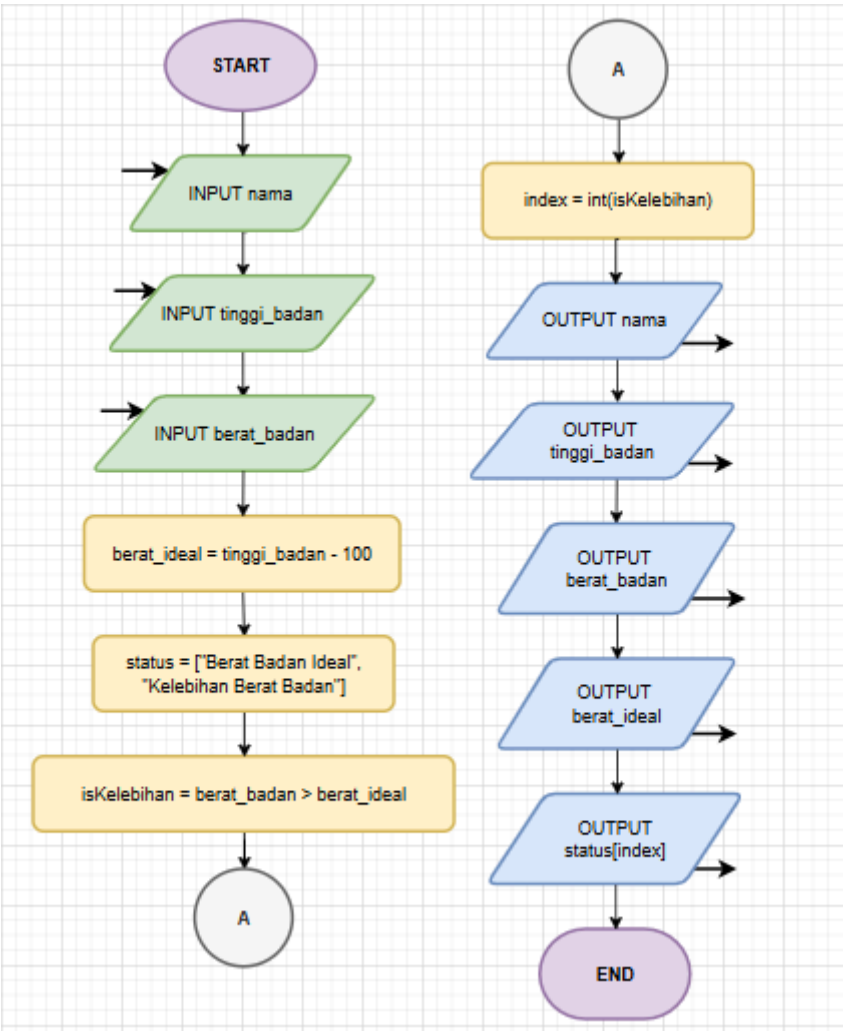
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

SAMARINDA

2025

1. Flowchart



Gambar 1 Flowchart

Flowchart dibagi menjadi tiga langkah atau fase. Langkah pertama adalah bagian *input* yang dilakukan oleh pengguna. Input disini terdiri dari tiga bagian, yaitu input nama, tinggi badan (dinyatakan dalam cm) dan berat badan (dinyatakan dalam kg).

Langkah selanjutnya adalah bagian menghitung berat badan ideal dari pasien, kemudian melakukan perbandingan antara berat badan pasien dengan berat idealnya. Dikarenakan terdapat batasan tidak boleh menggunakan percabangan (*IF* dan *ELSE*) maka bisa menggunakan list, hal ini dikarenakan hanya terdapat dua pilihan / kemungkinan saja.

Untuk prinsip kerjanya, dimulai dari bagian pembentukan *list* terlebih dahulu. *List* disini akan diberi nama “status”, fungsinya untuk menyimpan dua kondisi yang ingin kita gunakan nantinya, yaitu “berat badan ideal” dan “kelebihan berat badan”.

Proses kemudian dilanjutkan dengan perbandingan antara berat badan pasien dengan berat idealnya. Di dalam flowchart dapat digambarkan sebagai berikut :

isKelebihan = berat_badan > berat_ideal

Proses ini bersifat *boolean*, artinya dapat bernilai benar atau salah tergantung dari kedua variabel diatas (berat badan dan berat ideal). Berikut ini merupakan tabel yang merepresentasikan kedua kemungkinan diatas :

Berat Badan	**Berat Ideal**	**Berat Badan > Berat Ideal**	**Penjelasan**
80	90	False	Berat badan kurang dari berat ideal, maka pernyataan menjadi salah
90	90	True	Berat badan lebih dari berat ideal, maka pernyataan menjadi benar

Dari sini diperoleh dua buah kondisi, yaitu *True* dan *False*. Dua kondisi ini juga dapat digambarkan dalam bentuk angka atau *integer*, yaitu 0 (untuk *False*) dan 1 (untuk *True*). Dikarenakan *index* dari *list* dimulai dari angka 0, kita hanya perlu mengkonversi nilai *True* dan *False* menjadi bentuk angka agar dapat digunakan sebagai *index* pada saat memanggil *list*. Oleh karena itu langkah adalah mengkonversi kondisi *True* dan *False*. Proses ini di dalam flowchart digambarkan dengan :

index = int(isKelebihan)

Fase terakhir dari flowchart adalah bagian Output. Bagian ini berfungsi untuk menampilkan semua data yang sudah dimasukkan oleh pasien menjadi satu, mulai dari nama hingga status kesehatannya.

