

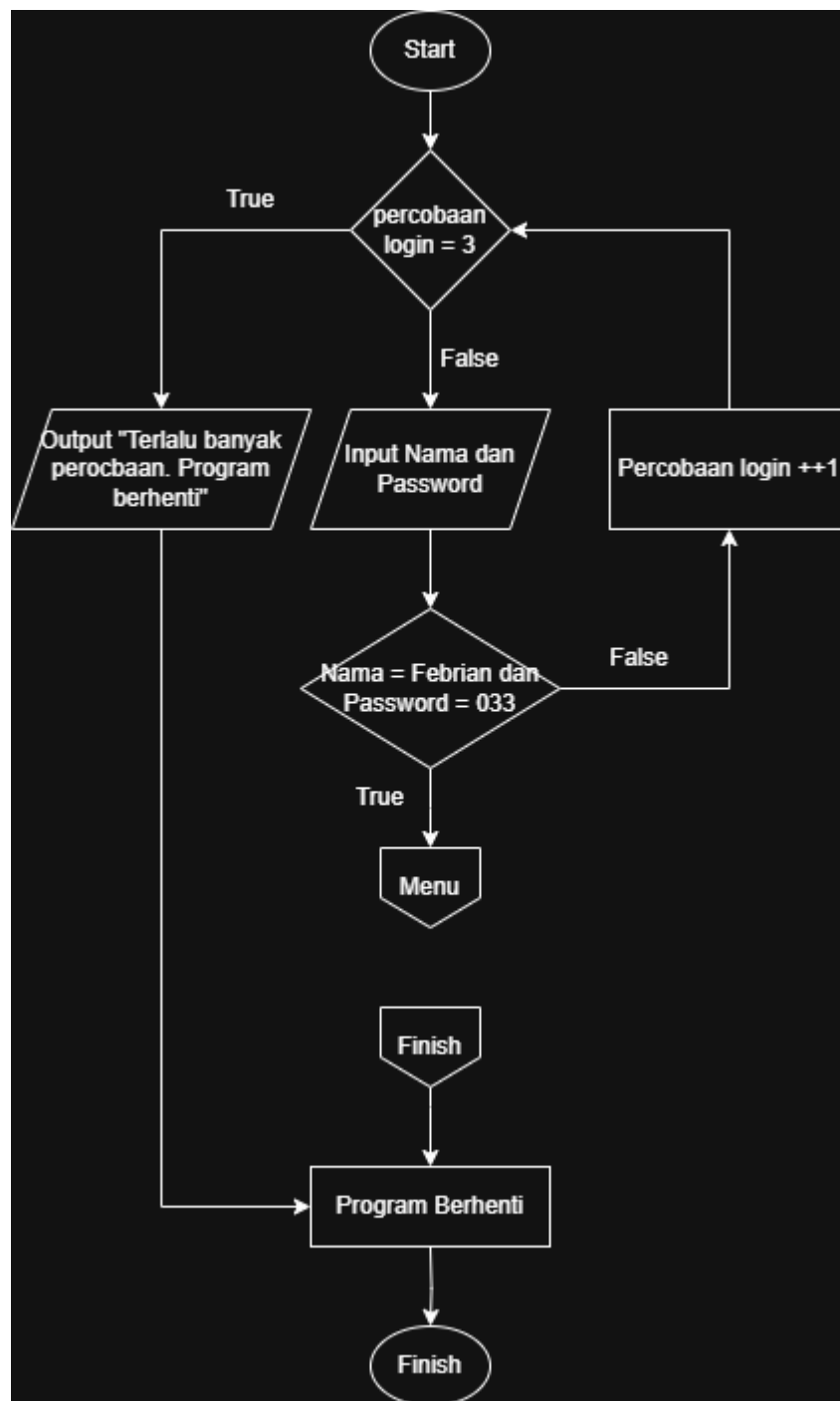
LAPORAN PRAKTIKUM
POSTTEST 5
ALGORITMA PEMROGRAMAN LANJUT



