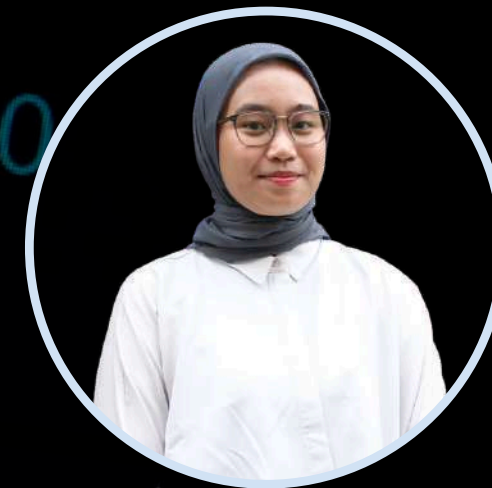


# VITALS

## Vital Information Tracking and Assessment Logic System



Alicia Kiyoumi B.  
2306168952

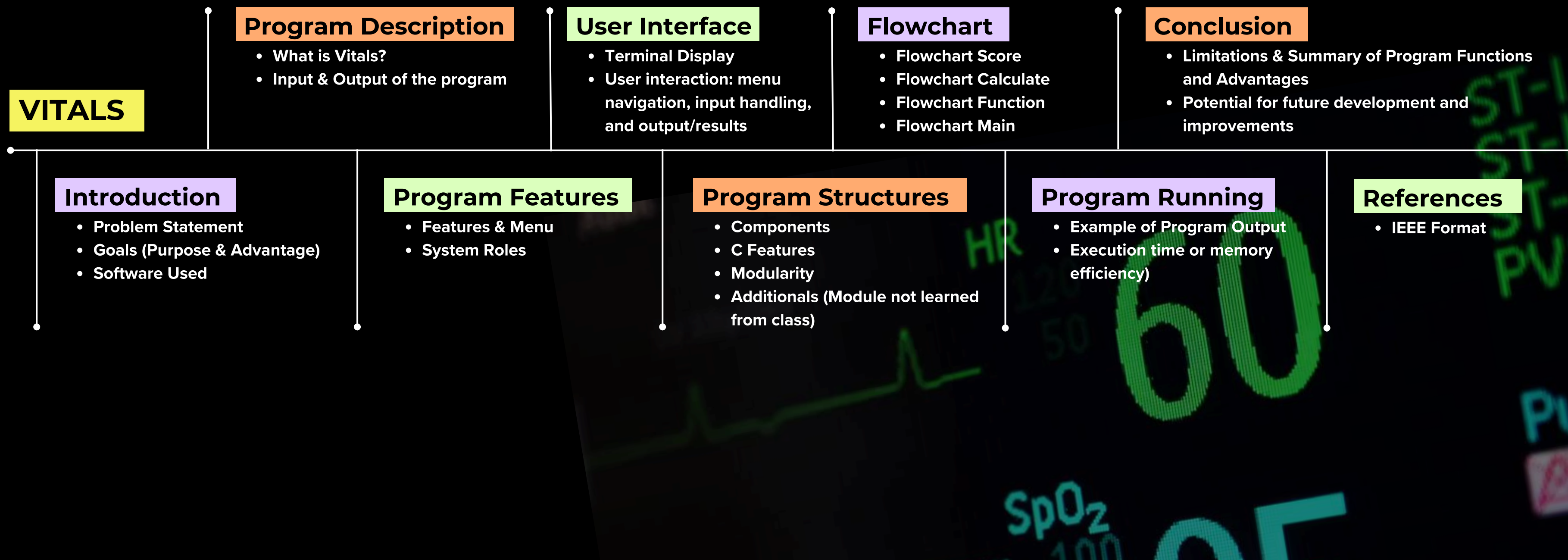


Aqila Ananti  
2306251582



Belva Nayla S.  
2306267201

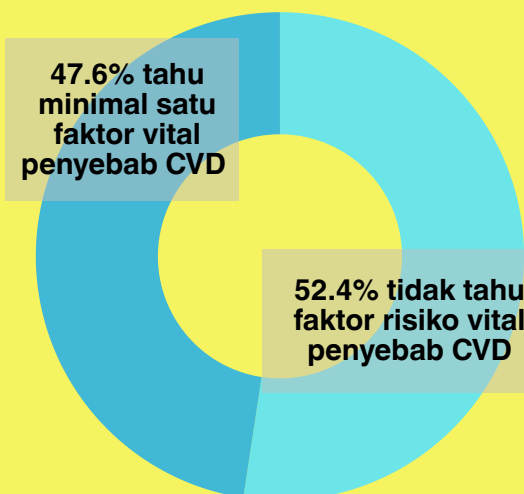
# Contents of VITALS





# Introduction

## PROBLEM



1.574

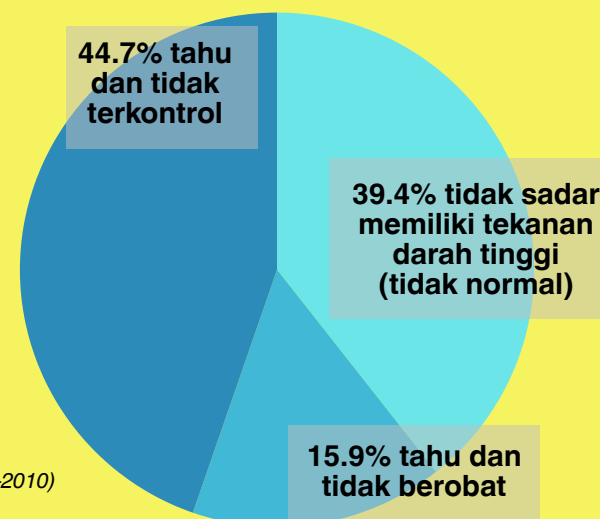
dari 3.000 orang tidak sadar akan risiko faktor vital penyebab CVD

(BMC Public Health, 2020)

14,1 juta jiwa dari 35,8 juta

tidak sadar akan penyakit hipertensi yg diderita, dimana tekanan darah tidak normal

(CDC MMWR, data NHANES 2003–2010)



## Key Takeaways

- Kesadaran kesehatan vital **sangat rendah**.
- **Mayoritas tidak tahu** bahwa hal-hal vital dengan kadar tidak normal adalah indikator **risiko kesehatan** serius.

## GOALS

- Mengotomatisasi **evaluasi tanda vital** pasien agar akurat dan efisien.
- **Mengklasifikasikan risiko** kesehatan berdasarkan data vital dengan sistem penilaian yang jelas.
- Menyediakan **rekomendasi gaya hidup** berdasarkan kategori risiko untuk mendukung pencegahan penyakit kronis.
- Memfasilitasi **pencatatan data pasien** secara sistematis.

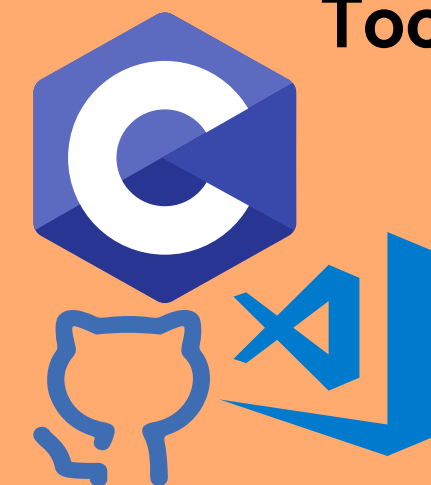
## SOLUTION

# VITALS

Vital Information  
Tracking and  
Assessment Logic  
System

Sistem Monitoring Pasien Berbasis C untuk Evaluasi Tanda Vital dan Risiko Kesehatan

## Tools Used



- Bahasa Pemrograman C
- Visual Studio Code (VS Code)
- Git + GitHub (Version Control)
- GCC / MinGW (kompilasi program C)

# Deskripsi Program

**VITALS** adalah sistem monitoring pasien yang dirancang untuk **mengumpulkan, mengevaluasi, dan mengklasifikasikan tanda-tanda vital** serta **risiko** kesehatan pasien secara dinamis dan efisien.

## Input

Pengguna akan memasukkan informasi berikut untuk setiap pasien:

### 1. **Identitas Pasien**

- Nama
- Usia
- Gender

### 2. **Status Berat Badan (BMI)**

- Tinggi badan
- Berat badan

### 3. **Parameter Kesehatan**

- Tekanan darah sistolik (mmHg)
- Tekanan darah diastolik (mmHg)
- Suhu tubuh (°C)
- Denyut jantung (bpm)
- Kadar glukosa (mmol/L)
- Kadar kolesterol (mmol/L)
- Status merokok (opsional)

### 4. **Tipe Pengguna (Role):**

- Admin
- Dokter
- Perawat

## Output

- Tampilan data pasien yang telah dimasukkan
- Status risiko kesehatan berdasarkan hasil analisis
- File output dalam format .txt atau .csv untuk dokumentasi data pasien

Low Risk

Moderate Risk

High Risk



# Program Features

## Main Highlights

### Role-Based Access Control

(melindungi data medis sensitif & mencegah kesalahan operasional)

### Patient Data Management

(7 Health Parameters)

### Health Risk Analysis & Evaluation

(menghasilkan skor risiko & rekomendasi gaya hidup)

### Data Persistence

(menyimpan & modifikasi file)

### Validated Data Input

(memastikan integritas dan akurasi data, seperti rentang nilai, format tanggal, tipe data)

## Admin

Add Patient

Display Patients

Delete Patient

Update Patient

Save File (.txt)

Save CSV (.csv)

New Database

Find Patient

Export 1 Patient

Patient Status

## Dokter

Add Patient

Display Patients

Save File (.txt)

Save CSV (.csv)

New Database

Find Patient

Export 1 Patient

Patient Status

## Perawat

Display Patients

Export 1 Patient

Find Patient

Patient Status



# Program Features

## Main Highlights

**Role-Based Access Control**

(melindungi data medis sensitif & mencegah kesalahan operasional)

**Patient Data Management**

(7 Health Parameters)

**Health Risk Analysis & Evaluation**

(menghasilkan skor risiko & rekomendasi gaya hidup)

**Data Persistence**

(menyimpan & modifikasi file)

**Validated Data Input**

(memastikan integritas dan akurasi data, seperti rentang nilai, format tanggal, tipe data)

Sources : [1], [2], [3], [4], [5], [6], [7], [8], [9]

Health Parameters	Healthy (Low Risk)	Needs Attention (Moderate Risk)	High Risk
BMI	18.5 - 24.9	25 - 29.9   < 18.5	≥ 30
Glucose (mmol/L)	< 7.8	7.8 - 11.0	≥ 11.1
Blood Pressure (mmHg)	< 130/80	< 160/100	≥ 160/100
Cholesterol (mmol/L)	< 5.0	5.0 - 7.4	≥ 7.5
Smoker Status	no	-	yes
Combined Total Score	0 - 2	3-5	> 5



# Program Features

## Main Highlights

### Role-Based Access Control

(melindungi data medis sensitif & mencegah kesalahan operasional)

### Patient Data Management

(7 Health Parameters)

### Health Risk Analysis & Evaluation

(menghasilkan skor risiko & rekomendasi gaya hidup)

### Data Persistence

(menyimpan & modifikasi file)

### Validated Data Input

(memastikan integritas dan akurasi data, seperti rentang nilai, format tanggal, tipe data)

## Health Risk Analysis and Evaluation

Status : Sehat

Diagram Risiko (0-6):  
[ -\*----- ]

Rekomendasi Gaya Hidup:  
- Pertahankan pola hidup sehat.

Status : Perlu Perhatian

Diagram Risiko (0-6):  
[ ---\*--- ]

Rekomendasi Gaya Hidup:  
- Kurangi konsumsi gula dan garam.  
- Rutin olahraga ringan.

Status : Risiko Tinggi

Diagram Risiko (0-6):  
[ -----\* ]

Rekomendasi Gaya Hidup:  
- Konsultasi rutin dengan dokter.  
- Perubahan gaya hidup menyeluruh diperlukan.

## Data Persistence

Pilihan Anda: 5  
Data berhasil disimpan ke 'laporan\_pasien.txt'.

Pilihan Anda: 6  
Data berhasil diekspor ke 'laporan\_pasien.csv'

Pilihan Anda: 9  
Masukkan ID Pasien yang ingin disimpan laporannya: 4  
Laporan untuk pasien ID 4 berhasil disimpan ke 'laporan\_pasien\_4.txt'.  
Laporan pasien dengan ID 4 berhasil disimpan.

## Validated Data Input

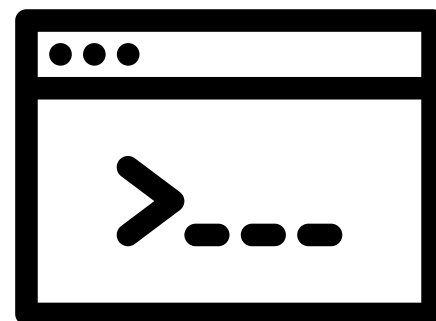
Jenis Kelamin (pria/wanita): wania  
Jenis kelamin tidak valid. Silakan masukkan 'pria' atau 'wanita'.

Tinggi (meter) : 170  
Tinggi harus antara 0.4 dan 3.0 meter. Silakan coba lagi.

Tanggal (DD/MM/YYYY) : 05/31/2025  
Tanggal tidak valid. Silakan coba lagi.

# User Interface

## Command-Line Interface



Text-Based

Structured & Guided

Dynamic & Responsive

### Tampilan Awal & Autentikasi Role User

```
=====
VITALS: Vital Information Tracking and Assessment Logic System
=====
Masukkan nama Anda: cia
Masukkan peran Anda (Admin/Dokter/Perawat): Admin
=====
```

## User Interaction Flow

Autentikasi Sederhana

Menu Dinamis

Pemilihan Aksi (Switch Case)

Feedback yang jelas

### Menu Dinamis

Menu (User: admin):

1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar

Menu (User: perawat):

2. Tampilkan Data Pasien
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar

Apakah Anda ingin:

1. Ganti Nama dan Role
2. Keluar Program

Pilihan Anda: 1

Masukkan nama Anda: cia

Masukkan peran Anda (Admin/Dokter/Perawat): perawat

```
▼ FINPRO_PROGC_GROUP7
C database.c
C database.h
C main.c
E medicalChecku... M
C patient.c
C patient.h
C progress1.c
E progress1.exe
C utils.c
C utils.h
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "patient.h"
#include "database.h"
#include "utils.h"
```

Struktur  
File



# Modularity

Setiap fitur dipecah ke file dan fungsi terpisah.

**main.c**                      Menu interaktif dan manajemen role

**patient.c**                      Perhitungan BMI, skor risiko, dan evaluasi pasien

**database.c**                      Manajemen database pasien (add, delete, update, export)

**utils.c**                      Fungsi pendukung seperti createAndAddPatient(), validasi ID, buffer

**header files**                      patient.h, database.h, utils.h

# Program Components

Program ini mengintegrasikan semua komponen dalam satu sistem yang modular.

main.c

main program, UI, menu program (role)

patients.c

Struktur data pasien dan fungsi evaluasi kesehatan; menghitung skor risiko, BMI, dan memberi rekomendasi.

patients.h

Mendefinisikan struktur struct Patient, Deklarasi semua fungsi evaluasi kesehatan, seperti calculateBMI(), scoreGlucose(), evaluatePatient(), dll.

database.c

Menyimpan dan mengelola data pasien: tambah, hapus, update, simpan/load ke file, ekspor CSV, dll.

database.h

Deklarasi fungsi-fungsi database: addPatient(), deletePatient(), initDB(), exportToCSV(), dll.  
Deklarasi variabel eksternal: struct Patient patients[MAX];, int patientCount;, nextID;

utils.c

Utilitas tambahan, input pasien, buffer, validasi ID, fungsi pembantu lainnya.

utils.h

Deklarasi fungsi utilitas:  
• clearInputBuffer()  
• createAndAddPatient(), updatePatientData(), dll.

## Tujuan

- Struktur data efisien
- Pengelolaan memori terkontrol
- Penyimpanan data persisten.

## C Features

- if / switch
- for, while, do while
- Input dari user
- Function
- Array
- Pointer
- Struct



# C Features

Setiap fitur digunakan sesuai kebutuhan dan terintegrasi antarfile, berikut highlight beberapa contoh kode implementasi.