Untuk menampilkan status kesehatannya, kita memanggil salah satu elemen yang terdapat pada list yang telah dibuat sebelumnya. Dalam flowchart itu sendiri, digambarkan oleh *OUTPUT status[index]*. Disini *index* menyimpan nilai perbandingan yang telah dikerjakan sebelumnya, yaitu 0 dan 1. Index ini kemudian digunakan untuk memanggil elemen yang ada di list, 0 untuk elemen “Berat badan ideal” dan 1 untuk elemen “Kelebihan berat badan”.

2. Deskripsi Singkat Program

Tujuan Program	:	1. Membantu pengguna (pasien) untuk memeriksa status berat badannya berdasarkan tinggi badan dan berat badan 2. Membantu pasien untuk memeriksa apakah tergolong kelebihan berat badan atau tidak.
Fitur Program	:	1. Input nama, berat badan dan tinggi pasien 2. Menghitung berat badan ideal serta membandingkannya dengan berat badan pasien 3. Menampilkan hasil pemeriksaan pasien dalam bentuk tabel yang rapi

3. Source Code

A. Fitur Input Data Pasien

Fitur ini digunakan untuk memasukkan nama, tinggi badan, dan berat badan pasien.

```
# Bagian Input nama, tinggi dan berat badan
nama = str(input(f"{'Nama pasien' :32}: "))
tinggi_badan = float(input(f"{'Tinggi badan pasien (dalam cm)' :32}: "))
berat_badan = float(input(f"{'Berat badan pasien (dalam kg)' :32}: "))
```

B. Fitur Menampilkan Tabel Hasil Pemeriksaan

Fitur ini bertujuan untuk menampilkan data hasil pemeriksaan dalam bentuk tabel.

```
# Bagian isi dari tabel akhir
print(f"{'|'} {'Nama Pasien' :30}: {nama} {' ' * (30 - len(nama))} {'|'}")

print(f"{'|'} {'Tinggi Badan' :30}: {tinggi_badan:.2f} {' ' * (29 -
len(str(tinggi_badan)))} {'|'}")

print(f"{'|'} {"Berat Badan" :30}: {berat_badan:.2f} {' ' * (29 -
len(str(berat_badan)))} {'|'}")

print(f"{'|'} {'Berat Ideal' :30}: {berat_ideal:.2f} {' ' * (29 -
len(str(berat_ideal)))} {'|'}")

print(f"{'|'} {'Status' :30}: {status[int(isKelebihan)]} {' ' * (30 -
len(str(status[int(isKelebihan)])))} {'|'}")
```

4. Hasil Output

```
------------------------------------------------------------------
|              PROGRAM MEMERIKSA STATUS BERAT BADAN              |
------------------------------------------------------------------
Nama pasien                     : Yoga
Tinggi badan pasien (dalam cm)  : 155
Berat badan pasien (dalam kg)   : 54
------------------------------------------------------------------
|                    HASIL CEK BERAT BADAN                       |
------------------------------------------------------------------
| Nama Pasien                   : Yoga                           |
| Tinggi Badan                  : 155.00                         |
| Berat Badan                   : 54.00                          |
| Berat Ideal                   : 55.00                          |
| Status                        : Berat Badan Ideal              |
------------------------------------------------------------------
```

Gambar 4.1 Output jika Berat Badan Ideal

```
------------------------------------------------------------------
|              PROGRAM MEMERIKSA STATUS BERAT BADAN              |
------------------------------------------------------------------
Nama pasien                     : Yoga Ananda Prasetya
Tinggi badan pasien (dalam cm)  : 160
Berat badan pasien (dalam kg)   : 65
------------------------------------------------------------------
|                    HASIL CEK BERAT BADAN                       |
------------------------------------------------------------------
| Nama Pasien                   : Yoga Ananda Prasetya           |
| Tinggi Badan                  : 160.00                         |
| Berat Badan                   : 65.00                          |
| Berat Ideal                   : 60.00                          |
| Status                        : Kelebihan Berat Badan          |
------------------------------------------------------------------
```

Gambar 4.2 Output jika Kelebihan Berat Badan

5. Langkah-langkah GIT

5.1 GIT Init

```
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git init
Initialized empty Git repository in E:/Git/praktikum-apd/post-test/post-test-apd-2/.git/
```

Gambar 5.1 Git Init

"Git init" berfungsi untuk membentuk sebuah *repository* di komputer kita. Disini *repository*-nya bersifat lokal, artinya hanya ada di komputer pembuat *repository* tersebut.

