**LAPORAN PRAKTIKUM POSTTEST 5**

**ALGORITMA PEMROGRAMAN LANJUT**

****

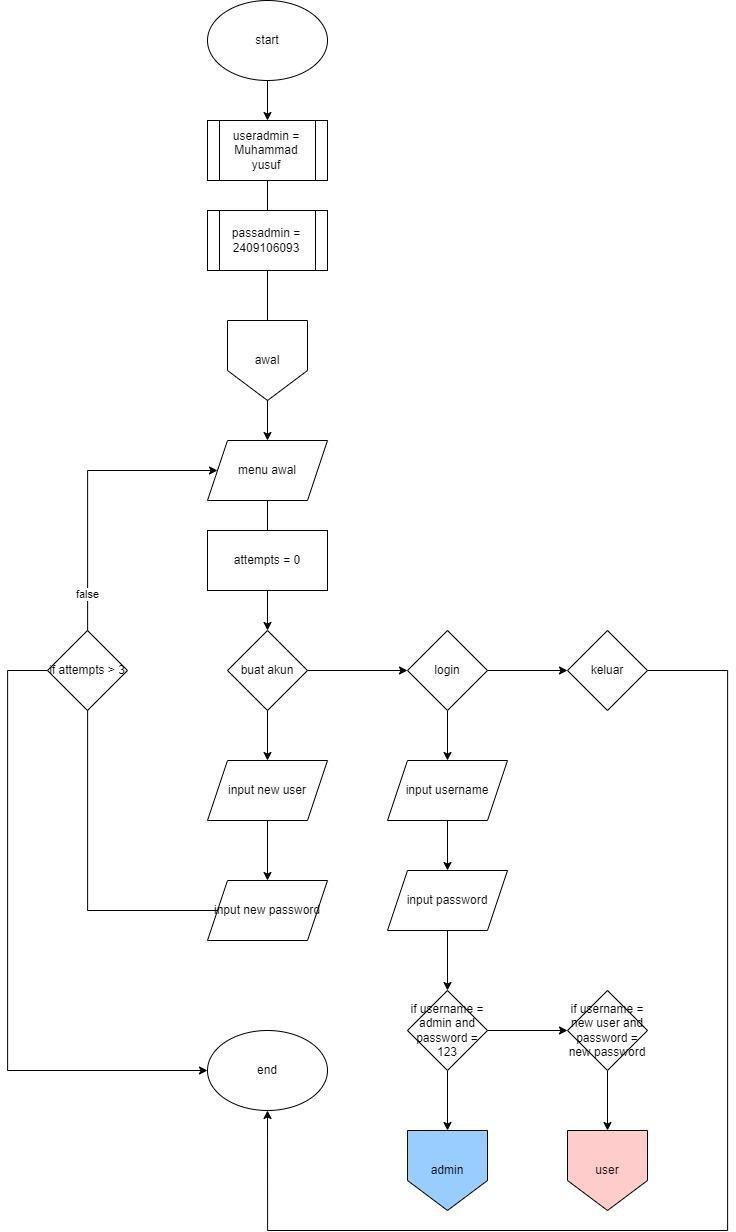
**Disusun oleh: Nama (2409106093)**

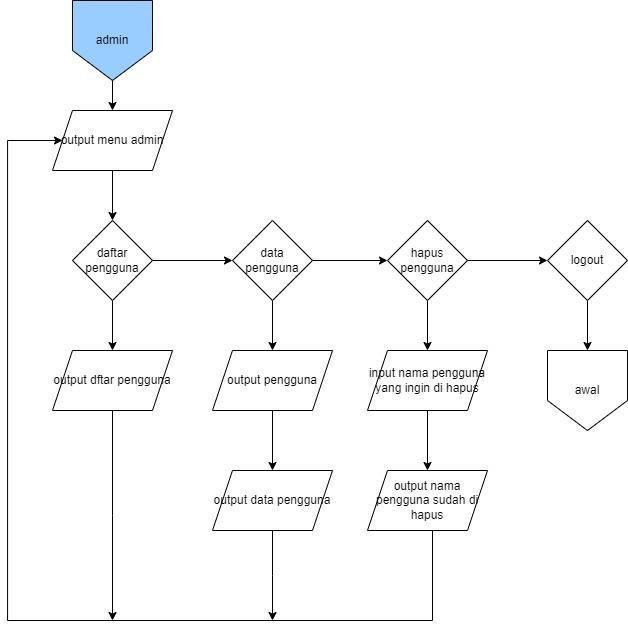
**Kelas (C1‘24)**

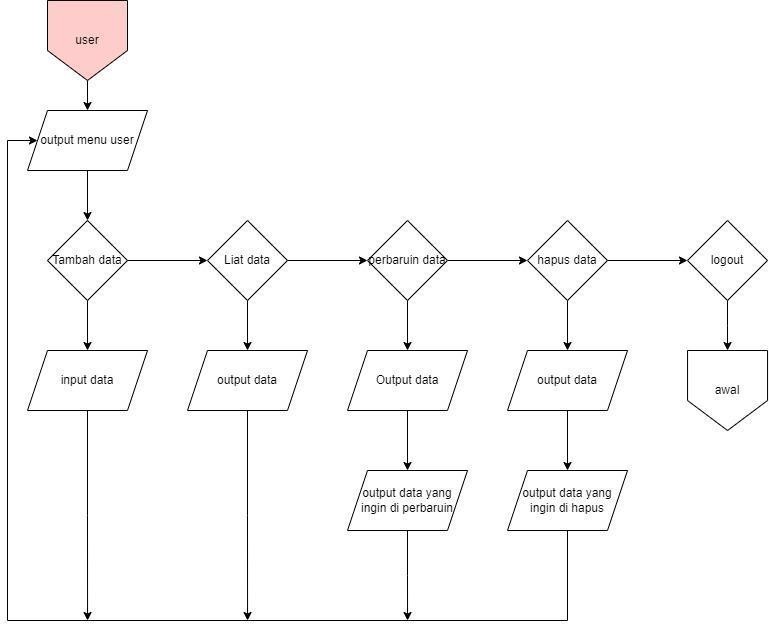
**PROGRAM STUDI INFORMATIKA UNIVERSITAS MULAWARMAN SAMARINDA**

**2025**

1. **Flowchart**

****

****

****

1. **Analisis Program**
   1. **Deskripsi Singkat Program**

Program **AI Archive** ini bertujuan untuk menyediakan sistem penyimpanan dan pengelolaan data berbasis akun pengguna. Manfaat utama program ini adalah:

# Manajemen Pengguna

* + - * Memungkinkan pengguna untuk membuat akun dengan nama (username) dan NIM (password).
      * Admin memiliki kontrol penuh untuk melihat, mengelola, dan menghapus akun pengguna.

# Penyimpanan Data pengguna

* + - * Setiap pengguna dapat menambahkan, membaca, memperbarui, dan menghapus data mereka sendiri.
      * Data yang dimasukkan tersimpan secara terstruktur sesuai dengan akun pengguna masing-masing.

# Keamanan dan Validasi

* + - * Sistem login dengan batasan tiga kali percobaan untuk mencegah akses tidak sah.
      * Admin memiliki akses khusus untuk melihat seluruh data pengguna dan melakukan pengelolaan akun.

# Interaksi yang Mudah

* + - * Menggunakan menu berbasis teks yang intuitif untuk navigasi dan pengelolaan data.
      * Menyediakan fitur logout agar pengguna bisa keluar dari sesi mereka dengan aman.

Dengan fitur-fitur ini, program dapat digunakan sebagai sistem pencatatan sederhana yang memungkinkan pengguna menyimpan dan mengelola informasi pribadi dengan aman.