**if-else****Mengecek kondisi logis (conditional)**

```
if (scanf(...) != 1)
```

**switch-case****Memilih salah satu dari banyak kasus(conditional)**

```
switch(choice)
```

**for loop****Looping ketika jumlah iterasi diketahui**

```
for (int i = 0; role[i]; i++)
```

**while loop****Looping iterasi selama kondisi benar**

```
while (!exitProgram)
```

**do loop****Looping dijalankan setidaknya sekali**

```
do while di patients.c
```

**user input****Mengambil data dari pengguna**

```
inputPatient() berisi scanf
```

**function****Blok kode dengan tugas tertentu**

```
displayMenu(), checkPermission(), main()
```

**array****Kumpulan banyak elemen dengan tipe yang sama, memori alokasi statis**

```
struct Patient patients[MAX]
```

**pointer****Variabel yang menyimpan alamat memori dari variabel lain**

```
inputPatient(struct Patient* p)
```

**struct****Mengelompokkan beberapa data berbeda dalam satu variabel**

```
struct Patient patients[MAX];
```

**file handling****membuka, membaca, menulis, dan menutup file dalam program**

```
FILE* file = fopen("laporan_pasien.csv", "r");  
while (fgets(lineBuffer, sizeof(lineBuffer), file) != NULL) {  
    sscanf(lineBuffer, ...);  
}  
fclose(file);
```



main.c

# Main Components

## LIBRARIES

```
#include <stdio.h> //import library c standar
#include <string.h> //import library modifikasi string
#include <stdlib.h> //import library untuk fungsi utilitas umum
#include <ctype.h> //import library untuk fungsi karakter
#include "patient.h" //import header file patient
#include "database.h" //import header file database
#include "utils.h" //import header file utils
```

## DEFINE ACTION

```
#define ACTION_ADD_PATIENT "add_patient" //definisi aksi menambah pasien
#define ACTION_DISPLAY_PATIENTS "display_patients" //definisi aksi tampilan pasien
#define ACTION_DELETE_PATIENT "delete_patient" //definisi aksi hapus pasien
#define ACTION_UPDATE_PATIENT "update_patient" //definisi aksi update pasien
#define ACTION_SAVE_FILE "save_file" //definisi aksi simpan ke file .txt
#define ACTION_SAVE_CSV "save_csv" //definisi aksi simpan ke file .csv
#define ACTION_FIND_PATIENT "find_patient" //definisi aksi cari pasien
#define ACTION_VIEW_PATIENT_STATUS "view_patient_status" //definisi aksi ngeliat status pasien
#define ACTION_EXPORT_ONE_PATIENT "export_one_patient" //definisi aksi simpan satu pasien
#define ACTION_NEW_DATABASE "new_database" //definisi aksi membuat database baru
```

## SWITCH CASE

```
switch(choice){
//pakai switch case untuk proses pilihan user dari tampil case 5:
case 1: { //pilihan tambah pasien
if(checkPermission(role, ACTION_ADD_PATIENT)){
createAndAddPatient(); //memanggil fungsi untuk tambah pasien
else {
printf("Anda tidak memiliki izin untuk menambah pasien.\n");
break;
}
}
case 2: { //pilihan tampilan pasien
displayPatients(); //memanggil fungsi untuk menampilkan pas
break;
}
case 3: { //pilihan hapus pasien
if(checkPermission(role, ACTION_DELETE_PATIENT)){
deletePatientData(); //memanggil fungsi untuk hapus pasien
else {
printf("Anda tidak memiliki izin untuk menghapus pasien.\n");
break;
}
}
case 4: { //pilihan update pasien
if(checkPermission(role, ACTION_UPDATE_PATIENT)){
updatePatientData(); //memanggil fungsi untuk update pasien
else {
printf("Anda tidak memiliki izin untuk memperbarui pasien.\n");
break;
}
}
case 5: { //pilihan simpan ke file .txt
if(checkPermission(role, ACTION_SAVE_FILE)){
saveToFile(); //memanggil fungsi untuk simpan ke file .txt
else {
printf("Anda tidak memiliki izin untuk menyimpan ke file.\n");
break;
}
}
case 6: { //pilihan simpan ke file .csv
if(checkPermission(role, ACTION_SAVE_CSV)){
exportToCSV(); //memanggil fungsi untuk simpan ke file .csv
else {
printf("Anda tidak memiliki izin untuk mengeksport ke CSV.\n");
break;
}
}
case 7: { //pilihan membuat database baru
if(checkPermission(role, ACTION_NEW_DATABASE)){
newDatabase(); //memanggil fungsi untuk membuat database baru
else {
printf("Anda tidak memiliki izin untuk membuat database baru.\n");
break;
}
}
case 8: { //pilihan cari pasien
if(checkPermission(role, ACTION_FIND_PATIENT)){
searchPatientData(); //memanggil fungsi untuk cari pasien
else {
printf("Anda tidak memiliki izin untuk mencari pasien.\n");
break;
}
}
case 9: { //pilihan simpan lap
if(checkPermission(role, ACTION_EXPORT_ONE_PATIENT)){
saveSinglePatientData(); //me
else {
printf("Anda tidak memiliki izin
}
}
case 10: { //pilihan lihat stat
if(checkPermission(role, ACTION_VIEW_PATIENT_STATUS)){
checkPatientStatusData(); //me
else {
printf("Anda tidak memiliki izin
}
}
case 0: { //pilihan keluar dari
int subChoice; //variabel pilihan
printf("Apakah Anda ingin:\n");
printf("1. Ganti Nama dan Role\n");
printf("2. Keluar Program\n");
printf("Pilihan Anda: ");
}
}
```

## STATIC FUNCTION

```
static int checkPermission(const char* role, const char* action) {}
```

```
static void displayMenu(const char* currentRole_param) {}
```

## DYNAMIC PROGRAM

```
scanf("%s", role); scanf("%d", &subChoice);
```

```
scanf("%s", role); if (scanf("%d", &choice) != 1) {}
```

## FOR & WHILE LOOPING

```
while (!exitProgram) { //loop utama progr
printf("Masukkan nama Anda: "); //minta input nama
scanf("%s", &adminName); //membaca nama user

int validRole = 0; //periksa validitas role
while (!validRole) { //looping utk memastikan r
printf("Masukkan peran Anda (Admin/Dokter/Perawat)");
scanf("%s", &role); //membaca input
for (int i = 0; role[i]; i++) { //membaca penuh
role[i] = tolower(role[i]);
}
//memeriksa apakah rolenya udah valid apa blm
if (strcmp(role, "admin") == 0 || strcmp(role, "do
validRole = 1; //role valid ketika termas
} else {
printf("Peran tidak valid. Silakan coba lagi.\n");
}
}

while (!exitProgram) { //loop utama program
int sessionEnded = 0; //mengakhiri sesi
while (!sessionEnded) { //loop sesi user
displayMenu(role); //tampilan menu berdasarkan role yg dipilih

if (scanf("%d", &choice) != 1) { //input pilihan user
printf("Input tidak valid. Masukkan angka 0-10.\n"); //kalo pilihan dilu
while (getchar() != '\n'); //menghapus karakter newline (sisa)
continue; //kembali ke awal loop
} while (getchar() != '\n'); //menghapus karakter newline

switch(choice){ //pakai switch case untuk proses pilihan user dari tampil
case 1: { //pilihan tambah pasien
if(checkPermission(role, ACTION_ADD_PATIENT)){
createAndAddPatient(); //memanggil fungsi untuk tambah pasien
else {
printf("Anda tidak memiliki izin untuk menambah pasien.\n");
break;
}
}
case 2: { //pilihan tampilan pasien
displayPatients(); //memanggil fungsi untuk menampilkan pa
break;
}
case 3: { //pilihan hapus pasien
if(checkPermission(role, ACTION_DELETE_PATIENT)){
deletePatientData(); //memanggil fungsi untuk hapus pasien
else {
printf("Anda tidak memiliki izin untuk menghapus pasien.\n");
break;
}
}
case 4: { //pilihan update pasien
if(checkPermission(role, ACTION_UPDATE_PATIENT)){
updatePatientData(); //memanggil fungsi untuk update pasien
else {
printf("Anda tidak memiliki izin untuk memperbarui pasien.\n");
break;
}
}
case 5: { //pilihan simpan ke file .txt
if(checkPermission(role, ACTION_SAVE_FILE)){
saveToFile(); //memanggil fungsi untuk simpan ke file .txt
else {
printf("Anda tidak memiliki izin untuk menyimpan ke file.\n");
break;
}
}
case 6: { //pilihan simpan ke file .csv
if(checkPermission(role, ACTION_SAVE_CSV)){
exportToCSV(); //memanggil fungsi untuk simpan ke file .csv
else {
printf("Anda tidak memiliki izin untuk mengeksport ke CSV.\n");
break;
}
}
case 7: { //pilihan membuat database baru
if(checkPermission(role, ACTION_NEW_DATABASE)){
newDatabase(); //memanggil fungsi untuk membuat database baru
else {
printf("Anda tidak memiliki izin untuk membuat database baru.\n");
break;
}
}
case 8: { //pilihan cari pasien
if(checkPermission(role, ACTION_FIND_PATIENT)){
searchPatientData(); //memanggil fungsi untuk cari pasien
else {
printf("Anda tidak memiliki izin untuk mencari pasien.\n");
break;
}
}
case 9: { //pilihan simpan lap
if(checkPermission(role, ACTION_EXPORT_ONE_PATIENT)){
saveSinglePatientData(); //memanggil fungsi untuk simpan satu pasien
else {
printf("Anda tidak memiliki izin untuk menyimpan satu pasien.\n");
break;
}
}
case 10: { //pilihan lihat status
if(checkPermission(role, ACTION_VIEW_PATIENT_STATUS)){
checkPatientStatusData(); //memanggil fungsi untuk lihat status pasien
else {
printf("Anda tidak memiliki izin untuk ngeliat status pasien.\n");
break;
}
}
case 0: { //pilihan keluar dari
int subChoice; //variabel pilihan
printf("Apakah Anda ingin:\n");
printf("1. Ganti Nama dan Role\n");
printf("2. Keluar Program\n");
printf("Pilihan Anda: ");
}
}
}
```

## IF-ELSE CONDITIONAL

```
if (strcmp(action, ACTION_DISPLAY_PATIENTS) == 0 ||
strcmp(action, ACTION_VIEW_PATIENT_STATUS) == 0 ||
strcmp(action, ACTION_EXPORT_ONE_PATIENT) == 0 ||
strcmp(action, ACTION_FIND_PATIENT) == 0) {
//admin, dokter, dan perawat yang memiliki izin
if (strcmp(role, "admin") == 0 || strcmp(role, "dokter") == 0 || strcmp(role, "perawat") == 0) {
return 1; //izin diberikan
}
}

if (strcmp(action, ACTION_ADD_PATIENT) == 0 ||
strcmp(action, ACTION_SAVE_FILE) == 0 ||
strcmp(action, ACTION_NEW_DATABASE) == 0 ||
strcmp(action, ACTION_SAVE_CSV) == 0) {
//hanya admin dan dokter yang memiliki izin
if (strcmp(role, "admin") == 0 || strcmp(role, "dokter") == 0) {
return 1; //izin diberikan
}
}

if (strcmp(action, ACTION_DELETE_PATIENT) == 0 ||
strcmp(action, ACTION_UPDATE_PATIENT) == 0) {
//hanya admin yang memiliki izin
if (strcmp(role, "admin") == 0) {
return 1; //izin diberikan
}
}

return 0; //izin ditolak
```

```
if (checkPermission(currentRole_param, ACTION_ADD_PATIENT)) {
printf(" 1. Tambah Pasien Baru\n"); //opsi tambah pasien
}
printf(" 2. Tampilkan Data Pasien\n"); //opsi tampilan pasien
if (checkPermission(currentRole_param, ACTION_DELETE_PATIENT)) {
printf(" 3. Hapus Data Pasien\n"); //opsi hapus pasien
}
if (checkPermission(currentRole_param, ACTION_UPDATE_PATIENT)) {
printf(" 4. Perbarui Data Pasien\n"); //opsi update pasien
}
if (checkPermission(currentRole_param, ACTION_SAVE_FILE)) {
printf(" 5. Simpan ke File\n"); //opsi simpan file ke .txt
}
if (checkPermission(currentRole_param, ACTION_NEW_DATABASE)) {
printf(" 6. Ekspor ke CSV\n"); //opsi simpan file ke .csv
}
if (checkPermission(currentRole_param, ACTION_NEW_DATABASE)) {
printf(" 7. Buat Database Baru\n"); //opsi membuat database baru
}
printf(" 8. Cari Pasien\n"); //opsi mencari pasien
printf(" 9. Simpan Laporan Satu Pasien\n"); //opsi simpan laporan satu pasien
printf(" 10. Lihat Status Pasien\n"); //opsi ngelihat status pasien
printf(" 0. Keluar\n"); //opsi keluar
printf("===== \n");
printf("Pilihan Anda: "); //opsi minta input pilihan user
```



# Main Components

## LIBRARIES

```
#include "patient.h" // Definisi struktur data pasien
#include "utils.h" // Fungsi-fungsi utilitas pendukung (misalnya, clearInputBuffer)
#include <string.h> // Untuk operasi string (misalnya, strcmp, strlen)
#include <ctype.h> // Untuk fungsi manipulasi karakter (misalnya, toupper, tolower)
#include <stdio.h> // Untuk fungsi input/output standar (misalnya, printf, scanf)
```

## FUNCTION

```
float calculateBMI(float weight, float height);
int scoreBMI(float bmi, const char* gender);
int scoreGlucose(float glucose);
int scoreBP(float systolic, float diastolic);
int scoreChol(float cholesterol);
int scoreSmoker(const char* status);
int calculateTotalRiskScore(const struct Patient* p);
const char* overallRiskLevel(int totalScore);
void inputPatient(struct Patient* p);
void evaluatePatient(const struct Patient* p);
void lifestyleRecommendation(int score);
void asciiDiagram(int score);
```

## IF ELSE

```
const char* overallRiskLevel(int totalScore) {
    if (totalScore <= 2) return "Sehat"; // Skor rendah
    else if (totalScore <= 5) return "Perlu Perhatian";
    else return "Risiko Tinggi"; // Skor tinggi mengindil
}
```

## DYNAMIC PROGRAM

```
scanf(" %49[^\n]", p->name); scanf("%f", &p->weight) != 1);
scanf("%9s", p->gender); if (scanf("%d", &p->age) != 1){}
```

## POINTER

```
void evaluatePatient(const struct Patient* p) {
    float bmi = calculateBMI(p->weight, p->height);
    int score = 0; // Inisialisasi skor total risiko
    score += scoreBMI(bmi, p->gender); // Menambahkan
    score += scoreGlucose(p->glucose); // Menambahkan
    score += scoreBP(p->systolic, p->diastolic); //
    score += scoreChol(p->cholesterol); // Menambahk
    score += scoreSmoker(p->smokerStatus); // Menamk
```

## DO WHILE

```
do {
    printf("Status Merokok (ya/tidak) : "); // Meminta input status merokok
    scanf(" %5s", p->smokerStatus); // Membaca input (maks. 5 karakter), mengabaikan spa
    clearInputBuffer(); // Membersihkan buffer

    for (int i = 0; p->smokerStatus[i]; i++) { // Iterasi pada karakter input
        p->smokerStatus[i] = tolower(p->smokerStatus[i]); // Konversi ke huruf kecil
    }

    if (strcmp(p->smokerStatus, "ya") != 0 && strcmp(p->smokerStatus, "tidak") != 0) { //
        printf("Status merokok tidak valid. Silakan masukkan 'ya' atau 'tidak'. \n"); //
    }
} while (strcmp(p->smokerStatus, "ya") != 0 && strcmp(p->smokerStatus, "tidak") != 0); /
```

## STRUCT & ARRAY

```
struct Patient {
    int id;
    char name[50];
    int age;
    char gender[10];
    float height;
    float weight;
    float systolic;
    float diastolic;
    float temperature;
    int heartRate;
    float glucose;
    float cholesterol;
    char smokerStatus[6];
    char date[20];
};
```



# Main Components

## LIBRARIES

```
#include "database.h"           // Header untuk fungsi database
#include "patient.h"           // Header untuk definisi dan fungsi pasien
#include <stdio.h>              // Library standar untuk fungsi input/output
#include <string.h>             // Library fungsi string
#include <stdlib.h>             // Library fungsi utilitas
#include <ctype.h>              // Library fungsi karakter
```

## STRUCT & POINTER

```
int updatePatient(int id, const struct Patient* p) {
    struct Patient* found = searchPatient(id); // Cari pasien
    if (!found) {
        printf("Pasien dengan ID %d tidak ditemukan.\n", id);
        return -1; // Kode gagal
    }
    *found = *p; // Perbarui data
    printf("Data pasien berhasil diperbarui.\n");
    return 0; // Kode sukses
}
```

## LOOPING

```
struct Patient* searchPatient(int id) {
    for (int i = 0; i < patientCount; i++) { //
        if (patients[i].id == id) { //
            return &patients[i]; //
        }
    }
    return NULL; //
}
```

## VARIABLE DECLARING

```
struct Patient patients[MAX];
int patientCount;
int nextID;
```

## FILE HANDLING

```
FILE* file = fopen("laporan_pasien.csv", "w"); // Buka file CSV untuk tulis
if (!file) { // Jika gagal buka file
    printf("Gagal membuka file CSV 'laporan_pasien.csv'\n"); // Info kegagalan
    return; // Kembali tanpa ekspor
}
// Tulis header kolom CSV
fprintf(file, "ID,Nama,Usia,Jenis Kelamin,Tinggi(m),Berat(kg),BMI,Glukosa(mmol/L),Kolesterol\n");
for (int i = 0; i < patientCount; i++) { // Loop semua pasien
    struct Patient* p = &patients[i]; // Pointer ke pasien
    float bmi = calculateBMI(p->weight, p->height); // Hitung BMI
    int score = calculateTotalRiskScore(p); // Hitung skor risiko
    const char* status = overallRiskLevel(score); // Status risiko

    // Tulis data pasien dalam format CSV
    fprintf(file, "%d,\"%s\",%d,%s,%.2f,%.2f,%.2f,%.2f,%.0f,%.0f,%.1f,%d,%s,%s,%s\n",
        p->id, p->name, p->age, p->gender,
        p->height, p->weight, bmi, p->glucose,
        p->cholesterol, p->systolic, p->diastolic,
        p->temperature, p->heartRate,
        p->smokerStatus, p->date, status);
}
fclose(file); // Tutup file CSV
printf("Data berhasil diekspor ke 'laporan_pasien.csv'\n"); // Info ekspor sukses
```

## FUNCTIONS

```
void initDB(void); // Menginisialisasi database pasien dan
int addPatient(struct Patient* pDetails); // Menambahkan p
struct Patient* searchPatient(int id); // Mencari pasien b
int deletePatient(int id); // Menghapus pasien dari databa
int updatePatient(int id, const struct Patient* pDetails);
void displayPatients(void); // Menampilkan semua data pasi
void saveToFile(void); // Menyimpan seluruh data pasien ke
void exportToCSV(void); // Mengekspor semua data pasien ke
int saveSinglePatient(int id); // Menyimpan laporan untuk
void newDatabase(void); // Menghapus semua pasien dan mere
void searchAndDisplayPatientDetails(int id); // Memilih pa
int viewPatientStatus(int id); // Melihat status dan nama
```