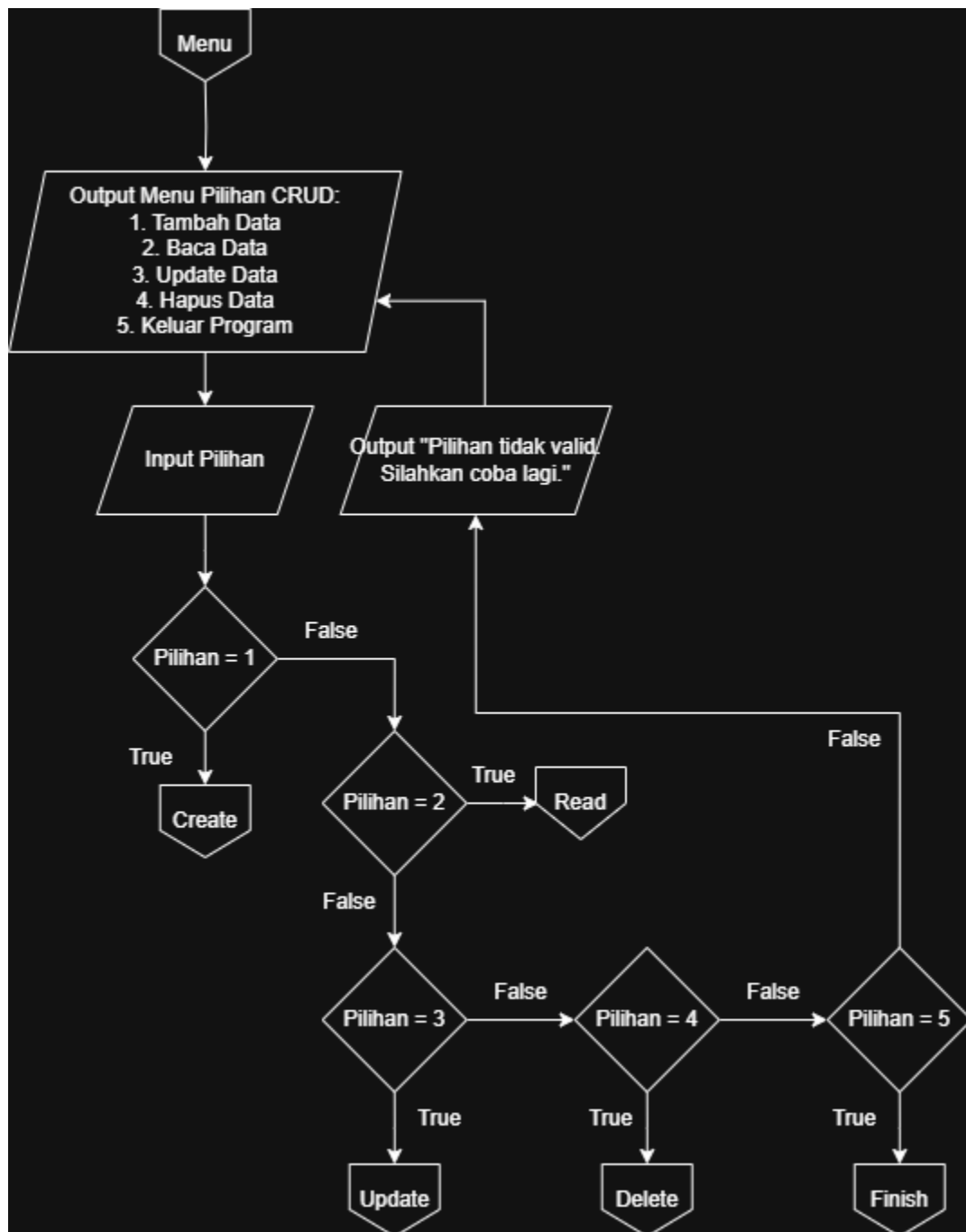
Disusun oleh:
Febrian Pratama Saputra (2409106033)
Kelas (A2 '24)