* 1. **Penjelasan Alur & Algoritma**

1. Deklarasi dan validasi

#include <iostream> #include <string>

#include <iomanip> // Untuk setw dan setfill

using namespace std;

1. Bagian main()

int main() {

    User users[MAX\_USERS];

    int total\_users = 0;

    while (true) {

        displayMainMenu();

        int opsi;

        cin >> opsi;

        cin.ignore();

        switch (opsi) {

            case 1:

                createAccount(users, &total\_users);

                break;

            case 2:

                login(users, &total\_users);

                break;

            case 3:

                cout << "Keluar dari sistem.\n";

                return 0;

            default:

                cout << "Pilihan tidak valid.\n";

        }

    }

}

Di sini kita mengirimkan &total\_users, artinya **alamat dari total\_users** dikirim ke fungsi. Sehingga kalau di dalam fungsi nilainya berubah, maka yang di main() juga berubah. Inilah prinsip **parameter dengan address-of**.

1. Fungsi createAccount

void createAccount(User\* users, int\* total\_users) {

    if (\*total\_users >= MAX\_USERS) {

        cout << "Maksimal pengguna telah tercapai!\n";

        return;

    }

    string username, password;

    cout << "Masukkan Nama: ";

    getline(cin, username);

    if (findUser(users, \*total\_users, username) != -1) {

        cout << "Username sudah terdaftar!\n";

    } else {

        cout << "Masukkan password: ";

        getline(cin, password);

        users[\*total\_users].username = username;

        users[\*total\_users].password = password;

        (\*total\_users)++;

        cout << "Akun berhasil dibuat!\n";

    }

}

Di sini pointer digunakan untuk:

* Mengecek dan mengubah total\_users
* Akses dan manipulasi data user lewat users[\*total\_users]

1. Fungsi userMenu

void userMenu(User\* user) {

    bool logged\_in = true;

    while (logged\_in) {

        cout << "\n=== AI Archive ===\n";

        cout << "+----+---------------------+\n";

        cout << "| 1  | Tambah Data         |\n";

        cout << "| 2  | Baca Data           |\n";

        cout << "| 3  | Perbarui Data       |\n";

        cout << "| 4  | Hapus Data          |\n";

        cout << "| 5  | Logout              |\n";

        cout << "+----+---------------------+\n";

        cout << "Pilih menu: ";

        int opsiuser;

        cin >> opsiuser;

        cin.ignore();

        switch (opsiuser) {

            case 1:

                addUserData(user);

                break;

            case 2:

                viewUserData(\*user);

                break;

            case 3:

                updateUserData(user);

                break;

            case 4:

                deleteUserData(user);

                break;

            case 5:

                cout << "Logout berhasil.\n";

                logged\_in = false;

                break;

            default:

                cout << "Pilihan tidak valid.\n";

        }

    }

}

Fungsi ini menunjukkan contoh **parameter dereference (\*user)** ketika mengakses nilai struct User.

1. Fungsi adduser data

void addUserData(User\* user) {

    if (user->data\_count >= MAX\_DATA) {

        cout << "Maksimal data tercapai!\n";

        return;

    }

    cout << "Masukkan data baru: ";

    getline(cin, user->data[user->data\_count]);

    user->data\_count++;

    cout << "Data berhasil ditambahkan.\n";

}

Ini adalah contoh **akses anggota struct pakai pointer**, yaitu dengan -> bukan .