## IF CONDITIONAL

```
int updatePatient(int id, const struct Patient* p) {
    struct Patient* found = searchPatient(id); // Cari pasien b
    if (!found) {
        printf("Pasien dengan ID %d tidak ditemukan.\n", id);
        return -1; // Kode gagal
    }
    *found = *p; // Perbarui data p
    printf("Data pasien berhasil diperbarui.\n");
    return 0; // Kode sukses
}
```



# Main Components

## LIBRARIES

```
#include "utils.h" // Mengimpor header utils
#include "database.h" // Mengimpor header fungsi database
#include "patient.h" // Mengimpor header definisi dan fungsi pasien
```

## DYNAMIC PROGRAM

```
printf("Masukkan ID Pasien yang ingin diperiksa statusnya: ");
if (scanf("%d", &id) != 1) { // Baca dan validasi input ID
    clearInputBuffer(); // Bersihkan buffer jika input tidak valid
    printf("Input ID tidak valid.\n"); // Pesan error input tidak valid
    return; // Hentikan fungsi jika input tidak valid
}
```

## STRUCT

```
struct Patient p; // Definisi struktur Patient
printf("--- Memasukkan detail baru untuk pasien\n"); // Prompt
printf("Catatan: ID Pasien tidak dapat kosong\n"); // Catatan
inputPatient(&p); // Fungsi inputPatient
p.id = id; // Assign ID
if (updatePatient(id, &p) == 0) { // Fungsi updatePatient
    evaluatePatient(&p); // Fungsi evaluatePatient
}
```

## FUNCTION

```
void clearInputBuffer(void); // Deklarasi fungsi clearInputBuffer
void createAndAddPatient(void); // Fungsi createAndAddPatient
void deletePatientData(void); // Deklarasi fungsi deletePatientData
void updatePatientData(void); // Deklarasi fungsi updatePatientData
void searchPatientData(void); // Deklarasi fungsi searchPatientData
void saveSinglePatientData(void); // Fungsi saveSinglePatientData
void checkPatientStatusData(void); // Fungsi checkPatientStatusData
```

## IF CONDITIONAL

```
void deletePatientData(void) {
    int id; // Variabel untuk menampung ID pasien
    printf("Masukkan ID Pasien yang ingin dihapus: ");
    if (scanf("%d", &id) != 1) { // Baca input ID dan validasi
        clearInputBuffer(); // Bersihkan buffer input
        printf("Input ID tidak valid.\n"); // Informasi input tidak valid
        return; // Menghentikan fungsi jika input tidak valid
    }
    clearInputBuffer(); // Bersihkan buffer input
    deletePatient(id); // Panggil fungsi untuk menghapus pasien
}
```

## POINTER

```
struct Patient p; // Definisi struktur Patient
printf("--- Memasukkan detail baru untuk pasien\n"); // Prompt
printf("Catatan: ID Pasien tidak dapat kosong\n"); // Catatan
inputPatient(&p); // Fungsi inputPatient
p.id = id; // Assign ID
if (updatePatient(id, &p) == 0) { // Fungsi updatePatient
    evaluatePatient(&p); // Fungsi evaluatePatient
}
```

## WHILE LOOPING

```
void clearInputBuffer(void) { // Variabel untuk menampung karakter
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```



# Additional

Module Not Learned from Class

## Export laporan ke file CSV

```
void exportToCSV(void) {
    FILE* file = fopen("laporan_pasien.csv", "w"); // Buka file CSV untuk tulis
    if (!file) {
        // Jika gagal buka file
        printf("Gagal membuka file CSV 'laporan_pasien.csv'!\n"); // Info kegagalan
        return; // Kembali tanpa ekspor
    }
    // Tulis header kolom CSV
    fprintf(file, "ID,Nama,Usia,Jenis Kelamin,Tinggi(m),Berat(kg),BMI,Glukosa(mmol/L),Kolesterol(mmol/L),TD Sistolik\n");
    for (int i = 0; i < patientCount; i++) { // Loop semua pasien
        struct Patient* p = &patients[i]; // Pointer ke pasien
        float bmi = calculateBMI(p->weight, p->height); // Hitung BMI
        int score = calculateTotalRiskScore(p); // Hitung skor risiko
        const char* status = overallRiskLevel(score); // Status risiko

        // Tulis data pasien dalam format CSV
        fprintf(file, "%d,\"%s\",%d,%s,%.2f,%.2f,%.2f,%.2f,%.2f,%.0f,%.0f,%.1f,%d,%s,%s,%s\n",
            p->id, p->name, p->age, p->gender,
            p->height, p->weight, bmi, p->glucose,
            p->cholesterol, p->systolic, p->diastolic,
            p->temperature, p->heartRate,
            p->smokerStatus, p->date, status);
    }
    fclose(file); // Tutup file CSV
    printf("Data berhasil diekspor ke 'laporan_pasien.csv'\n"); // Info ekspor sukses
}
```

## STATIC FUNCTION

```
static int checkPermission(const char* role, const char* action) {}
```

```
static void displayMenu(const char* currentRole_param) {}
```

## Menyimpan hanya 1 pasien ke txt

```
int saveSinglePatient(int id) {
    struct Patient* p = searchPatient(id); // Cari pasien berdasarkan ID
    if (!p) {
        printf("Pasien dengan ID %d tidak ditemukan untuk disimpan laporannya.\n", id);
        return -1; // Kode gagal
    }

    char filename[100]; // Buffer untuk nama file
    sprintf(filename, "laporan_pasien_%d.txt", id); // Format nama file
}
```

## Hapus File

```
void newDatabase(void) {
    patientCount = 0; //
    nextID = 1; //
    const char* txt = "laporan_pasien.txt";
    const char* csv = "laporan_pasien.csv";

    remove(txt); //
    remove(csv); //
    printf("Database baru telah dibuat.\n");
}
```

## Operator Ternary

```
for (int i = 0; i <= 6; i++) { // Iterasi
    if (i == (score > 6 ? 6 : score)) /
        printf("*"); // Tandai posisi s
    else
        printf("-"); // Bagian lain dari array
}
```

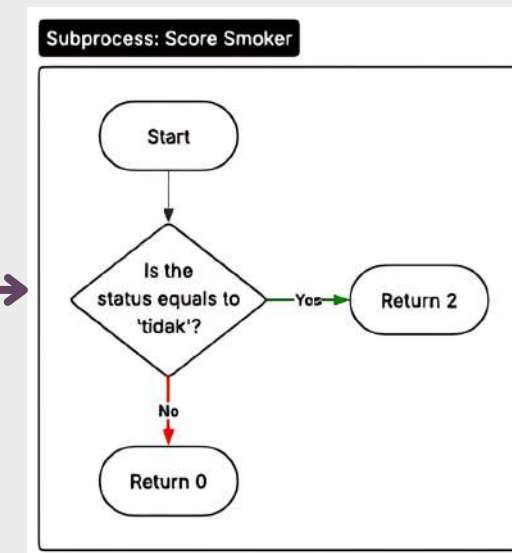


# Flowchart Score

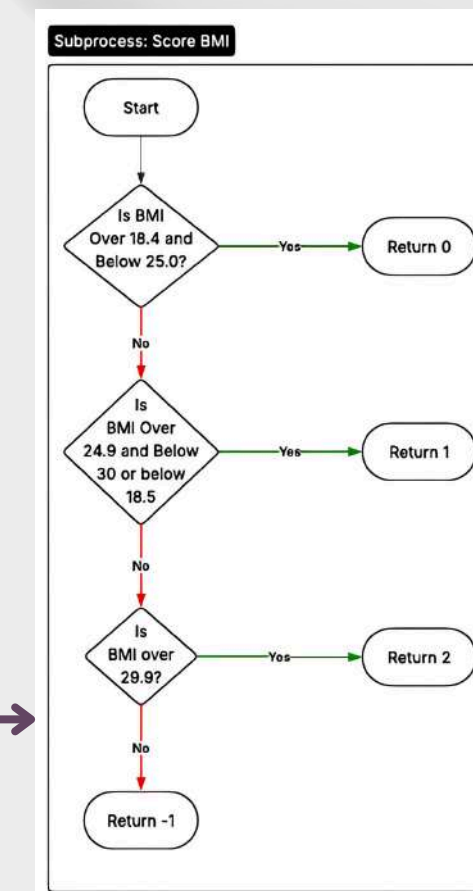
## Kode Score Smoker

```
// Fungsi untuk menentukan skor berdasarkan status merokok pasien
int scoreSmoker(const char* status) {
    if (strcmp(status, "tidak") == 0) return 0; // Jika tidak merokok, skor 0
    else return 2; // Jika merokok ("ya"), skor 2
}
```

## Score Smoker



## Score BMI



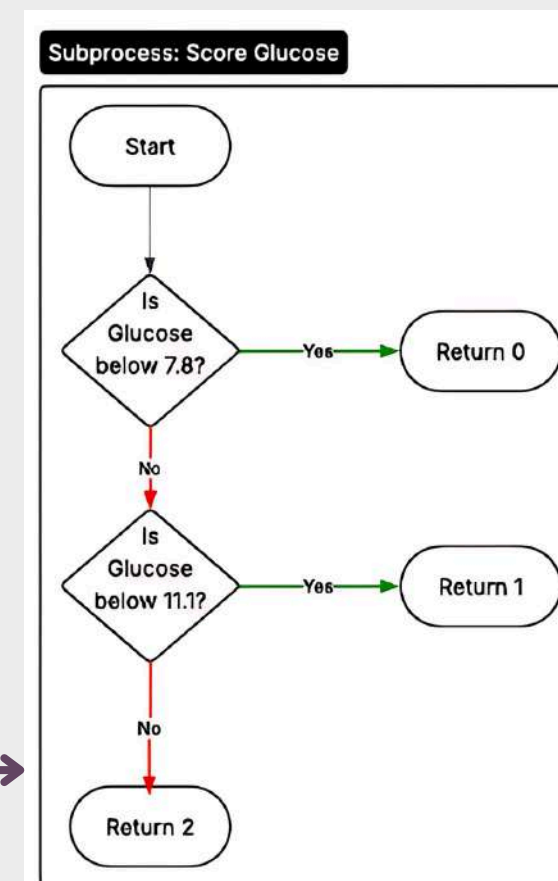
## Kode Score BMI

```
// Fungsi untuk menentukan skor berdasarkan nilai BMI
int scoreBMI(float bmi, const char* gender) {
    (void)gender; // Parameter gender saat ini tidak digunakan, ditandai untuk menghindari peringatan kompilator
    if (bmi >= 18.5 && bmi <= 24.9) return 0; // BMI ideal, skor 0
    else if ((bmi >= 25 && bmi <= 29.9) || (bmi < 18.5)) return 1; // BMI lebih atau kurang dari ideal, skor 1
    else if (bmi >= 30) return 2; // Kondisi obesitas, skor 2
    return -1; // Nilai default jika tidak ada kategori yang cocok (seharusnya tidak tercapai dengan input valid)
}
```

## Kode Score Glucose

```
// Fungsi untuk menentukan skor berdasarkan kadar glukosa
int scoreGlucose(float glucose) {
    if (glucose < 7.8) return 0; // Kadar glukosa normal, skor 0
    else if (glucose < 11.1) return 1; // Kadar glukosa pra-diabetes, skor 1
    else return 2; // Kadar glukosa diabetes, skor 2
}
```

## Score Glucose



Parameter patient.c



# Flowchart Score

## Kode Score Cholesterol

```
// Fungsi untuk menentukan skor berdasarkan kadar kolesterol
int scoreChol(float cholesterol) {
    if (cholesterol < 5.0) return 0; // Kadar kolesterol normal, skor 0
    else if (cholesterol < 7.5) return 1; // Kadar kolesterol batas tinggi, skor 1
    else return 2; // Kadar kolesterol tinggi, skor 2
}
```

## Kode Score Risk Level

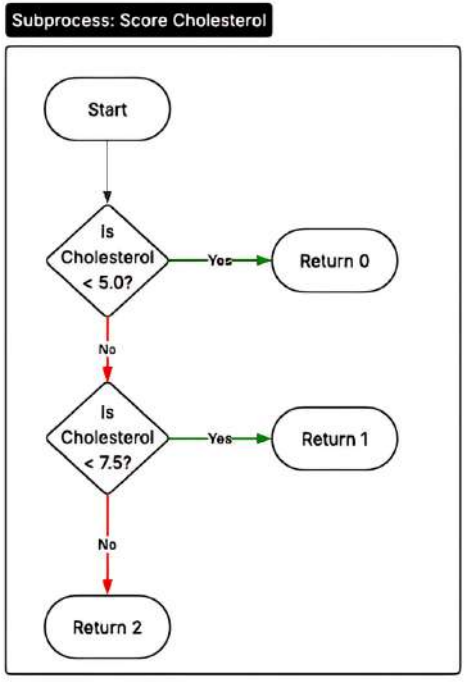
```
// Fungsi untuk menentukan level risiko keseluruhan berdasarkan total skor
const char* overallRiskLevel(int totalScore) {
    if (totalScore <= 2) return "Sehat"; // Skor rendah mengindikasikan level "Sehat"
    else if (totalScore <= 5) return "Perlu Perhatian"; // Skor sedang mengindikasikan "Perlu Perhatian"
    else return "Risiko Tinggi"; // Skor tinggi mengindikasikan "Risiko Tinggi"
}
```

## Kode Score Blood Pressure

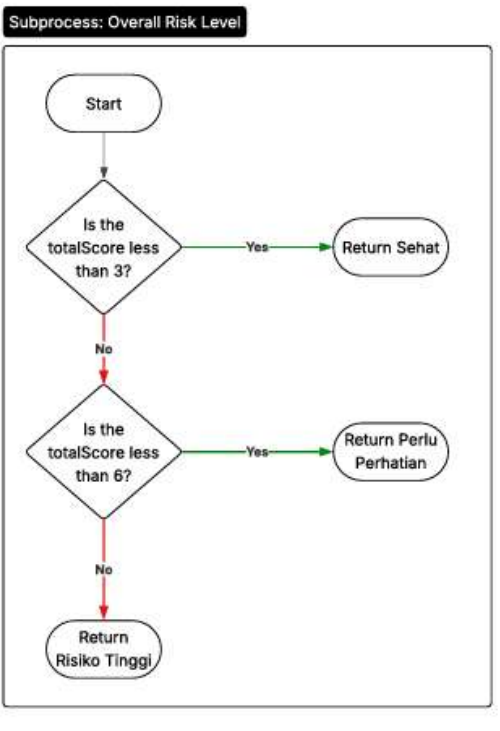
```
// Fungsi untuk menentukan skor berdasarkan tekanan darah
int scoreBP(float systolic, float diastolic) {
    if (systolic < 130 && diastolic < 80) return 0; // Tekanan darah normal, skor 0
    else if ((systolic < 160) || (diastolic < 100)) return 1; // Tekanan darah prahipertensi/hipertensi tahap 1, skor 1
    else return 2; // Tekanan darah hipertensi tahap 2 atau lebih, skor 2
}
```

Parameter patient.c

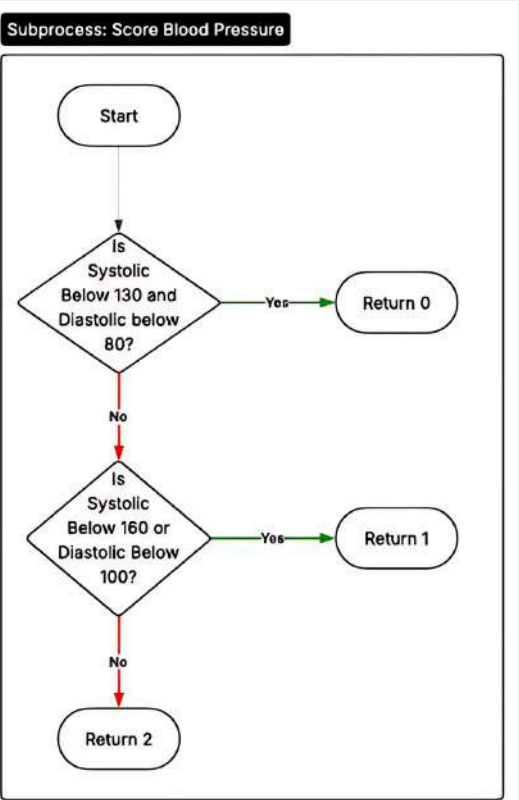
## Score Cholesterol



## Score Risk Level



## Score Blood Pressure

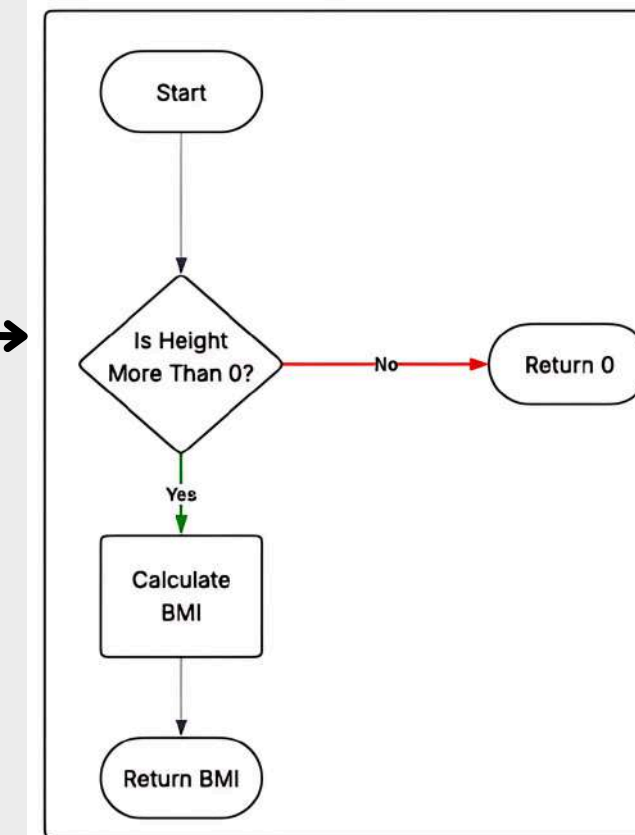


# Flowchart Calculate

## Calculate BMI