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN
SAMARINDA
2025

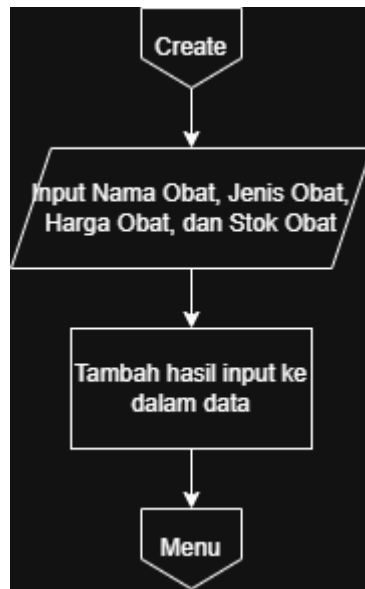
1. Flowchart



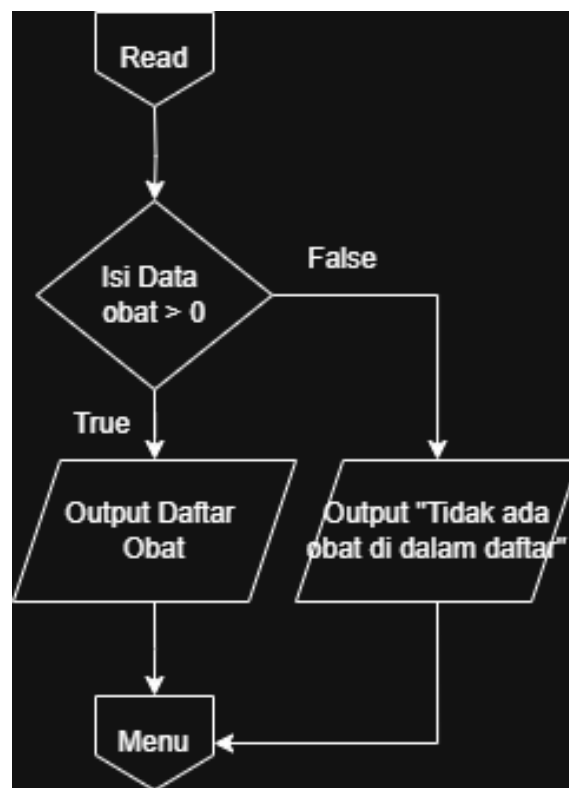
Gambar 1.1 Flowchart Start dan Login



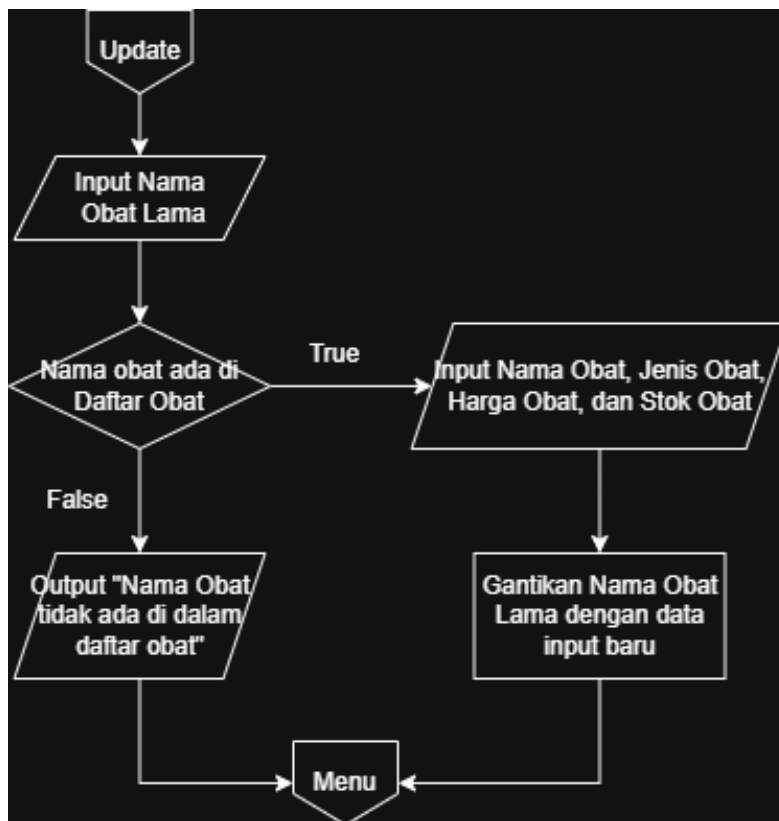
Gambar 1.2 Flowchart Menu



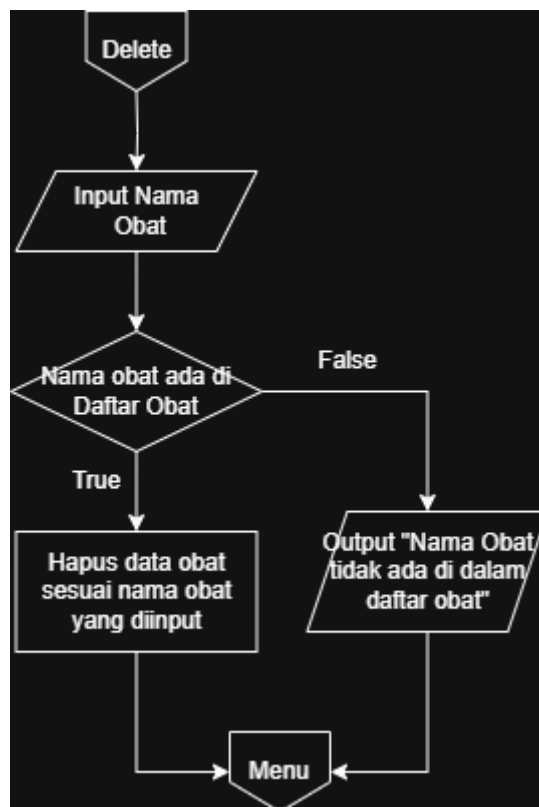
Gambar 1.3 Flowchart Create



Gambar 1.4 Flowchart Read



Gambar 1.5 Flowchart Update



Gambar 1.6 Flowchart Delete

2. Analisis Program

Program berbasis C++ ini berfungsi untuk mengurus data stok obat di sebuah apotek dengan fitur CRUD (Create, Read, Update, Delete). Program ini telah diamankan dengan fitur login untuk mencegah akses tidak sah. Program ini menggunakan tipe data Struct untuk penyimpanan lebih efisien.

3. Source Code

A. Fitur Login

Fitur ini berfungsi untuk mengamankan program dengan membutuhkan kredensial user.

```
bool login() {
    string namaLogin, passwordLogin;
    string namaBenar = "Febrian", passwordBenar = "033";
    int percobaanLogin = 0;

    while (percobaanLogin < 3) {
        cout << "Masukkan Username: ";
        cin >> namaLogin;
        cout << "Masukkan Password: ";
        cin >> passwordLogin;

        if (namaLogin == namaBenar && passwordLogin == passwordBenar) {
            cout << "Login berhasil! Selamat datang pengguna program.\n";
            return true;
        } else {
            cout << "Username atau Password salah! Silahkan coba lagi.\n";
            percobaanLogin++;
        }
    }

    cout << "Maaf! Terlalu banyak percobaan gagal. Program akan segera berhenti.\n";
    return false;
}
```