1. Fungsi updateUserData

void updateUserData(User\* user) {

    viewUserData(\*user);

    if (user->data\_count == 0) return;

    cout << "Masukkan nomor data yang ingin diperbarui: ";

    int index;

    cin >> index;

    cin.ignore();

    if (index > 0 && index <= user->data\_count) {

        cout << "Masukkan data baru: ";

        getline(cin, user->data[index - 1]);

        cout << "Data berhasil diperbarui.\n";

    } else {

        cout << "Nomor data tidak valid!\n";

    }

}

**3. Source Code**

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

const int MAX\_USERS = 100;

const int MAX\_DATA = 50;

struct User {

    string username;

    string password;

    string data[MAX\_DATA];

    int data\_count = 0;

};

const string useradmin = "Muhammad Yusuf";

const string passadmin = "2409106093";

// Fungsi dan Prosedur

int findUser(User users[], int total\_users, const string& username);

void displayMainMenu();

void createAccount(User\* users, int\* total\_users);

void login(User\* users, int\* total\_users);

void adminMenu(User\* users, int\* total\_users);

void userMenu(User\* user);

void viewAllUsers(const User\* users, int total\_users);

void viewAllUserData(const User\* users, int total\_users);

void deleteUser(User\* users, int\* total\_users);

void addUserData(User\* user);

void viewUserData(const User& user);

void updateUserData(User\* user);

void deleteUserData(User\* user);

int main() {

    User users[MAX\_USERS];

    int total\_users = 0;

    while (true) {

        displayMainMenu();

        int opsi;

        cin >> opsi;

        cin.ignore();

        switch (opsi) {

            case 1:

                createAccount(users, &total\_users);

                break;

            case 2:

                login(users, &total\_users);

                break;

            case 3:

                cout << "Keluar dari sistem.\n";

                return 0;

            default:

                cout << "Pilihan tidak valid.\n";

        }

    }

}

void displayMainMenu() {

    cout << "\n====== AI Archive ======\n";

    cout << "+----+---------------------+\n";

    cout << "| No | Opsi                |\n";

    cout << "+----+---------------------+\n";

    cout << "| 1  | Buat Akun           |\n";

    cout << "| 2  | Login               |\n";

    cout << "| 3  | Keluar              |\n";

    cout << "+----+---------------------+\n";

    cout << "Pilih opsi: ";

}

int findUser(User users[], int total\_users, const string& username) {

    for (int i = 0; i < total\_users; i++) {

        if (users[i].username == username) return i;

    }

    return -1;

}

void createAccount(User\* users, int\* total\_users) {

    if (\*total\_users >= MAX\_USERS) {

        cout << "Maksimal pengguna telah tercapai!\n";

        return;

    }

    string username, password;

    cout << "Masukkan Nama: ";

    getline(cin, username);

    if (findUser(users, \*total\_users, username) != -1) {

        cout << "Username sudah terdaftar!\n";

    } else {

        cout << "Masukkan password: ";

        getline(cin, password);

        users[\*total\_users].username = username;

        users[\*total\_users].password = password;

        (\*total\_users)++;

        cout << "Akun berhasil dibuat!\n";

    }

}

void login(User\* users, int\* total\_users) {

    string username, password;

    int attempts = 3;

    while (attempts > 0) {

        cout << "Input Nama: ";

        getline(cin, username);

        cout << "Input password: ";

        getline(cin, password);

        if (username == useradmin && password == passadmin) {

            cout << "Login Admin Berhasil!\n";

            adminMenu(users, total\_users);

            break;

        }

        int userIndex = findUser(users, \*total\_users, username);

        if (userIndex == -1 || users[userIndex].password != password) {

            cout << "Username atau password salah!\n";

            attempts--;

            cout << "Sisa percobaan: " << attempts << endl;

            if (attempts == 0) {

                cout << "Gagal login 3 kali. Coba lagi nanti.\n";

            }

            continue;

        }

        cout << "Login berhasil!\n";

        userMenu(&users[userIndex]);

        break;