```
// Fungsi untuk menghitung Indeks Massa Tubuh (BMI)
float calculateBMI(float weight, float height) {
    if (height <= 0) return 0.0f; // Jika tinggi tidak valid (<= 0), kembalikan BMI 0
    return weight / (height * height); // Rumus standar BMI: berat / (tinggi * tinggi)
}
```

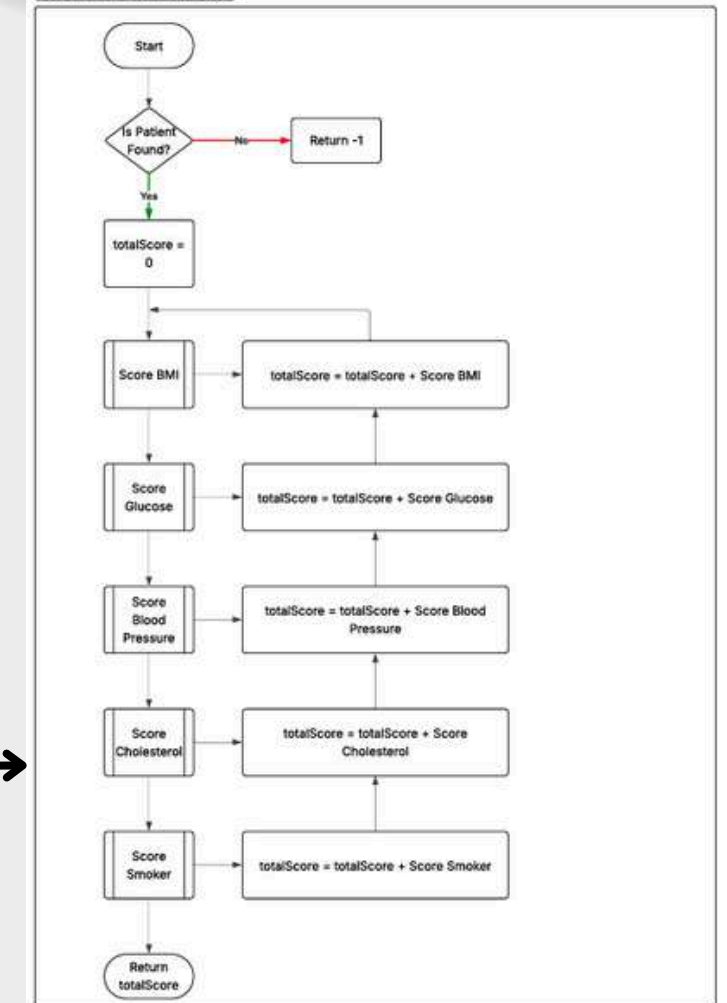
### Subprocess: Calculate BMI



## Calculate total risk

```
// Fungsi untuk menghitung total skor risiko kesehatan seorang pasien
int calculateTotalRiskScore(const struct Patient* p) {
    if (!p) return -1; // Jika pointer pasien null, kembalikan -1 sebagai indikasi error
    float bmi = calculateBMI(p->weight, p->height); // Hitung nilai BMI pasien
    int totalScore = 0; // Inisialisasi total skor dengan 0
    totalScore += scoreBMI(bmi, p->gender); // Akumulasi skor dari BMI
    totalScore += scoreGlucose(p->glucose); // Akumulasi skor dari glukosa
    totalScore += scoreBP(p->systolic, p->diastolic); // Akumulasi skor dari tekanan darah
    totalScore += scoreChol(p->cholesterol); // Akumulasi skor dari kolesterol
    totalScore += scoreSmoker(p->smokerStatus); // Akumulasi skor dari status merokok
    return totalScore; // Kembalikan totalskor risiko yang terhitung
}
```

### Subprocess: Calculate Total Risk

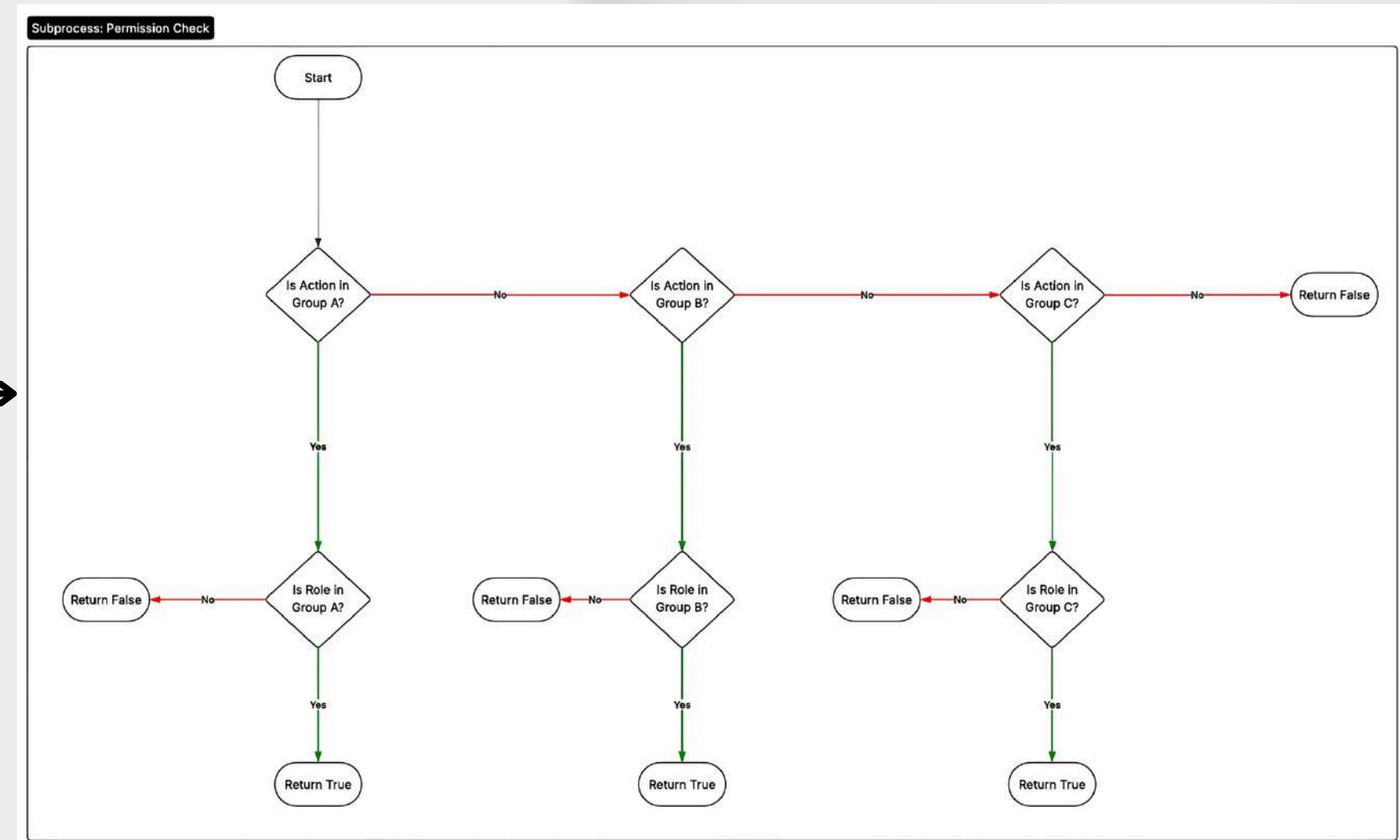




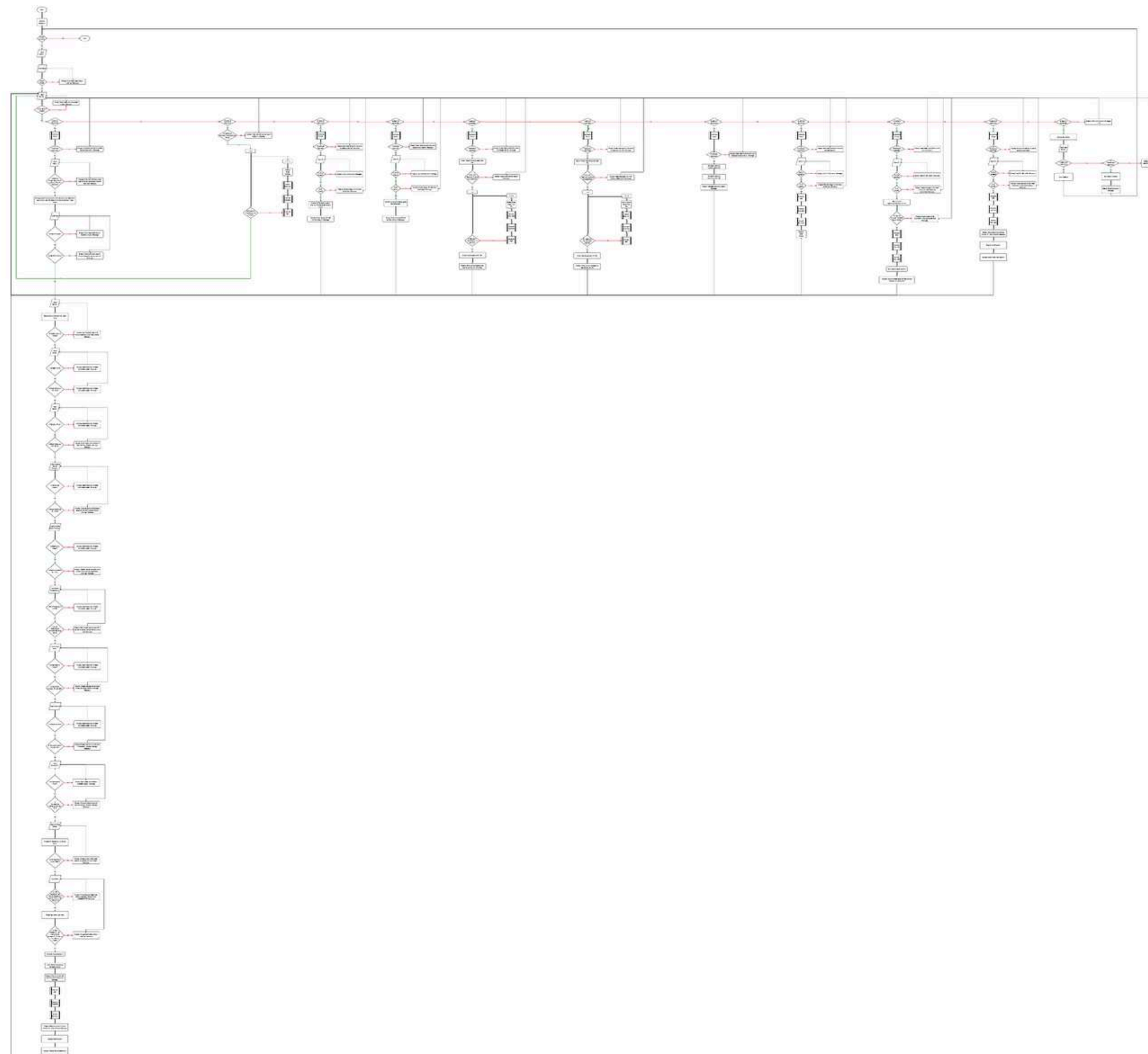
# Flowchart Function

## Code (Check Permission)

```
static int checkPermission(const char* role, const char* action) {    //memeriksa izin berdasarkan role
    if (strcmp(action, ACTION_DISPLAY_PATIENTS) == 0 ||
        strcmp(action, ACTION_VIEW_PATIENT_STATUS) == 0 ||
        strcmp(action, ACTION_EXPORT_ONE_PATIENT) == 0 ||
        strcmp(action, ACTION_FIND_PATIENT) == 0) {
        //admin, dokter, dan perawat yang memiliki izin
        if (strcmp(role, "admin") == 0 || strcmp(role, "dokter") == 0 || strcmp(role, "perawat") == 0) {
            return 1;    //izin diberikan
        }
    }
    if (strcmp(action, ACTION_ADD_PATIENT) == 0 ||
        strcmp(action, ACTION_SAVE_FILE) == 0 ||
        strcmp(action, ACTION_NEW_DATABASE) == 0 ||
        strcmp(action, ACTION_SAVE_CSV) == 0) {
        //hanya admin dan dokter yang memiliki izin
        if (strcmp(role, "admin") == 0 || strcmp(role, "dokter") == 0) {
            return 1;    //izin diberikan
        }
    }
    if (strcmp(action, ACTION_DELETE_PATIENT) == 0 ||
        strcmp(action, ACTION_UPDATE_PATIENT) == 0) {
        //hanya admin yang memiliki izin
        if (strcmp(role, "admin") == 0) {
            return 1;    //izin diberikan
        }
    }
    return 0;    //izin ditolak
}
```



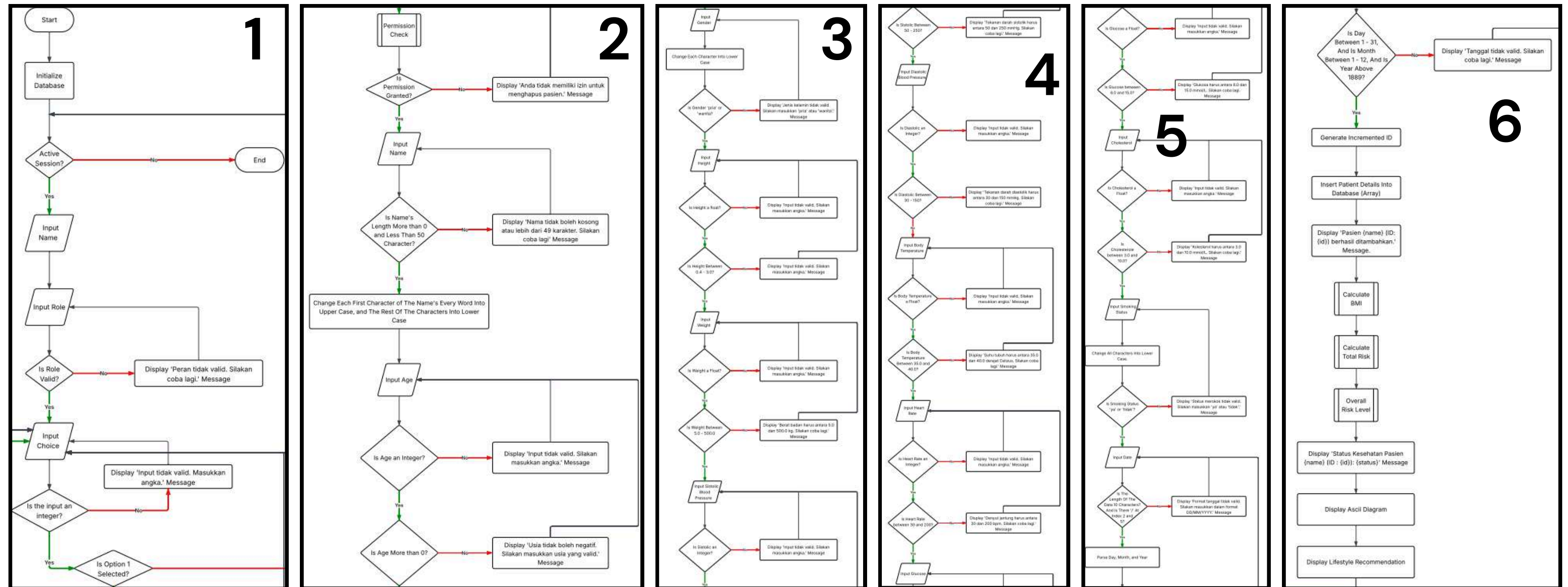
# Flowchart Main





# Flowchart Main

Case/option 1

Pilihan menambah  
data pasien

# Flowchart Main

**Case/option 1**

## Pilihan menambah data pasien

```
case 1: { //pilihan tambah pasien
    if(checkPermission(role, ACTION_ADD_PATIENT)){
        createAndAddPatient(); //memanggil fungsi untuk tambah pasien
    } else {
        printf("Anda tidak memiliki izin untuk menambah pasien.\n");
    }
    break;}
}
```