B. Fitur Menu

Fitur ini berfungsi sebagai menu utama user untuk melakukan CRUD data. Program akan selalu looping kembali ke fitur ini untuk setiap fitur CRUD. Program menggunakan switch case untuk pengalihan fitur lebih efisien.

```
int main() {
    if (!login()) return 0;

    Obat daftarObat[100];
    int jumlah = 0, pilihan;

    do {
        cout << "\nManajemen Stok Obat Apotek";
        cout << "\n1. Tambah Obat";
        cout << "\n2. Tampilkan Obat";
        cout << "\n3. Ubah Obat";
```

```

    cout << "\n4. Hapus Obat";
    cout << "\n5. Keluar";
    cout << "\nPilih menu: ";
    cin >> pilihan;
    cin.ignore();

    switch (pilihan) {
        case 1:
            tambahObat(&daftarObat[jumlah]);
            jumlah++;
            break;
        case 2:
            tampilkanObat(daftarObat, jumlah);
            break;
        case 3:
            ubahObat(daftarObat, jumlah);
            break;
        case 4:
            hapusObat(daftarObat, jumlah);
            break;
        case 5:
            cout << "\nTerima kasih telah menggunakan program ini.\n";
            break;
        default:
            cout << "\nPilihan tidak valid! Silahkan coba lagi.\n";
    }
    while (pilihan != 5);

    return 0;
}

```

C. Fitur Tambah Data

Fitur ini berfungsi untuk menambahkan data ke dalam daftar data. Parameter memiliki sebuah pointer dereference.

```

void tambahObat(Obat* obat) {
    cout << "\nMasukkan Nama Obat: "; getline(cin, obat->nama);
    cout << "Masukkan Jenis Obat: "; getline(cin, obat->jenis);
    cout << "Masukkan Harga Obat: "; cin >> obat->harga;
    cout << "Masukkan Stok Obat: "; cin >> obat->stok;
    cin.ignore();
    cout << "Obat berhasil ditambahkan!\n";
}

```


D. Fitur Tampilkan Data

Fitur ini berfungsi untuk menampilkan isi dari daftar data jika ada isi di dalamnya. Jika nama obat tidak ada, maka user akan langsung dikembalikan ke menu.

```
void tampilkanObat(const Obat daftarObat[], int jumlah) {
    if (jumlah == 0) {
        cout << "Tidak ada obat dalam sistem.\n";
    } else {
        cout << "\nDaftar Obat:\n";
        for (int i = 0; i < jumlah; i++) {
            cout << "Nama: " << daftarObat[i].nama
                << ", Jenis: " << daftarObat[i].jenis
                << ", Harga: " << daftarObat[i].harga
                << ", Stok: " << daftarObat[i].stok << "\n";
        }
    }
}
```

E. Fitur Memperbarui Data

Fitur ini berfungsi untuk memperbarui data yang telah berubah. Jika nama obat tidak ada, maka user akan langsung dikembalikan ke menu.

```
void ubahObat(Obat daftarObat[], int jumlah) {
    string ubah;
    cout << "\nMasukkan Nama Obat yang ingin diubah: "; getline(cin, ubah);
    bool ditemukan = false;
    for (int i = 0; i < jumlah; i++) {
        if (daftarObat[i].nama == ubah) {
            cout << "Masukkan Nama Baru: "; getline(cin, daftarObat[i].nama);
            cout << "Masukkan Jenis Baru: "; getline(cin, daftarObat[i].jenis);
            cout << "Masukkan Harga Baru: "; cin >> daftarObat[i].harga;
            cout << "Masukkan Stok Baru: "; cin >> daftarObat[i].stok;
            cin.ignore();
            cout << "Obat berhasil diperbarui!\n";
            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Nama obat tidak ditemukan!\n";
    }
}
```