    }

}

void adminMenu(User\* users, int\* total\_users) {

    bool admin\_logged\_in = true;

    while (admin\_logged\_in) {

        cout << "\n=== Admin Mode ===\n";

        cout << "+----+---------------------+\n";

        cout << "| 1  | Lihat Semua Pengguna|\n";

        cout << "| 2  | Lihat Data Pengguna |\n";

        cout << "| 3  | Hapus Pengguna      |\n";

        cout << "| 4  | Logout              |\n";

        cout << "+----+---------------------+\n";

        cout << "Pilih menu: ";

        int opsiadmin;

        cin >> opsiadmin;

        cin.ignore();

        switch (opsiadmin) {

            case 1:

                viewAllUsers(users, \*total\_users);

                break;

            case 2:

                viewAllUserData(users, \*total\_users);

                break;

            case 3:

                deleteUser(users, total\_users);

                break;

            case 4:

                cout << "Logout berhasil.\n";

                admin\_logged\_in = false;

                break;

            default:

                cout << "Pilihan tidak valid.\n";

        }

    }

}

void userMenu(User\* user) {

    bool logged\_in = true;

    while (logged\_in) {

        cout << "\n=== AI Archive ===\n";

        cout << "+----+---------------------+\n";

        cout << "| 1  | Tambah Data         |\n";

        cout << "| 2  | Baca Data           |\n";

        cout << "| 3  | Perbarui Data       |\n";

        cout << "| 4  | Hapus Data          |\n";

        cout << "| 5  | Logout              |\n";

        cout << "+----+---------------------+\n";

        cout << "Pilih menu: ";

        int opsiuser;

        cin >> opsiuser;

        cin.ignore();

        switch (opsiuser) {

            case 1:

                addUserData(user);

                break;

            case 2:

                viewUserData(\*user);

                break;

            case 3:

                updateUserData(user);

                break;

            case 4:

                deleteUserData(user);

                break;

            case 5:

                cout << "Logout berhasil.\n";

                logged\_in = false;

                break;

            default:

                cout << "Pilihan tidak valid.\n";

        }

    }

}

void viewAllUsers(const User\* users, int total\_users) {

    cout << "\nDaftar Pengguna:\n";

    for (int i = 0; i < total\_users; i++) {

        cout << "- " << users[i].username << endl;

    }

}

void viewAllUserData(const User\* users, int total\_users) {

    cout << "\n=== Data Seluruh Pengguna ===\n";

    if (total\_users == 0) {

        cout << "Tidak ada pengguna terdaftar.\n";

    } else {

        for (int i = 0; i < total\_users; i++) {

            cout << "\n🔹 Pengguna: " << users[i].username << "\n";

            if (users[i].data\_count == 0) {

                cout << "   ➝ Tidak ada data tersimpan.\n";

            } else {

                for (int j = 0; j < users[i].data\_count; j++) {

                    cout << "   " << j + 1 << ". " << users[i].data[j] << endl;

                }

            }

        }

    }

}

void deleteUser(User\* users, int\* total\_users) {

    cout << "Masukkan nama pengguna yang ingin dihapus: ";

    string user;

    getline(cin, user);

    int index = findUser(users, \*total\_users, user);

    if (index != -1) {

        for (int i = index; i < \*total\_users - 1; i++) {

            users[i] = users[i + 1];

        }

        (\*total\_users)--;

        cout << "Pengguna " << user << " berhasil dihapus.\n";

    } else {

        cout << "Pengguna tidak ditemukan.\n";

    }

}

void addUserData(User\* user) {

    if (user->data\_count >= MAX\_DATA) {

        cout << "Maksimal data tercapai!\n";

        return;

    }

    cout << "Masukkan data baru: ";

    getline(cin, user->data[user->data\_count]);

    user->data\_count++;

    cout << "Data berhasil ditambahkan.\n";

}

void viewUserData(const User& user) {

    cout << "\nData Anda:\n";

    if (user.data\_count == 0) {

        cout << "Tidak ada data tersimpan.\n";

    } else {

        for (int i = 0; i < user.data\_count; i++) {

            cout << i + 1 << ". " << user.data[i] << endl;

        }

    }

}

void updateUserData(User\* user) {

    viewUserData(\*user);

    if (user->data\_count == 0) return;

    cout << "Masukkan nomor data yang ingin diperbarui: ";

    int index;

    cin >> index;

    cin.ignore();

    if (index > 0 && index <= user->data\_count) {

        cout << "Masukkan data baru: ";

        getline(cin, user->data[index - 1]);

        cout << "Data berhasil diperbarui.\n";