```
int addPatient(struct Patient* p) {
    if(patientCount >= MAX) { // Cek bila database penuh
        printf("Database penuh, tidak bisa menambah pasien baru.\n"); // Beri tahu user
        return -1; // Kembalikan kode gagal
    }
    p->id = nextID; // Tetapkan ID baru untuk pasien
    if(searchPatient(p->id) != NULL) { // Cek jika ID sudah digunakan
        int temp_id = p->id; // Mulai dari ID sekarang
        while(searchPatient(temp_id) != NULL) { // Cari ID kosong terdekat
            temp_id++; // Tambah ID sementara
        }
        p->id = temp_id; // Set ID baru
        nextID = temp_id; // Update nextID
    }
    patients[patientCount++] = *p; // Tambahkan pasien ke array
    nextID++; // Increment nextID
    printf("Pasien %s (ID: %d) berhasil ditambahkan.\n", p->name, p->id);
    return 0; // Kode sukses
}
```



# Flowchart Main

Case/option 2

Pilihan menampilkan  
data pasien

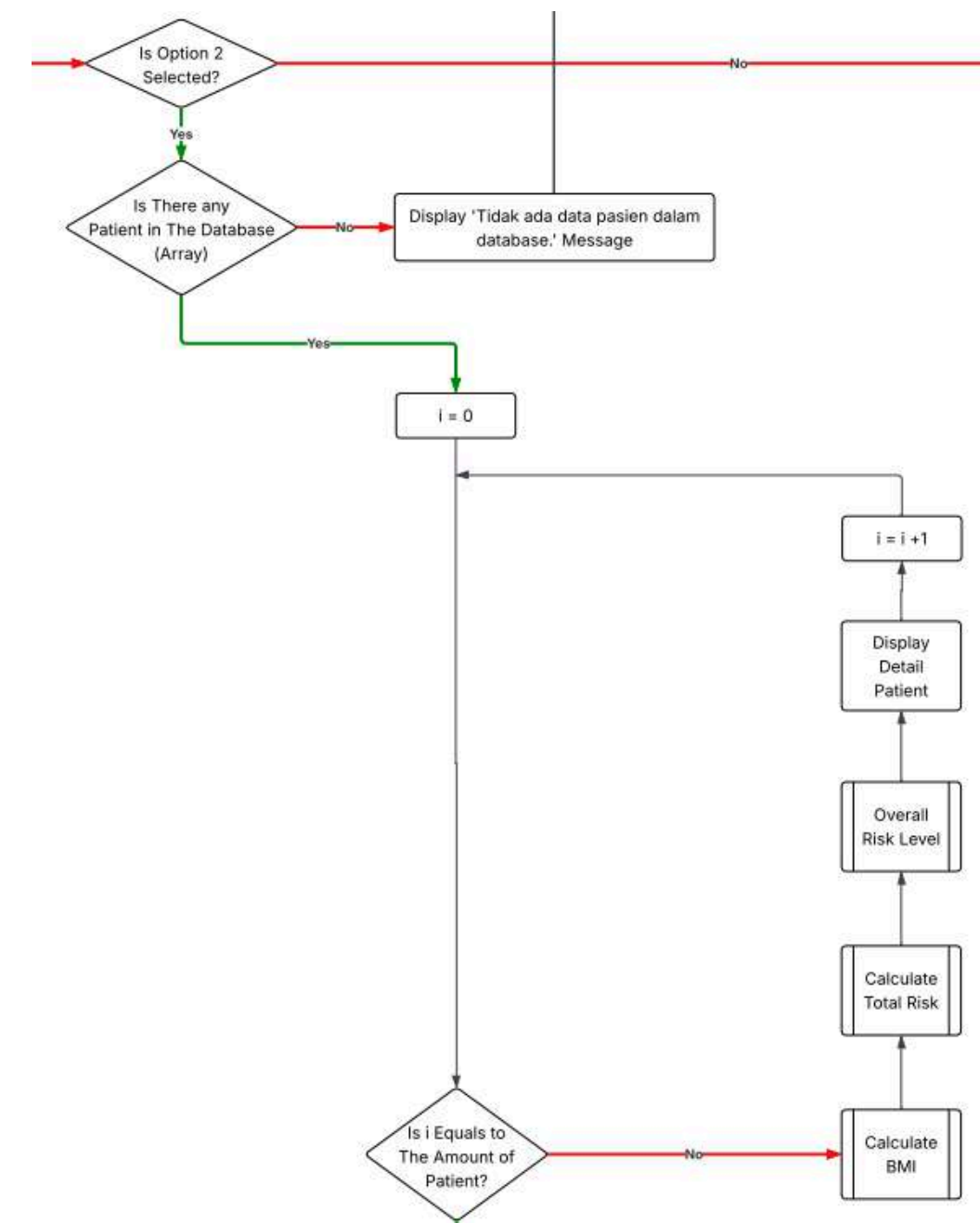
```
case 2: {                //pilihan tam
    displayPatients();
    break;}

```

```
void displayPatients(void) {
    if (patientCount == 0) {                // Cek apakah tidak ada pasien
        printf("Tidak ada data pasien dalam database.\n"); // Info kosong
        return;                            // Kembalikan tanpa menampilkan pasien
    }
    printf("\n=== Data Pasien ===\n");      // Header tampilan data pasien
    for (int i = 0; i < patientCount; i++) { // Loop semua pasien
        struct Patient* p = &patients[i]; // Ambil pointer ke pasien
        float bmi = calculateBMI(p->weight, p->height); // Hitung BMI
        int score = calculateTotalRiskScore(p); // Hitung skor risiko
        const char* status = overallRiskLevel(score); // Dapatkan status risiko

        // Cetak data pasien
        printf("\n%d. %s (ID: %d)\n", i + 1, p->name, p->id);
        printf("Usia          : %d tahun\n", p->age);
        printf("Jenis Kelamin   : %s\n", p->gender);
        printf("Tinggi Badan  : %.2f meter\n", p->height);
        printf("Berat Badan   : %.2f kg\n", p->weight);
        printf("BMI           : %.2f\n", bmi);
        printf("Glukosa       : %.2f mmol/L\n", p->glucose);
        printf("Tekanan Darah : %.0f/%.0f mmHg\n", p->systolic, p->diastolic);
        printf("Kolesterol    : %.2f mmol/L\n", p->cholesterol);
        printf("Status Merokok : %s\n", p->smokerStatus);
        printf("Status Risiko   : %s\n", status);
    }
}

```



# Flowchart Main

## Case/option 3

## Pilihan menghapus data pasien

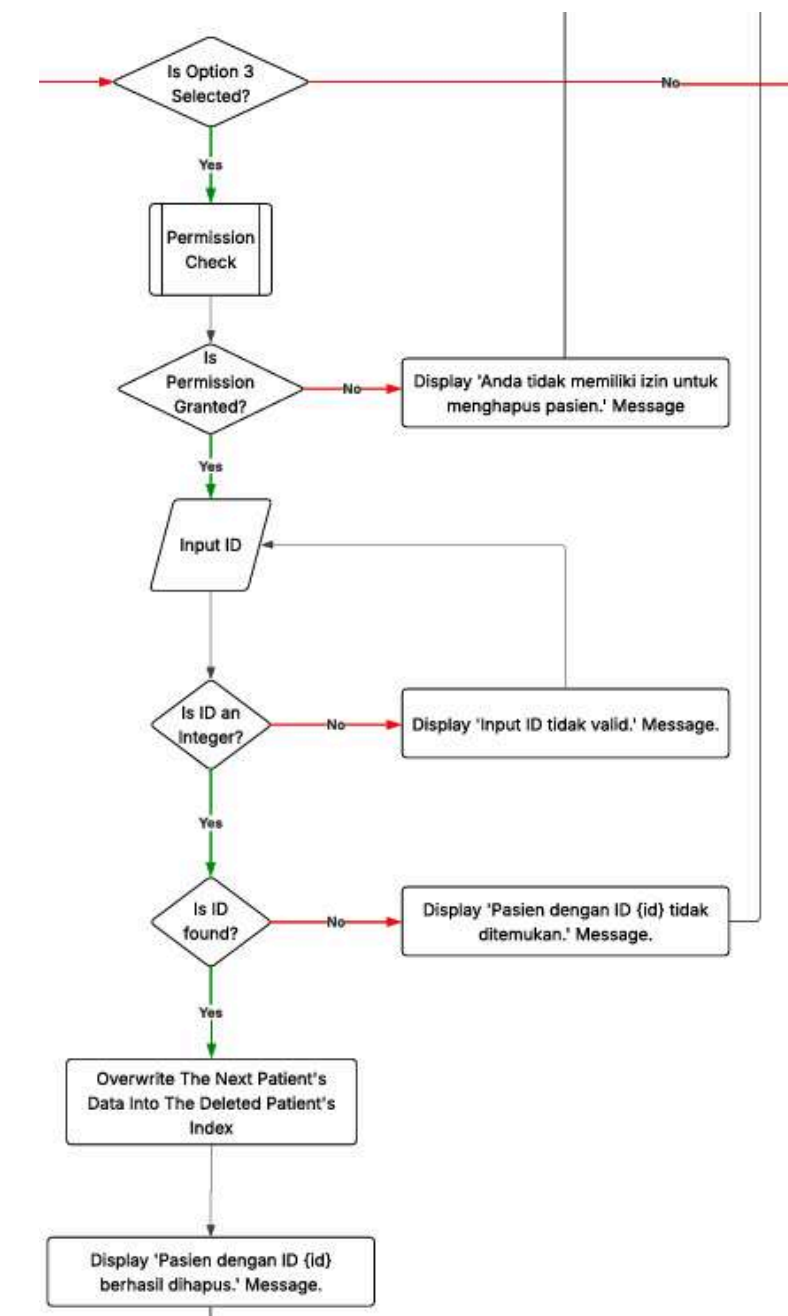
```

case 3: {
    if(checkPermission(role, ACTION_DELETE_PATIENT)){
        deletePatientData();
    }
    else {
        printf("Anda tidak memiliki izin untuk menghapus pasien.\n");
        break;
    }
}

int deletePatient(int id) {
    int found = -1; // Variabel untuk menyimpan indeks pasien yang ditemukan
    for (int i = 0; i < patientCount; i++) { // Loop cek seluruh pasien
        if (patients[i].id == id) { // Jika cocok ID
            found = i; // Simpan indeksnya jika cocok
            break; // Berhenti cari
        }
    }

    if (found != -1) {
        // Jika pasien ditemukan
        for (int j = found; j < patientCount - 1; j++) { // Geser array ke kiri untuk hapus pasien
            patients[j] = patients[j + 1]; // Pindahkan pasien selanjutnya ke indeks sebelumnya
        }
        patientCount--; // Kurangi jumlah pasien
        printf("Pasien dengan ID %d berhasil dihapus.\n", id); // Info sukses
        return 0; // Kode sukses
    }

    printf("Pasien dengan ID %d tidak ditemukan.\n", id); // Info pasien tidak ditemukan
    return -1; // Kode gagal
}
  
```





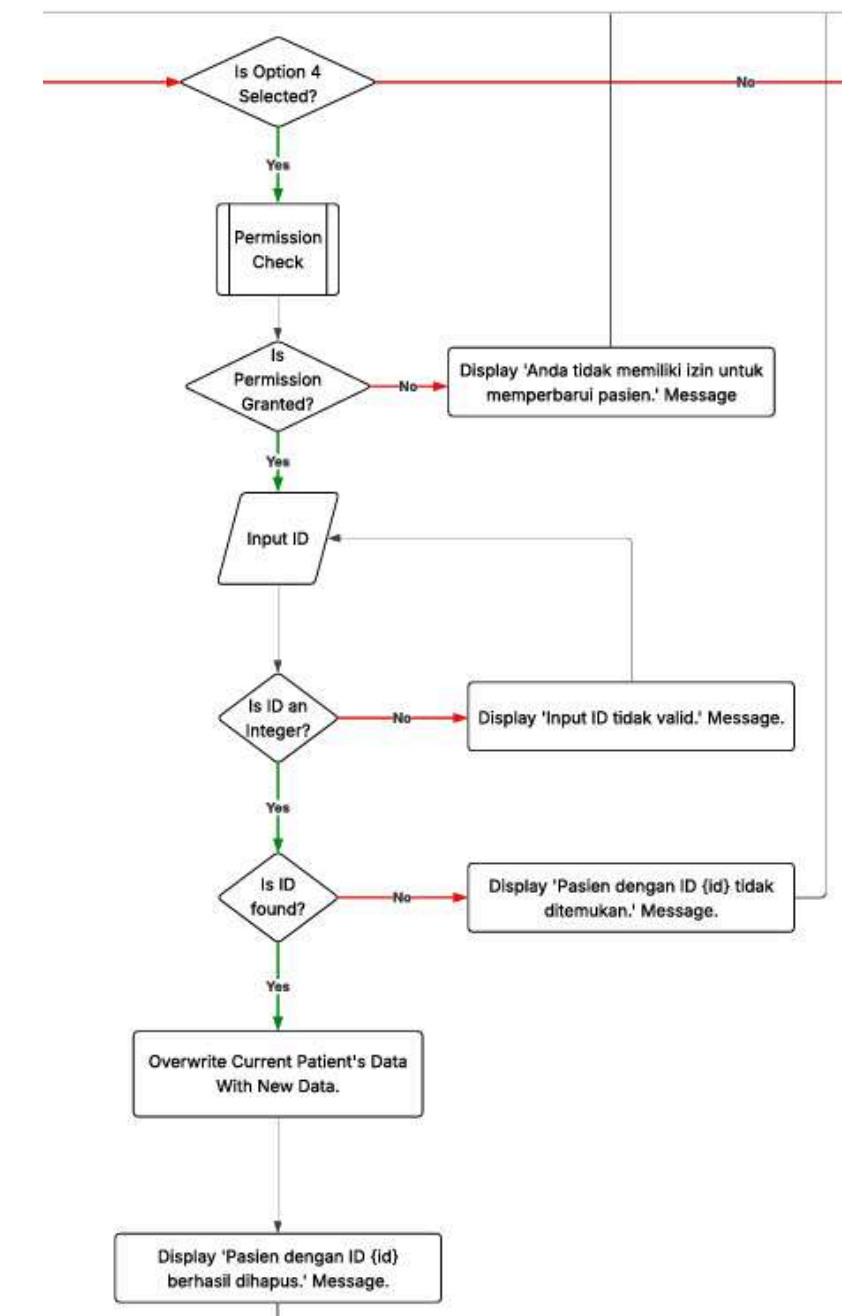
# Flowchart Main

## Case/option 4

## Pilihan memperbarui data pasien

```
case 4: {
    if(checkPermission(role, ACTION_UPDATE_PATIENT)){
        updatePatientData();
    }
    else {
        printf("Anda tidak memiliki izin untuk memperbarui pasien.\n");
        break;
    }
}
```

```
int updatePatient(int id, const struct Patient* p) {
    struct Patient* found = searchPatient(id); // Cari pasien berdasarkan ID
    if (!found) {
        printf("Pasien dengan ID %d tidak ditemukan.\n", id);
        return -1; // Kode gagal
    }
    *found = *p; // Perbarui data pasien
    printf("Data pasien berhasil diperbarui.\n");
    return 0; // Kode sukses
}
```



# Flowchart Main

Case/option 5

Pilihan menyimpan data  
pasien ke txt

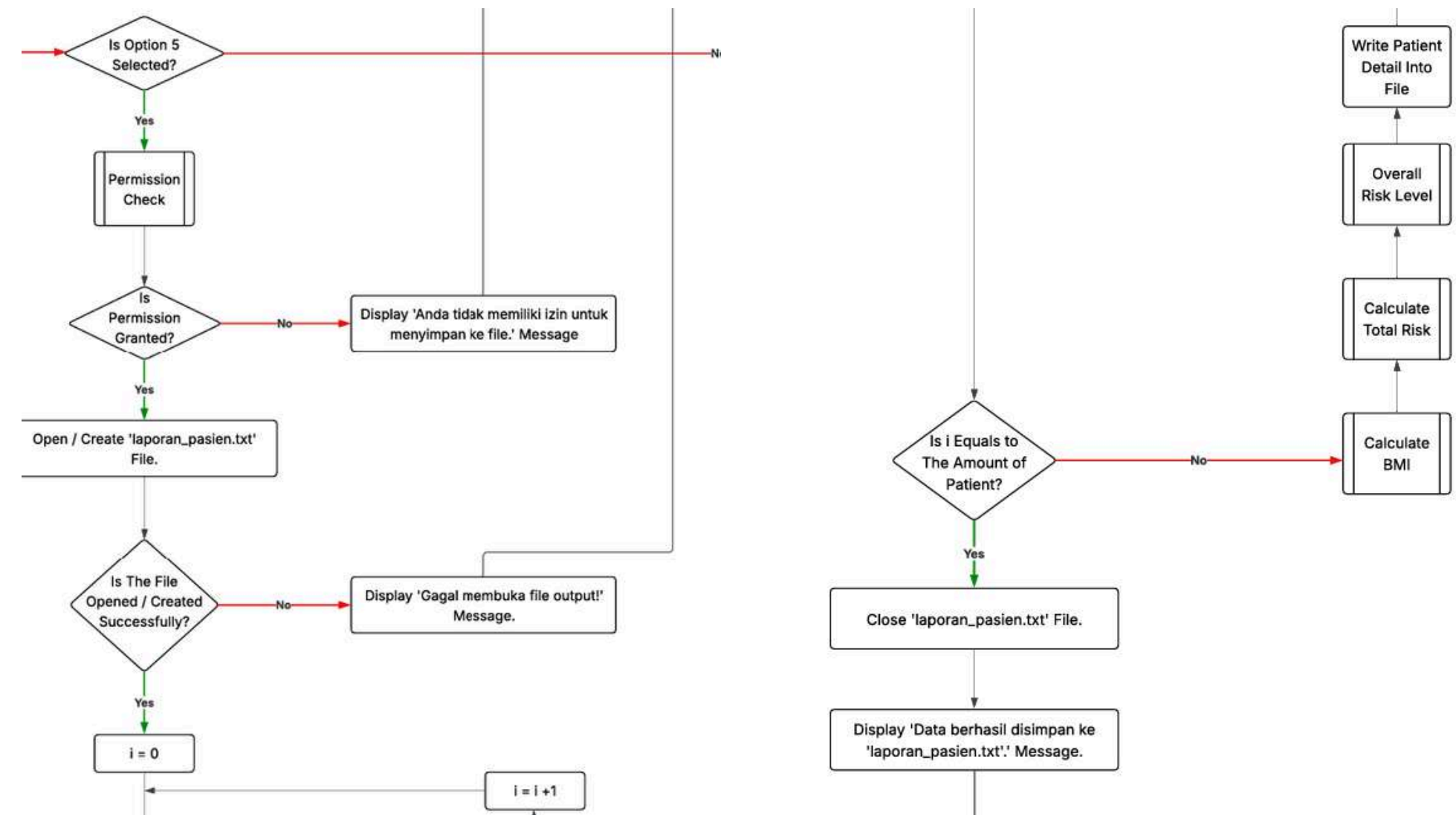
```
case 5: {
    if(checkPermission(role, ACTION_SAVE_FILE)){
        saveToFile();
    }
    else {
        printf("Anda tidak memiliki izin untuk menyimpan ke file.\n");
    }
    break;}

```