5.2 GIT Add

```
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git add 2509106017-yoga-ananda-prasetya-PT-2.py
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2509106017-yoga-ananda-prasetya-PT-2.py
```

Gambar 5.2 Git add

"Git add" disini berfungsi untuk memasukkan file yang sudah kita kerjakan ke dalam *staging area*. *Staging area* disini berfungsi sebagai tempat singgah sementara file-file yang ingin kita *commit* di git nantinya. "Git status" disini hanya berfungsi untuk memeriksa apakah file yang ingin di-*commit* sudah ada di *staging area* atau belum.

5.3 GIT Commit

```
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git commit -m "Update Source Code Python"
[main fe2ab84] Update Source Code Python
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Gambar 5.3 Git Commit

"Git Commit" disini berfungsi untuk menyimpan riwayat dari kode yang telah kita kerjakan ke dalam *repository* yang ada di dalam komputer kita (*repository* lokal). Disini kita juga dapat meninggalkan pesan singkat untuk mengingatkan kita atau orang lain terkait apa yang sudah kita kerjakan di kode tersebut.

5.4 GIT Remote

```
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git remote add origin https://github.com/Yoogaga/belajargit.git
error: remote origin already exists.
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git remote -v
origin  https://github.com/Yoogaga/belajargit.git (fetch)
origin  https://github.com/Yoogaga/belajargit.git (push)
```

Gambar 5.4 Git Remote

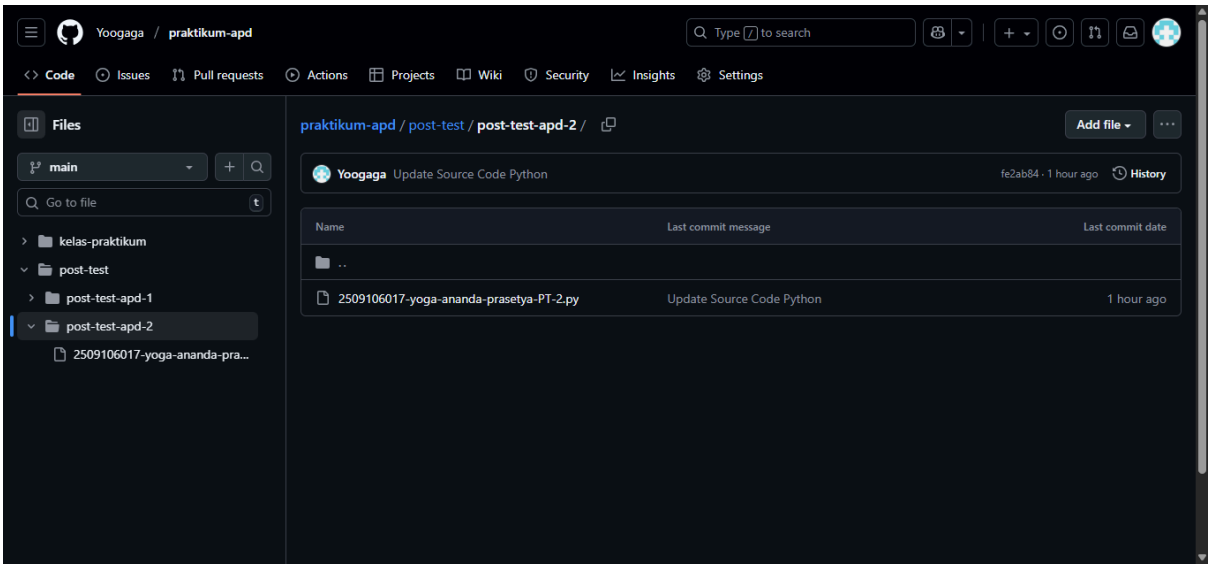
"Git Remote" disini berfungsi untuk menghubungkan *repository* lokal yang ada di komputer kita dengan *repository online* yang ada di Github. Disini karena saya sudah menghubungkan kedua *repository* saya, maka pesan yang muncul adalah error. Disini saya menggunakan "Git remote -v" untuk menampilkan sekaligus memeriksa hubungan antara kedua *repository* saya.

5.5 GIT Push

```
PS E:\Git\praktikum-apd\post-test\post-test-apd-2> git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 484 bytes | 53.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: This repository moved. Please use the new location:
remote:   https://github.com/Yoogaga/praktikum-apd.git
To https://github.com/Yoogaga/belajargit.git
   3cc3e16..fe2ab84  main -> main
branch 'main' set up to track 'origin/main'.
```

Gambar 5.5 Git Push

"Git Push" disini berfungsi untuk mengunggah file-file yang sudah di-*commit* di *repository* lokal menuju ke *repository* yang ada di Github. Tujuannya adalah agar file-file yang sudah dikerjakan dapat diakses secara *online* dan dapat diakses oleh siapapun selama pengaturan *repository*-nya tidak di-*private*.



Gambar 5.6 Tampilan di Github

Ini merupakan tampilan di github setelah file di-*push*. Nampak disitu tertulis "Update Source Code Python". Ini merupakan pesan yang sama dengan pesan yang ada pada *commit* pada gambar sebelumnya.