    } else {

        cout << "Nomor data tidak valid!\n";

    }

}

void deleteUserData(User\* user) {

    viewUserData(\*user);

    if (user->data\_count == 0) return;

    cout << "Masukkan nomor data yang ingin dihapus: ";

    int index;

    cin >> index;

    cin.ignore();

    if (index > 0 && index <= user->data\_count) {

        for (int i = index - 1; i < user->data\_count - 1; i++) {

            user->data[i] = user->data[i + 1];

        }

        user->data\_count--;

        cout << "Data berhasil dihapus.\n";

    } else {

        cout << "Nomor data tidak valid!\n";

    }

}

1. **Uji Coba dan Hasil Output**
   1. **Uji Coba**
2. Pengguna memilih membuat akun

Pengguna di minta mamasukan new user dan new password

1. Pengguna memilih login

Jika kombinasi nama dan password benar, login berhasil.

Jika salah, program memberikan **3 kesempatan** sebelum gagal login.

1. Metode admin (jika login menggunakan admin)

Pengguna memasukan user = “Muhammad Yusuf” dan password “2409106093” maka program masuk ke **Mode Admin**, dengan opsi:

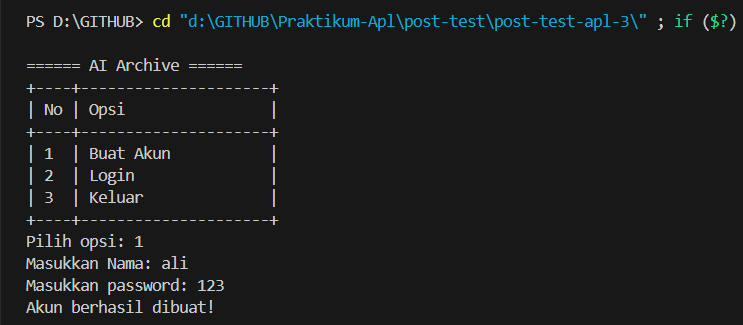
* + Admin bisa melihat semua user dan data mereka.
  + Admin juga bisa **menghapus pengguna** dengan memasukkan nama.

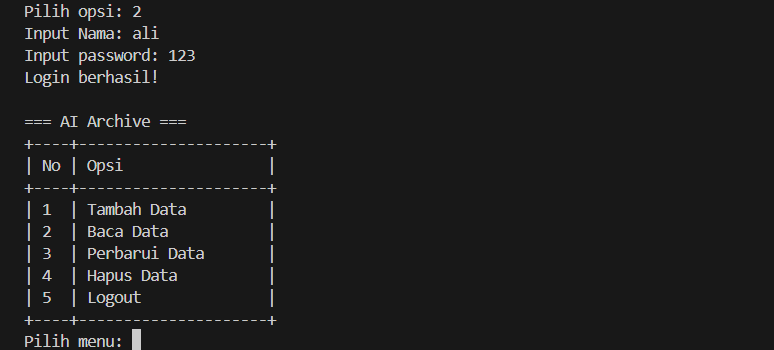
1. Mode User (Jika Login sebagai Pengguna Biasa)

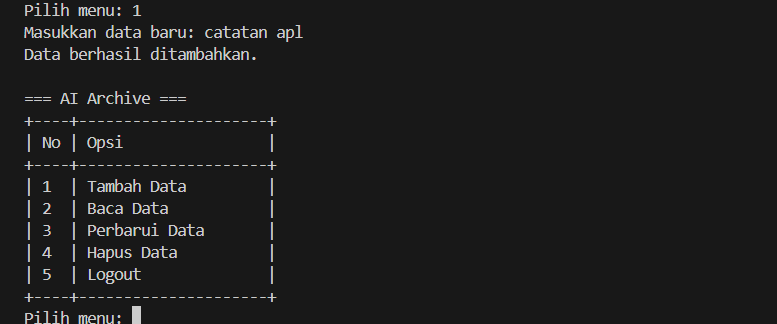
Setelah login, pengguna bisa mengelola data dengan menu