```
void saveToFile(void) {
    FILE* file = fopen("laporan_pasien.txt", "w"); // Buka file untuk tulis
    if (!file) {
        // Jika gagal buka file
        printf("Gagal membuka file output!\n"); // Info kegagalan
        return; // Kembali tanpa simpan
    }
    for (int i = 0; i < patientCount; i++) { // Loop semua pasien
        struct Patient* p = &patients[i]; // Pointer ke pasien
        float bmi = calculateBMI(p->weight, p->height); // Hitung BMI pasien
        int score = calculateTotalRiskScore(p); // Hitung skor risiko
        const char* status = overallRiskLevel(score); // Status risiko

        // Tulis laporan pasien ke file
        fprintf(file, "\n--- Laporan Evaluasi Pasien %s (ID: %d) ---\n", p->name, p->id);
        fprintf(file, "Usia : %d tahun\n", p->age);
        fprintf(file, "Jenis Kelamin : %s\n", p->gender);
        fprintf(file, "BMI : %.2f\n", bmi);
        fprintf(file, "Glukosa : %.2f mmol/L\n", p->glucose);
        fprintf(file, "Tekanan Darah : %.0f/%.0f mmHg\n", p->systolic, p->diastolic);
        fprintf(file, "Kolesterol : %.2f mmol/L\n", p->cholesterol);
        fprintf(file, "Status Merokok : %s\n", p->smokerStatus);
        fprintf(file, "Tanggal : %s\n", p->date);
        fprintf(file, "Status Risiko : %s\n", status);
    }
    fclose(file); // Tutup file
    printf("Data berhasil disimpan ke 'laporan_pasien.txt'.\n"); // Info simpan sukses
}

```





# Flowchart Main

Case/option 6

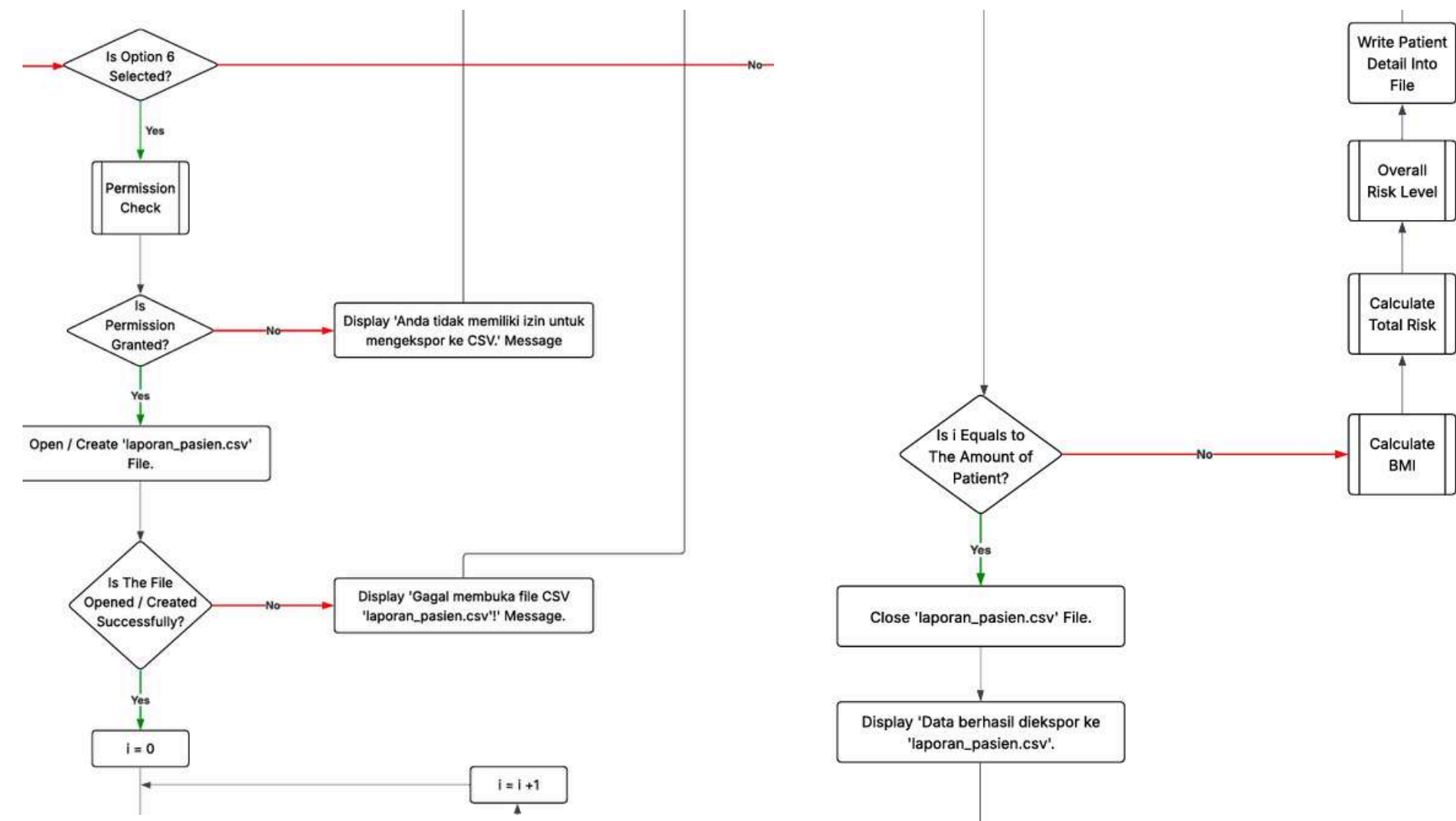
Pilihan menyimpan  
data pasien ke CSV

```

case 5: {
    if(checkPermission(role, ACTION_SAVE_FILE)){
        saveToFile();
    }
    else {
        printf("Anda tidak memiliki izin untuk menyimpan ke file.\n");
    }
    break;}

void exportToCSV(void) {
    FILE* file = fopen("laporan_pasien.csv", "w"); // Buka file CSV untuk tulis
    if (!file) {
        // Jika gagal buka file
        printf("Gagal membuka file CSV 'laporan_pasien.csv'!\n"); // Info kegagalan
        return; // Kembali tanpa ekspor
    }
    // Tulis header kolom CSV
    fprintf(file, "ID>Nama,Usia,Jenis Kelamin,Tinggi(m),Berat(kg),BMI,Glukosa(mmol/L),Kolesterol(mmol/L)\n");
    for (int i = 0; i < patientCount; i++) { // Loop semua pasien
        struct Patient* p = &patients[i]; // Pointer ke pasien
        float bmi = calculateBMI(p->weight, p->height); // Hitung BMI
        int score = calculateTotalRiskScore(p); // Hitung skor risiko
        const char* status = overallRiskLevel(score); // Status risiko

        // Tulis data pasien dalam format CSV
        fprintf(file, "%d,\"%s\",%d,%s,%.2f,%.2f,%.2f,%.2f,%.0f,%.0f,%.1f,%d,%s,%s,%s\n",
            p->id, p->name, p->age, p->gender,
            p->height, p->weight, bmi, p->glucose,
            p->cholesterol, p->systolic, p->diastolic,
            p->temperature, p->heartRate,
            p->smokerStatus, p->date, status);
    }
    fclose(file); // Tutup file CSV
    printf("Data berhasil diekspor ke 'laporan_pasien.csv'\n"); // Info ekspor sukses
}
  
```

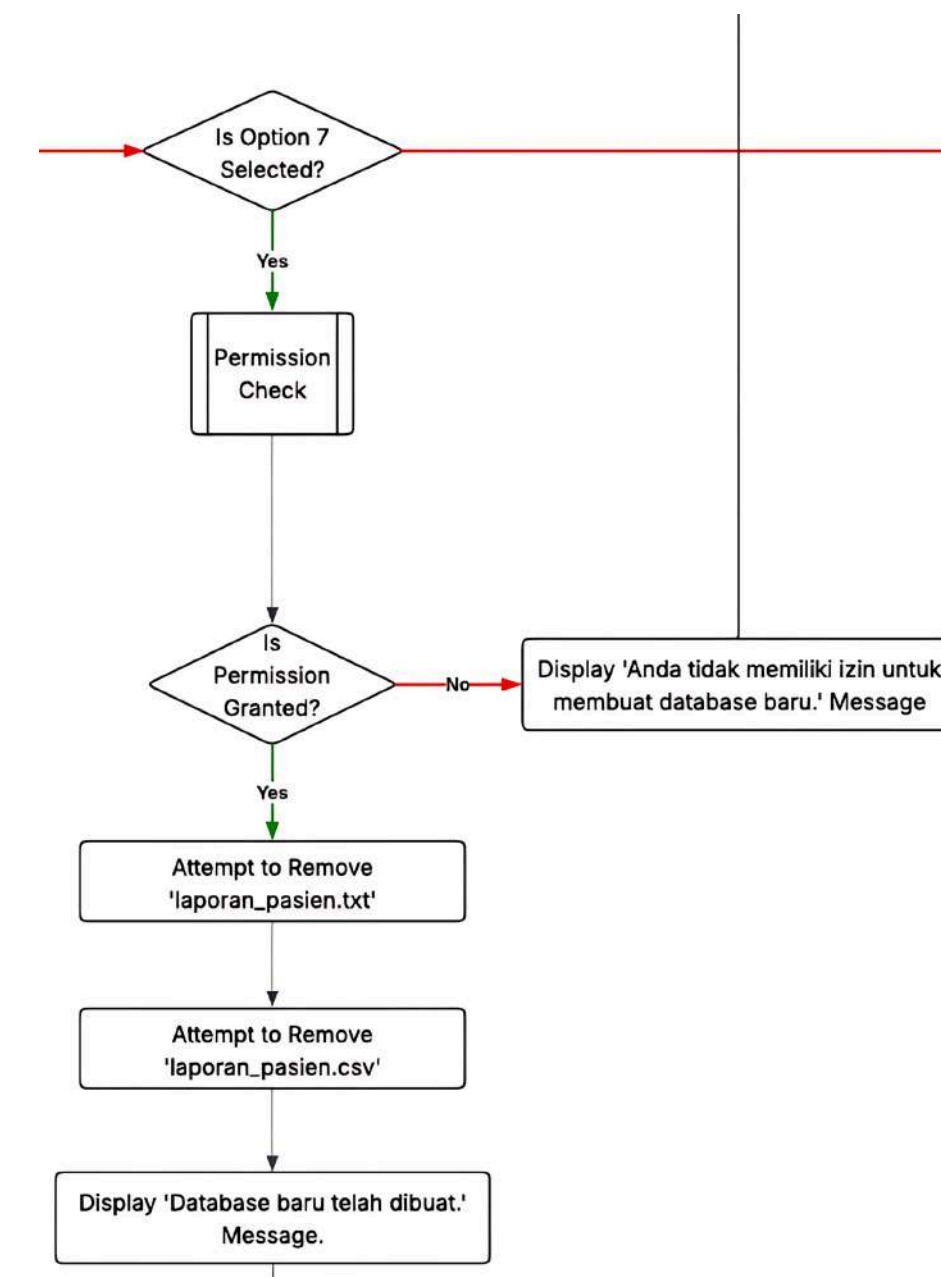


# Flowchart Main

## Case/option 7

Pilihan membuat  
database pasien baru

```
case 7: {           //pilihan membuat database baru
    if(checkPermission(role, ACTION_NEW_DATABASE)){
        newDatabase();}           //memanggil fungsi untuk membuat database baru
    else {
        printf("Anda tidak memiliki izin untuk membuat database baru.\n");
    }
    break;}
void newDatabase() {
    n = 0;
    printf("Database pasien baru telah dibuat (data lama dihapus).\n");
}
```





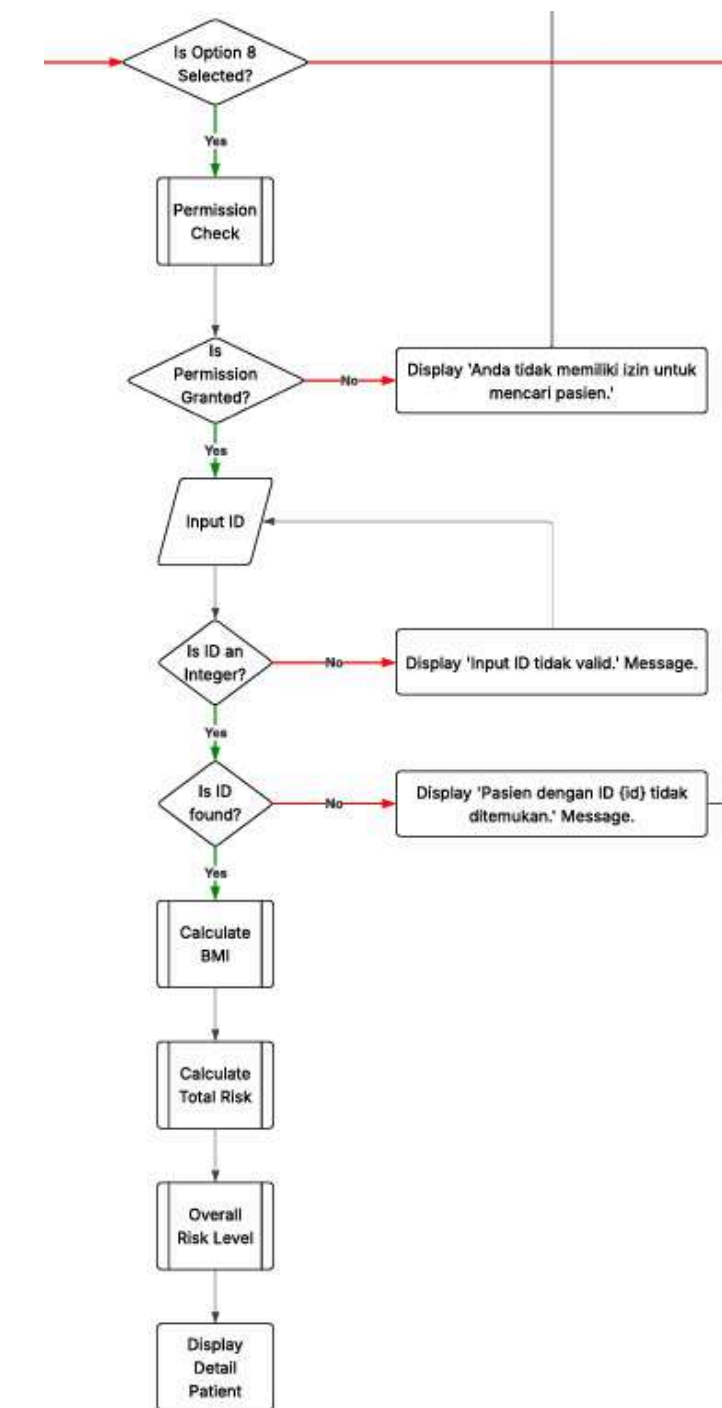
# Flowchart Main

Case/option 8

Pilihan mencari pasien

```

case 8: {           //pilihan cari pasien
    if(checkPermission(role, ACTION_FIND_PATIENT)){
        searchPatientData();}           //memanggil fungsi untuk cari pasien
    else {
        printf("Anda tidak memiliki izin untuk mencari pasien.\n");
    }
    break;}
void searchPatientData(void) {
    int id;           // Variabel untuk menampung ID pasien yang dicari
    printf("Masukkan ID Pasien yang ingin dicari: "); // Minta input ID pasien
    if (scanf("%d", &id) != 1) {           // Baca dan validasi input ID pasien
        clearInputBuffer();           // Bersihkan buffer jika input tidak valid
        printf("Input ID tidak valid.\n"); // Informasi kesalahan input
        return;           // Hentikan fungsi jika input salah
    }
    clearInputBuffer();           // Bersihkan buffer setelah input valid
    searchAndDisplayPatientDetails(id); // Cari pasien dan tampilkan detailnya
}
  
```



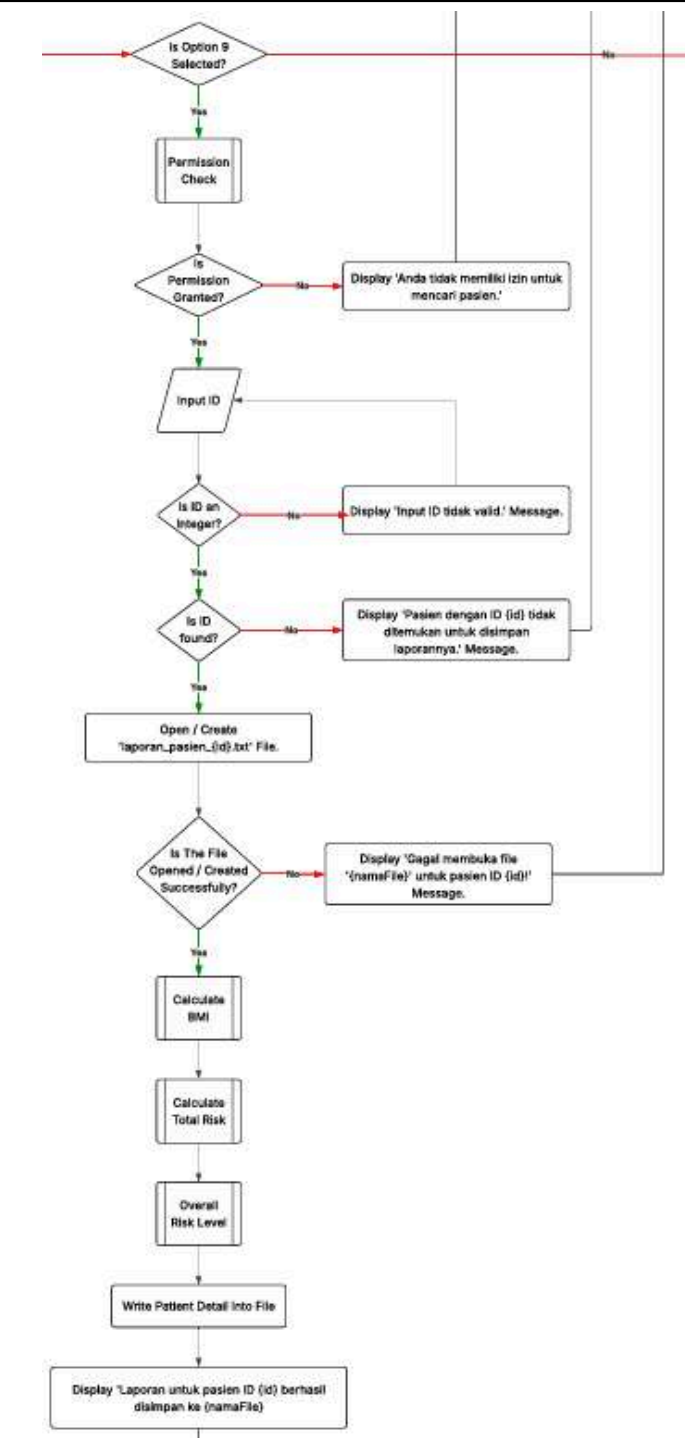
# Flowchart Main

## Case/option 9

## Pilihan simpan laporan satu pasien