F. Fitur Hapus Data

Fitur ini berfungsi untuk menghapus data jika tidak dibutuhkan lagi. Jika nama obat tidak ada, maka user akan langsung dikembalikan ke menu.

```
void hapusObat(Obat daftarObat[], int &jumlah) {
    string hapus;
    cout << "\nMasukkan Nama Obat yang ingin dihapus: "; getline(cin, hapus);
    bool ditemukan = false;
    for (int i = 0; i < jumlah; i++) {
        if (daftarObat[i].nama == hapus) {
            for (int j = i; j < jumlah - 1; j++) {
                daftarObat[j] = daftarObat[j + 1];
            }
            jumlah--;
            cout << "Obat berhasil dihapus!\n";
            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Nama obat tidak ditemukan!\n";
    }
}
```

4. Uji Coba dan Hasil Output

```
Masukkan Username: Febrian
Masukkan Password: 033
Login berhasil! Selamat datang pengguna program.

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 1

Masukkan Nama Obat: Amoxicillin
Masukkan Jenis Obat: Antibiotik
Masukkan Harga Obat: 30000
Masukkan Stok Obat: 100
Obat berhasil ditambahkan!

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 1

Masukkan Nama Obat: Cetirizine
Masukkan Jenis Obat: Antihistamin
Masukkan Harga Obat: 50000
Masukkan Stok Obat: 100
Obat berhasil ditambahkan!
```

Gambar 4.1 Output Login, Menu, dan Create

```
Daftar Obat:
Nama: Amoxicillin, Jenis: Antibiotik, Harga: 30000, Stok: 100
Nama: Cetirizine, Jenis: Antihistamin, Harga: 50000, Stok: 100

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 3

Masukkan Nama Obat yang ingin diubah: Amoxicillin
Masukkan Nama Baru: Vitamin C
Masukkan Jenis Baru: Suplemen-PT5
Masukkan Harga Baru: 20000
Masukkan Stok Baru: 100
Obat berhasil diperbarui!

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 2

Daftar Obat:
Nama: Vitamin C, Jenis: Suplemen-PT5, Harga: 20000, Stok: 100
Nama: Cetirizine, Jenis: Antihistamin, Harga: 50000, Stok: 100
```

Gambar 4.2 Output Read, dan Update

```
Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 4

Masukkan Nama Obat yang ingin dihapus: Cetirizine
Obat berhasil dihapus!

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 2

Daftar Obat:
Nama: Vitamin C, Jenis: Suplemen-PT5, Harga: 20000, Stok: 100

Manajemen Stok Obat Apotek
1. Tambah Obat
2. Tampilkan Obat
3. Ubah Obat
4. Hapus Obat
5. Keluar
Pilih menu: 5

Terima kasih telah menggunakan program ini.
```