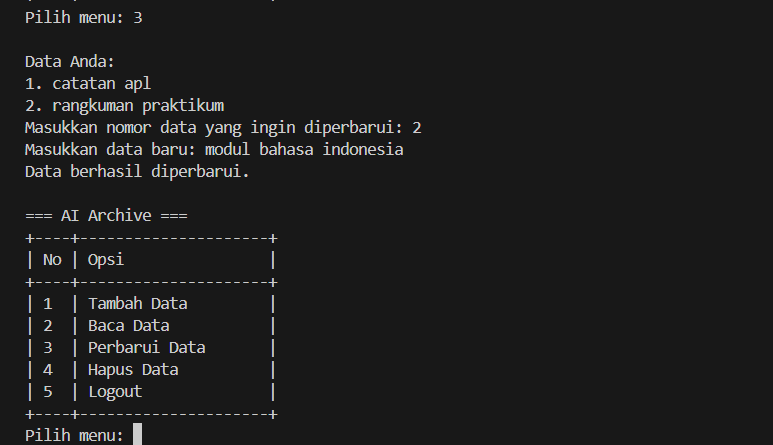
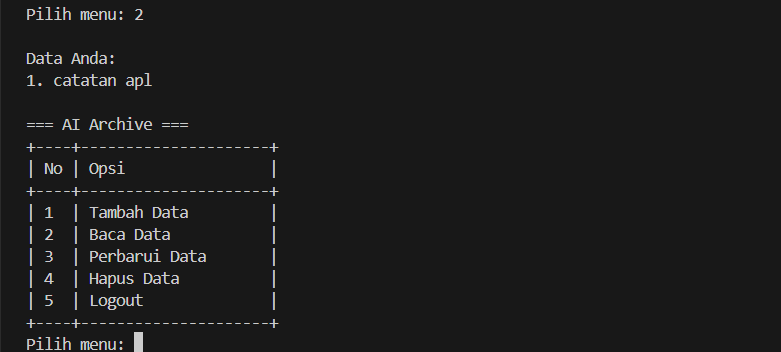
* + Jika memilih **1 (Tambah Data)**, pengguna bisa memasukkan teks
  + Jika memilih **2 (Baca Data)**, program menampilkan semua data pengguna
  + Jika memilih **3 (Perbarui Data)**, pengguna bisa memilih nomor data untuk diperbarui
  + Jika memilih **4 (Hapus Data)**, pengguna bisa menghapus data tertentu

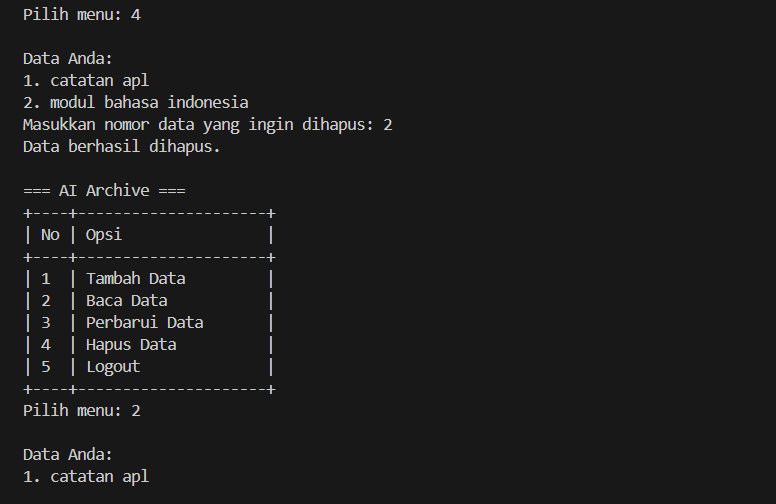
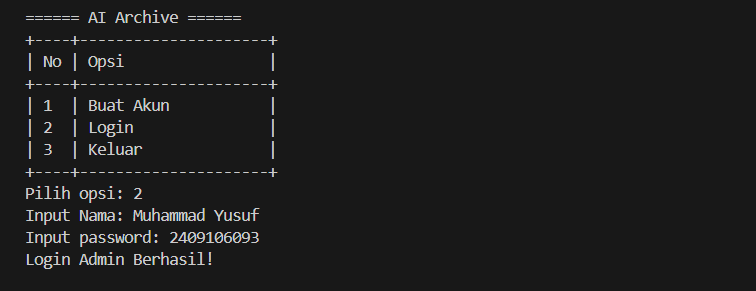
1. Log out
   * Jika memilih **5 (Logout)**, pengguna akan keluar ke menu utama.
   * Jika memilih **3 (Keluar)** di menu utama, program berhenti.
   1. **Hasil Output**