```

case 9: {           //pilihan simpan laporan satu pasien
    if(checkPermission(role, ACTION_EXPORT_ONE_PATIENT)){
        saveSinglePatientData(); //memanggil fungsi untuk simpan laporan satu pasien
    } else {
        printf("Anda tidak memiliki izin untuk menyimpan laporan satu pasien.\n");
    }
    break;}
void saveSinglePatientData(void) {
    int id;           // Variabel untuk menampung ID pasien yang laporannya akan disimpan
    printf("Masukkan ID Pasien yang ingin disimpan laporannya: "); // Minta input ID pasien
    if (scanf("%d", &id) != 1) { // Baca input dan validasi
        clearInputBuffer(); // Bersihkan buffer jika input salah
        printf("Input ID tidak valid.\n"); // Informasi kesalahan input
        return; // Hentikan fungsi jika salah input
    }
    clearInputBuffer(); // Bersihkan buffer input
    if (saveSinglePatient(id) == 0) { // Simpan laporan pasien jika berhasil
        printf("Laporan pasien dengan ID %d berhasil disimpan.\n", id); // Konfirmasi keberhasilan penyimpanan
    }
}
  
```





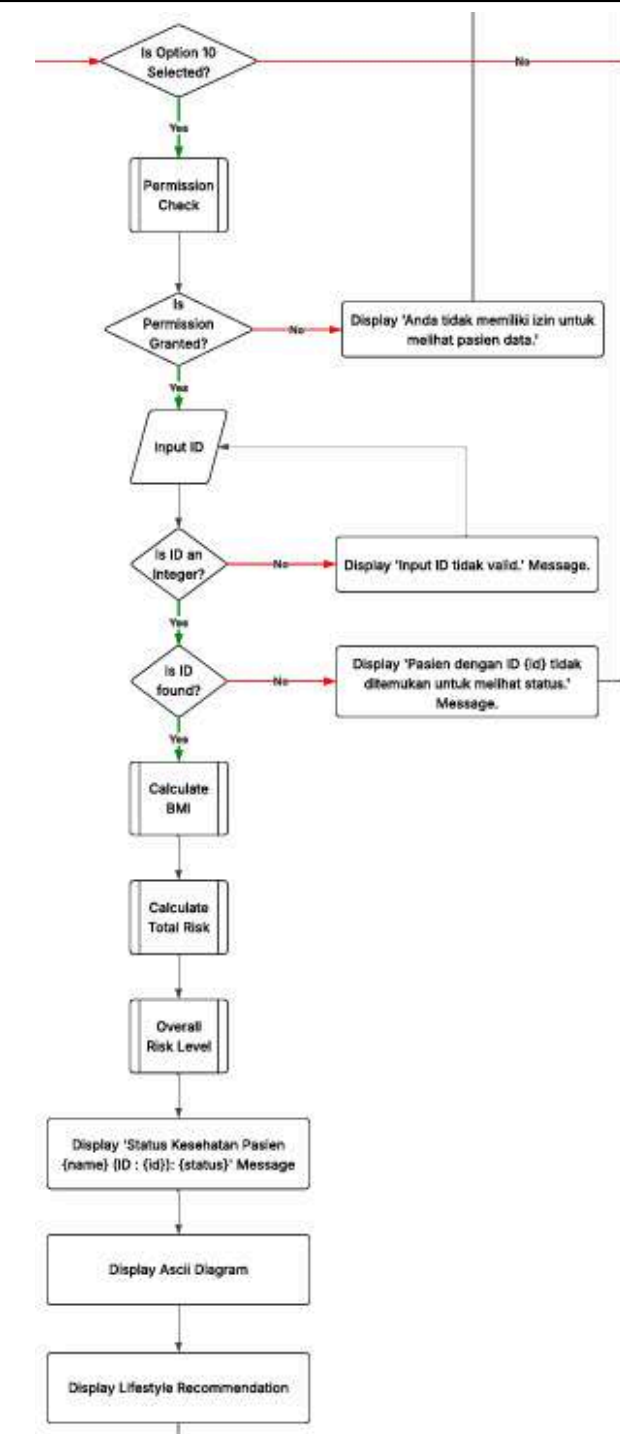
# Flowchart Main

Case/option 10

Pilihan lihat status  
pasien

```
case 10: {           //pilihan lihat status pasien
    if(checkPermission(role, ACTION_VIEW_PATIENT_STATUS)){
        checkPatientStatusData();} //memanggil fungsi untuk lihat status pasien
    else {
        printf("Anda tidak memiliki izin untuk melihat status pasien.\n");
    }
    break;
}

void checkPatientStatusData(void) {
    int id;           // Variabel untuk menampung ID pasien yang statusnya ingin dicek
    printf("Masukkan ID Pasien yang ingin diperiksa statusnya: "); // Minta input ID pasien
    if (scanf("%d", &id) != 1) { // Baca dan validasi input ID pasien
        clearInputBuffer(); // Bersihkan buffer jika input salah
        printf("Input ID tidak valid.\n"); // Pesan error input tidak valid
        return; // Hentikan fungsi jika input salah
    }
    clearInputBuffer(); // Bersihkan buffer input
    viewPatientStatus(id); // Tampilkan status pasien berdasarkan ID yang dimasukkan
}
```



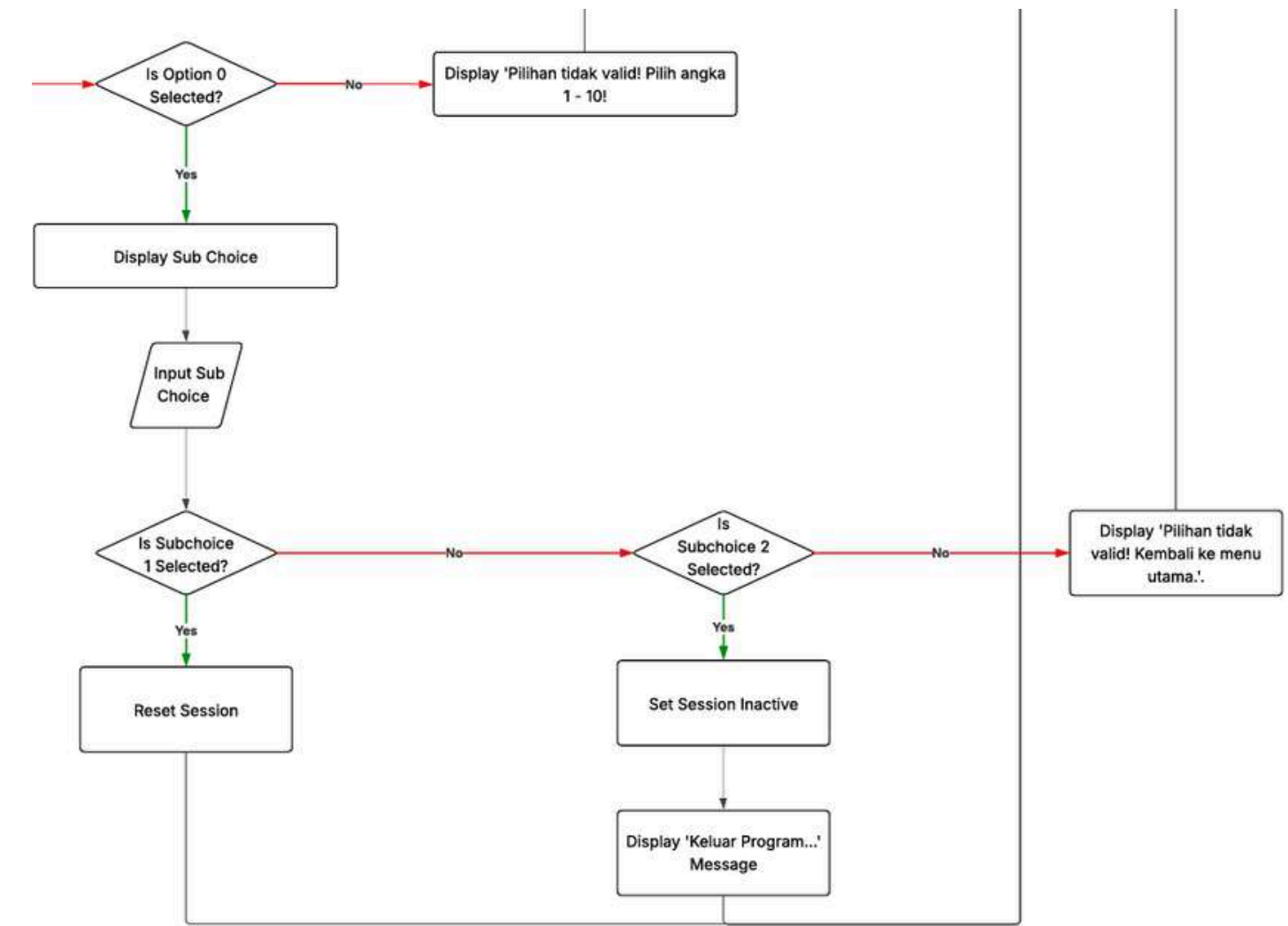
# Flowchart Main

Case/option 0

Pilihan keluar dari menu

```
case 0: {           //pilihan keluar dari menu
    int subChoice; //variabel pilihan sub-menu dari pilihan keluar
    printf("\nApakah Anda ingin:\n");           //pilihan keluar atau ganti role
    printf("1. Ganti Nama dan Role\n");         //opsi ganti nama dan role
    printf("2. Keluar Program\n");              //opsi keluar program
    printf("Pilihan Anda: ");                   //input dari user
    scanf("%d", &subChoice);                    //membaca pilihan sub-menu yg dipilih user

    //proses pilihan sub-menu
    if (subChoice == 1) {
        sessionEnded = 1;                      //mengakhiri sesi, bisa ganti role dan nama
    } else if (subChoice == 2) {
        printf("Keluar program...\n");
        printf("=====\n");
        sessionEnded = 1;                      //mengakhiri sesi dan program
        exitProgram = 1;
    } else {
        printf("Pilihan tidak valid! Kembali ke menu utama.\n"); //muncul klo yg dipilih gada di sub-menu
    }
    break;}
}
```





# Program Running

## INPUT

```
=====
Pilihan Anda: 1
Nama           : cia
Usia (tahun)   : 19
Jenis Kelamin (pria/wanita): wanita
Tinggi (meter) : 1.6
Berat Badan (kg) : 53
Tekanan Darah Sistolik (mmHg): 118
Tekanan Darah Diastolik (mmHg): 84
Suhu Tubuh (Celsius) : 36
Denyut Jantung (bpm) : 78
Glukosa (mmol/L) : 6.5
Kolesterol (mmol/L) : 4
Status Merokok (ya/tidak) : tidak
Tanggal (DD/MM/YYYY) : 09/06/2025
Pasien Cia (ID: 1) berhasil ditambahkan.
```

## OUTPUT

```
Status Kesehatan Pasien Cia (ID: 1): Sehat
```

```
Diagram Risiko (0-6):
```

```
[ -*----- ]
```

```
Rekomendasi Gaya Hidup:
```

```
- Pertahankan pola hidup sehat.
```

```
=====
```

# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 2
```

## OUTPUT

```
=== Data Pasien ===

1. Cia (ID: 1)
Usia           : 19 tahun
Jenis Kelamin  : wanita
Tinggi Badan   : 1.60 meter
Berat Badan    : 53.00 kg
BMI            : 20.70
Glukosa        : 6.50 mmol/L
Tekanan Darah  : 118/84 mmHg
Kolesterol     : 4.00 mmol/L
Status Merokok : tidak
Status Risiko   : Sehat
=====
```



# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
```

Pilihan Anda: 3

## OUTPUT

```
Masukkan ID Pasien yang ingin dihapus: 1
Pasien dengan ID 1 berhasil dihapus.
```

```
=====
Pilihan Anda: 2
Tidak ada data pasien dalam database.
=====
```

```
=====
Pilihan Anda: 4
Masukkan ID Pasien yang ingin diperbarui: 1
Pasien dengan ID 1 tidak ditemukan. Tidak dapat memperbarui.
=====
```

# Program Running

## INPUT

```
=====
Pilihan Anda: 4
Masukkan ID Pasien yang ingin diperbarui: 2
--- Memasukkan detail baru untuk pasien ID 2 ---
Catatan: ID Pasien tidak dapat diubah.
Nama                               : jack
Usia (tahun)                       : 20
Jenis Kelamin (pria/wanita): pria
Tinggi (meter)                    : 1.89
Berat Badan (kg)                   : 100
Tekanan Darah Sistolik (mmHg): 170
Tekanan Darah Diastolik (mmHg): 120
Suhu Tubuh (Celsius)              : 38
Denyut Jantung (bpm)              : 90
Glukosa (mmol/L)                  : 13
Kolesterol (mmol/L)               : 9
Status Merokok (ya/tidak) : ya
Tanggal (DD/MM/YYYY)              : 09/06/2025
Data pasien berhasil diperbarui.
```

## OUTPUT

```
=====
Status Kesehatan Pasien Bel (ID: 2): Perlu Perhatian

Diagram Risiko (0-6):
[ ----*-- ]

Rekomendasi Gaya Hidup:
- Kurangi konsumsi gula dan garam.
- Rutin olahraga ringan.
=====

Status Kesehatan Pasien Jack (ID: 2): Risiko Tinggi

Diagram Risiko (0-6):
[ -----* ]

Rekomendasi Gaya Hidup:
- Konsultasi rutin dengan dokter.
- Perubahan gaya hidup menyeluruh diperlukan.
=====
```



# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 5
Data berhasil disimpan ke 'laporan_pasien.txt'.
=====
```