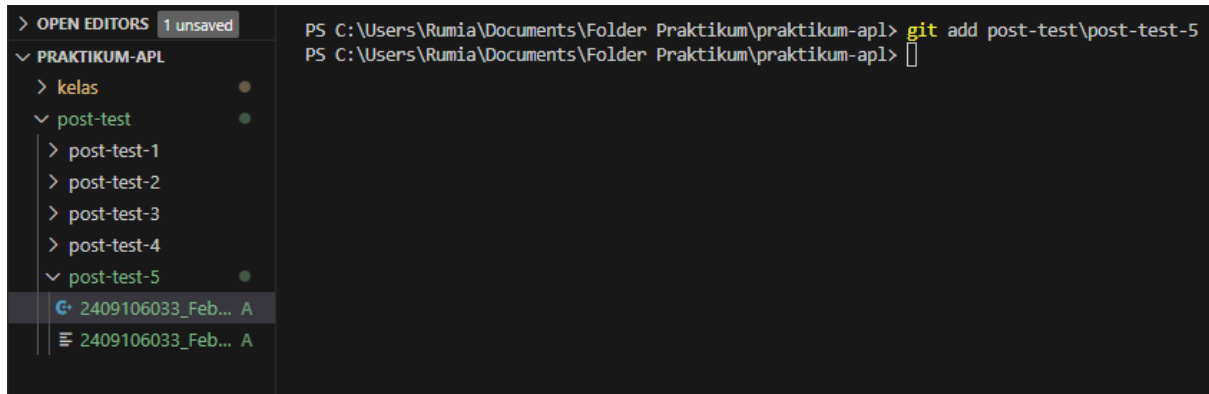
Gambar 4.3 Output Delete, dan Keluar

5. Langkah-Langkah Git pada VSCode

Dikarenakan git telah diinisiasi, dan remote telah tersambung ke github, kita bisa melewati langkah tersebut.

A. Git Add

Menambahkan file-file yang akan dicommit.

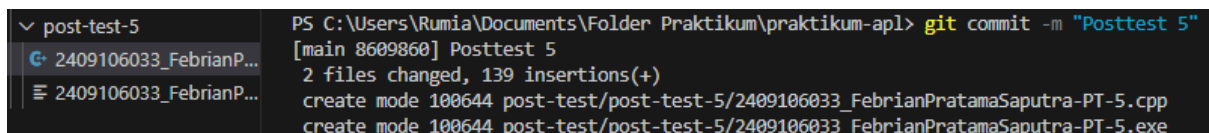


```
PS C:\Users\Rumia\Documents\Folder Praktikum\praktikum-apl> git add post-test\post-test-5
PS C:\Users\Rumia\Documents\Folder Praktikum\praktikum-apl>
```

Gambar 5.1 Git Add

B. Git Commit

Mengcommit file yang telah ditambahkan agar dijadikan menjadi sebuah checkpoint.

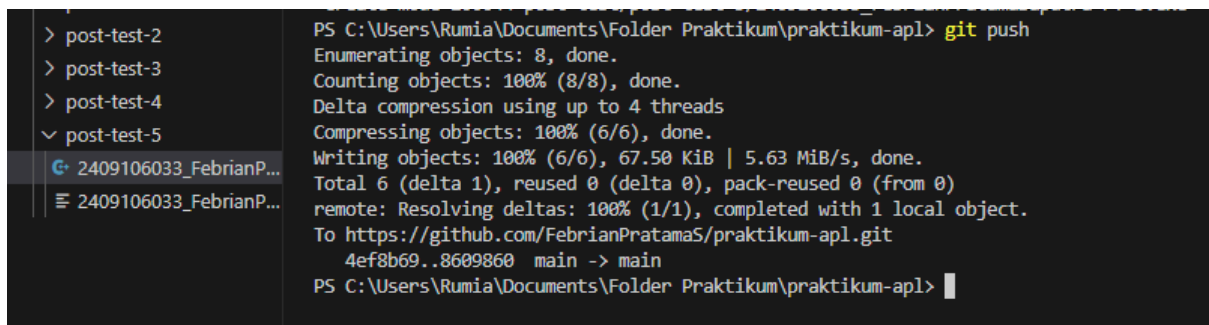


```
PS C:\Users\Rumia\Documents\Folder Praktikum\praktikum-apl> git commit -m "Posttest 5"
[main 8609860] Posttest 5
2 files changed, 139 insertions(+)
create mode 100644 post-test/post-test-5/2409106033_FebrianPratamaSaputra-PT-5.cpp
create mode 100644 post-test/post-test-5/2409106033_FebrianPratamaSaputra-PT-5.exe
```

Gambar 5.2 Git Commit

C. Git Push

Mengupload commit yang telah dilakukan ke repositori github.



```
PS C:\Users\Rumia\Documents\Folder Praktikum\praktikum-apl> git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 67.50 KiB | 5.63 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FebrianPratamaS/praktikum-apl.git
4ef8b69..8609860 main -> main
PS C:\Users\Rumia\Documents\Folder Praktikum\praktikum-apl>
```

Gambar 5.3 Git Push