## OUTPUT

```
EXPLORER  ...  C main.c  laporan_pasien.txt U X
v FINPRO_PROGC_GROUP7
  C database.c
  C database.h
  laporan_pasien.txt U
  C main.c
  medicalChecku... M
  C patient.c
  C patient.h
  C progress1.c
  progress1.exe
  C utils.c
  C utils.h

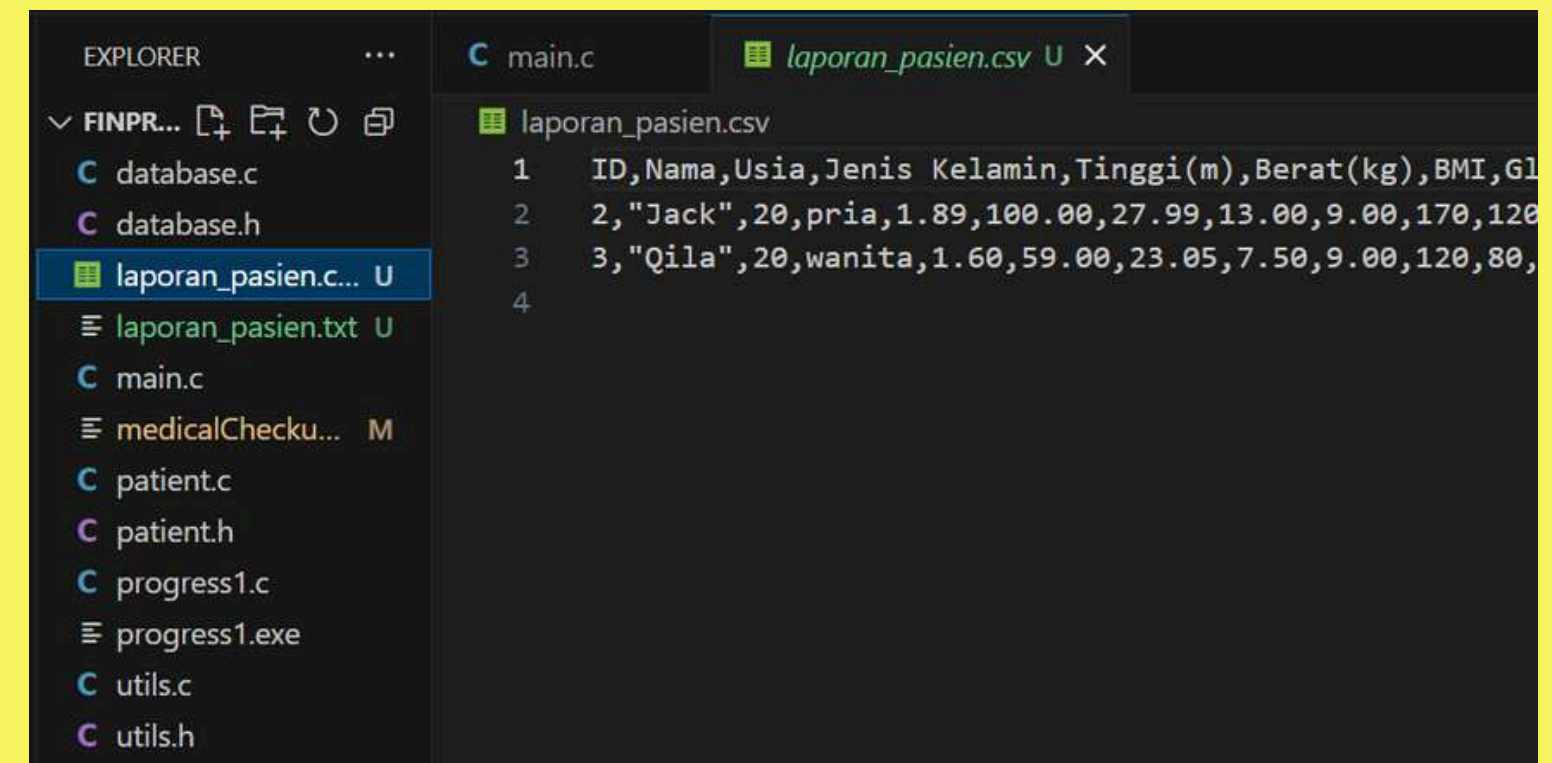
laporan_pasien.txt
1
2 --- Laporan Evaluasi Pasien Jack (ID: 2) ---
3 Usia : 20 tahun
4 Jenis Kelamin : pria
5 BMI : 27.99
6 Glukosa : 13.00 mmol/L
7 Tekanan Darah : 170/120 mmHg
8 Kolesterol : 9.00 mmol/L
9 Status Merokok : ya
10 Tanggal : 09/06/2025
11 Status Risiko : Risiko Tinggi
12
13 --- Laporan Evaluasi Pasien Qila (ID: 3) ---
14 Usia : 20 tahun
15 Jenis Kelamin : wanita
16 BMI : 23.05
17 Glukosa : 7.50 mmol/L
18 Tekanan Darah : 120/80 mmHg
19 Kolesterol : 9.00 mmol/L
20 Status Merokok : ya
21 Tanggal : 09/06/2025
22 Status Risiko : Perlu Perhatian
23
```

# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 6
Data berhasil diekspor ke 'laporan_pasien.csv'
=====
```

## OUTPUT



```
EXPLORER
FINPR...
  database.c
  database.h
  laporan_pasien.c... U
  laporan_pasien.txt U
  main.c
  medicalChecku... M
  patient.c
  patient.h
  progress1.c
  progress1.exe
  utils.c
  utils.h

C main.c
laporan_pasien.csv U X

laporan_pasien.csv
1 ID,Nama,Usia,Jenis Kelamin,Tinggi(m),Berat(kg),BMI,G1
2 2,"Jack",20,pria,1.89,100.00,27.99,13.00,9.00,170,120
3 3,"Qila",20,wanita,1.60,59.00,23.05,7.50,9.00,120,80,
```

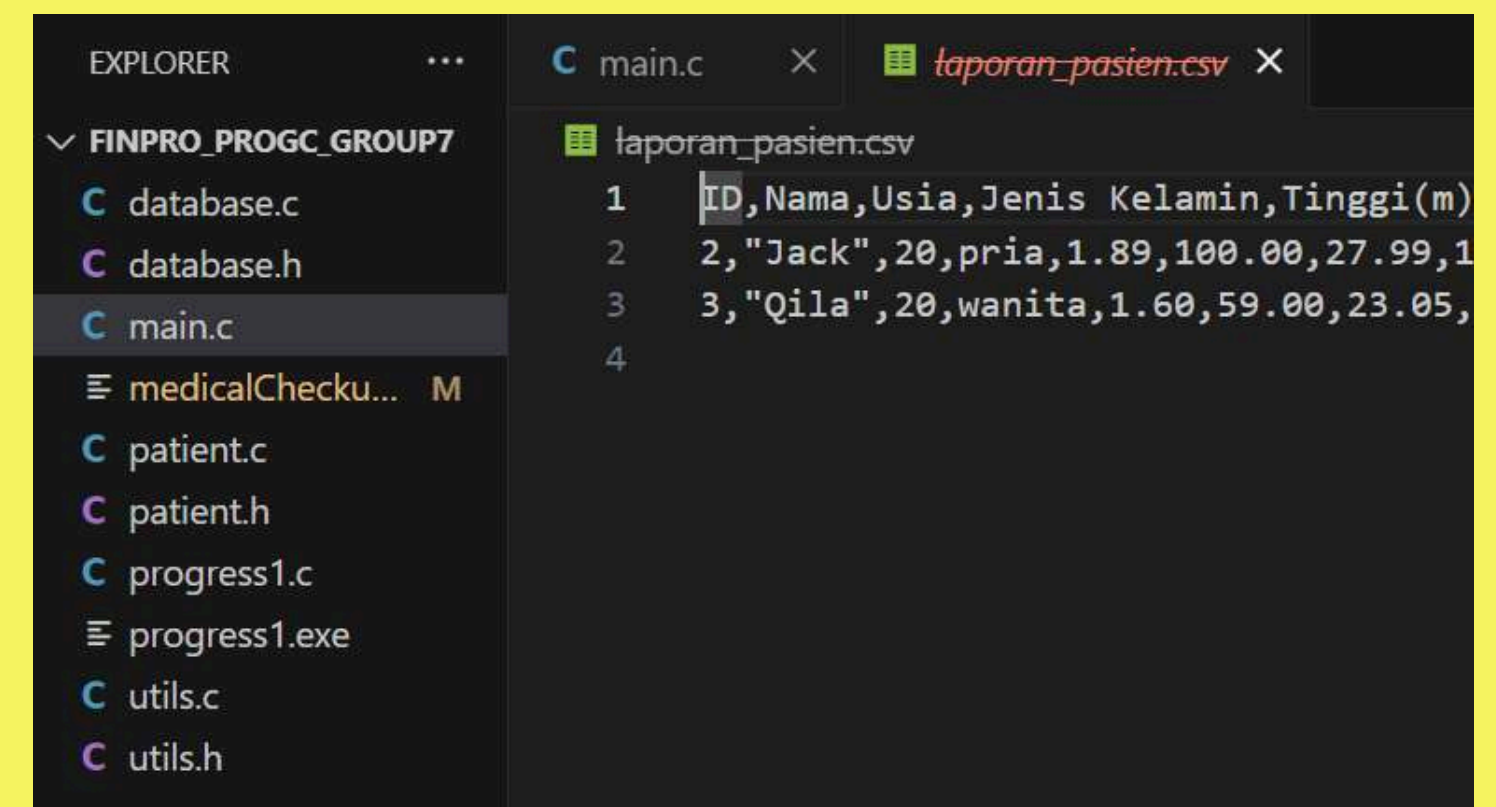


# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 7
Database baru telah dibuat.
```

## OUTPUT



The screenshot shows a code editor with a file explorer on the left and a CSV file on the right. The file explorer lists files in a project named 'FINPRO\_PROGC\_GROUP7', including 'database.c', 'database.h', 'main.c', 'medicalChecku...', 'patient.c', 'patient.h', 'progress1.c', 'progress1.exe', 'utils.c', and 'utils.h'. The CSV file, named 'laporan\_pasien.csv', contains the following data:

ID	Nama	Usia	Jenis Kelamin	Tinggi(m)
2	Jack	20	pria	1.89
3	Qila	20	wanita	1.60

# Program Running

## INPUT

```
=====
Pilihan Anda: 8
Masukkan ID Pasien yang ingin dicari: 2
Pasien dengan ID 2 tidak ditemukan.
```

```
Status Kesehatan Pasien Cici (ID: 1): Sehat
```

```
Diagram Risiko (0-6):
```

```
[ *----- ]
```

```
Rekomendasi Gaya Hidup:
```

```
- Pertahankan pola hidup sehat.
```

## OUTPUT

```
=====
Pilihan Anda: 8
Masukkan ID Pasien yang ingin dicari: 1

--- Detail Pasien ID: 1 ---
Nama           : Cici
Usia           : 19 tahun
Jenis Kelamin  : wanita
Tinggi Badan   : 1.60 meter
Berat Badan    : 54.00 kg
BMI            : 21.09
Tekanan Darah  : 120/78 mmHg
Suhu Tubuh     : 36.0 C
Denyut Jantung : 70 bpm
Kadar Glukosa  : 6.50 mmol/L
Kolesterol     : 4.00 mmol/L
Status Merokok : tidak
Tanggal Pemeriksaan: 09/06/2025
Status Risiko   : Sehat
Skor Risiko Total : 0
=====
```



# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 9
Masukkan ID Pasien yang ingin disimpan laporannya: 1
Laporan untuk pasien ID 1 berhasil disimpan ke 'laporan_pasien_1.txt'
Laporan pasien dengan ID 1 berhasil disimpan.
=====
```

## OUTPUT

```
EXPLORER  ...  C main.c  laporan_pasien_1.txt U X
└─ FINPRO_PROGC_GROUP7
   ├─ database.c
   ├─ database.h
   └─ laporan_pasien_1.txt U
└─ main.c
   ├─ medicalCheckup.exe M
   ├─ patient.c
   ├─ patient.h
   ├─ progress1.c
   ├─ progress1.exe
   ├─ utils.c
   └─ utils.h

laporan_pasien_1.txt
1  === Laporan Evaluasi Pasien Cici (ID: 1) ===
2  Tanggal Pemeriksaan: 09/06/2025
3  Usia : 19 tahun
4  Jenis Kelamin : wanita
5  Tinggi Badan : 1.60 meter
6  Berat Badan : 54.00 kg
7  BMI : 21.09
8  Glukosa : 6.50 mmol/L
9  Tekanan Darah : 120/78 mmHg
10 Suhu Tubuh : 36.0 C
11 Denyut Jantung : 70 bpm
12 Kolesterol : 4.00 mmol/L
13 Status Merokok : tidak
14 Status Risiko : Sehat
15 Skor Risiko Total : 0
16 =====
17
```

# Program Running

## INPUT

```
=====
Menu (User: admin):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
2. Tampilkan Data Pasien
3. Hapus Data Pasien
3. Hapus Data Pasien
4. Perbarui Data Pasien
5. Simpan ke File
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
Pilihan Anda: 10
```

## OUTPUT

```
=====
Pilihan Anda: 10
Masukkan ID Pasien yang ingin diperiksa statusnya: 1

--- Status untuk Pasien ID 1 ---

Status Kesehatan Pasien Cici (ID: 1): Sehat

Diagram Risiko (0-6):
[ *----- ]

Rekomendasi Gaya Hidup:
- Pertahankan pola hidup sehat.
-----
=====
```



# Program Running

## INPUT

```
=====
Pilihan Anda: 0

Apakah Anda ingin:
1. Ganti Nama dan Role
2. Keluar Program
Pilihan Anda: 1
Masukkan nama Anda: alice
Masukkan peran Anda (Admin/Dokter/Perawat): dokter
=====
```

```
=====
Pilihan Anda: 0

Apakah Anda ingin:
1. Ganti Nama dan Role
2. Keluar Program
Pilihan Anda: 1
Masukkan nama Anda: bel
Masukkan peran Anda (Admin/Dokter/Perawat): perawat
=====
```

## OUTPUT

```
=====
Menu (User: dokter):
1. Tambah Pasien Baru
2. Tampilkan Data Pasien
5. Simpan ke File
6. Expor ke CSV
7. Buat Database Baru
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
```

```
=====
Pilihan Anda: 0

Apakah Anda ingin:
1. Ganti Nama dan Role
2. Keluar Program
Pilihan Anda: 2
Keluar program...
=====
```

```
=====
Menu (User: perawat):
2. Tampilkan Data Pasien
8. Cari Pasien
9. Simpan Laporan Satu Pasien
10. Lihat Status Pasien
0. Keluar
=====
```

# CONCLUSION

## VITALS

Sistem Monitoring Pasien Berbasis C yang melakukan:

- Input data pasien, seperti tekanan darah, glukosa, suhu, dll.
- Evaluasi risiko kesehatan berdasarkan skor dari parameter vital.
- Rekomendasi gaya hidup sesuai tingkat risiko.
- Penyimpanan & pengelolaan data ke dalam file .txt dan .csv.

## Future Development

Penambahan Fitur Medis Lanjutan: Misalnya integrasi rekam medis lengkap, riwayat pemeriksaan sebelumnya, dan grafik tren kesehatan pasien.

## Limitations

- Tidak Ada Penyimpanan Database Permanen: Tidak terhubung ke database nyata (SQL/NoSQL), hanya berbasis file .txt dan .csv.
- Tidak Ada Validasi Login atau Keamanan Data Lanjutan: Program hanya membedakan peran (admin/dokter/perawat) tanpa sistem autentikasi atau enkripsi.

## Modularity

Program terbagi menjadi beberapa file terpisah:

- **main.c**, navigasi menu & peran user
- **patient.c**, evaluasi risiko & skor kesehatan
- **database.c**, manajemen data pasien
- **utils.c**, input, validasi, dan fungsionalitas bantu
- **Setiap file .h** menyimpan deklarasi yang diperlukan.

## C Features

- if, else, switch untuk pengambilan keputusan
- for, while, do while untuk perulangan dan validasi input
- scanf untuk input dari user
- function untuk memisahkan tugas-tugas spesifik
- array untuk menyimpan data pasien
- pointer untuk akses/modifikasi data pasien
- struct untuk menyimpan data pasien secara terstruktur
- file handling (fopen, fprintf, fgets, fclose) untuk menyimpan & membaca file



# References

VITALITY HEALTH CHECK RANGES					
Metric	 Blood Glucose	 Blood pressure	 Cholesterol*	 Weight Status**	 Smoker status
In-range	< 7.8mmol/L	< 130/80mmHg	< 5mmol/L	Weight in relation to height 18.5-24.9 kg/m <sup>2</sup> OR Weight in relation to height 25-29.9 kg/m <sup>2</sup> AND waist circumference of <94 cm in men and <80 cm in women	Non-smoker for at least the past 12 months
Intermediate risk	7.8mmol/L ≤ blood glucose < 11mmol/L	130/80mmHg ≤ blood pressure < 160/100mmHg	5mmol/L ≤ total cholesterol < 7.5mmol/L	Weight in relation to height <18.5 kg/m <sup>2</sup> OR Weight in relation to height 25-29.9 kg/m <sup>2</sup> AND waist circumference of ≥ 94 cm in men and ≥ 80 cm in women	n/a
High risk	≥ 11mmol/L	≥ 160/100mmHg	≥ 7.5mmol/L	Weight in relation to height ≥ 30	Smoker



# References (IEEE Format)

- [1] A. J. Muhihi et al., “Public knowledge of risk factors and warning signs for cardiovascular disease among young and middle-aged adults in rural Tanzania,” *BMC Public Health*, vol. 20, no. 1, Nov. 2020, doi: <https://doi.org/10.1186/s12889-020-09956-z>.
- [2] Centers for Disease Control and Prevention (CDC), “Vital signs: awareness and treatment of uncontrolled hypertension among adults--United States, 2003-2010,” *MMWR. Morbidity and mortality weekly report*, vol. 61, pp. 703–709, Sep. 2012, Available: <https://pubmed.ncbi.nlm.nih.gov/22951452/>
- [3] American Heart Association, “Body Mass Index (BMI) In Adults,” [www.heart.org](http://www.heart.org), 2024. <https://www.heart.org/en/healthy-living/healthy-eating/losing-weight/bmi-in-adults>
- [4] J. M. Guirguis-Blake, C. V. Evans, E. M. Webber, E. L. Coppola, L. A. Perdue, and M. S. Weyrich, “Table 1, Blood Pressure Classifications,” [www.ncbi.nlm.nih.gov](http://www.ncbi.nlm.nih.gov), Apr. 01, 2021. <https://www.ncbi.nlm.nih.gov/books/NBK570233/table/ch1.tab1/>
- [5] Medline Plus, “Body Temperature norms: MedlinePlus Medical Encyclopedia,” [Medlineplus.gov](http://Medlineplus.gov), Feb. 02, 2023. <https://medlineplus.gov/ency/article/001982.htm>
- [6] American Heart Association, “Target Heart Rates Chart,” American Heart Association, Mar. 09, 2021. <https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates>
- [7] National Cholesterol Education Program, “Determine Lipoprotein Levels—obtain Complete Lipoprotein Profile after 9-to 12-hour fast. Identify Presence of Clinical Atherosclerotic Disease That Confers High Risk for Coronary Heart Disease (CHD) Events (CHD Risk equivalent) High Blood Cholesterol ATP,” May 2001. Available: <https://www.nhlbi.nih.gov/files/docs/guidelines/atglance.pdf>
- [8] T. K. Mathew, P. Tadi, and M. Zubair, “Blood Glucose Monitoring,” National Library of Medicine, Apr. 23, 2023. <https://www.ncbi.nlm.nih.gov/books/NBK555976/>
- [9] M. C. Fiore, D. E. Jorenby, A. E. Schensky, S. S. Smith, R. R. Bauer, and T. B. Baker, “Smoking Status as the New Vital Sign: Effect on Assessment and Intervention in Patients Who Smoke,” *Mayo Clinic Proceedings*, vol. 70, no. 3, pp. 209–213, Mar. 1995, doi: <https://doi.org/10.4065/70.3.209>.



# Thank You