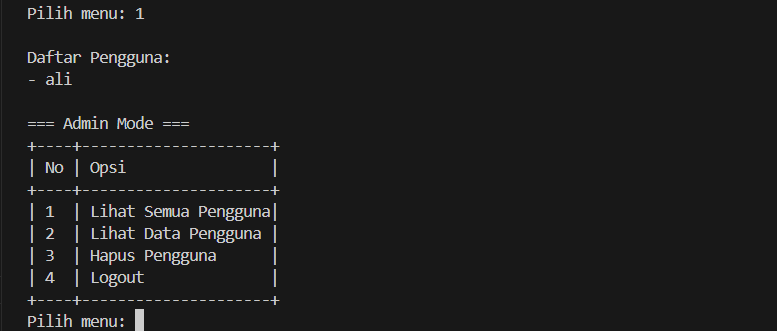
****

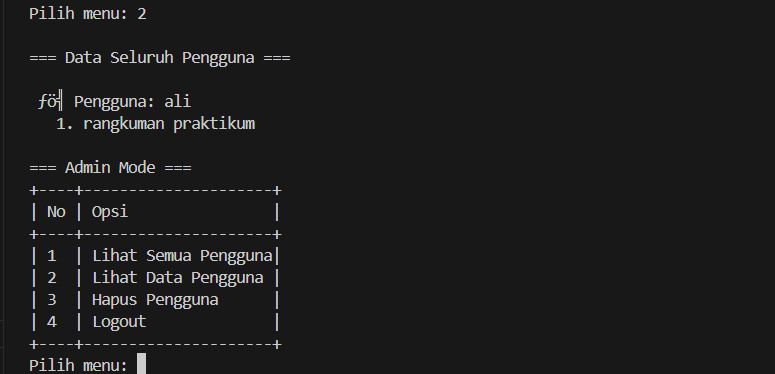


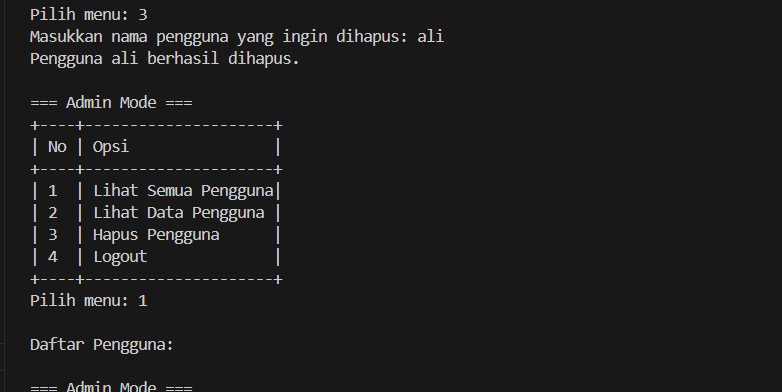
****

****

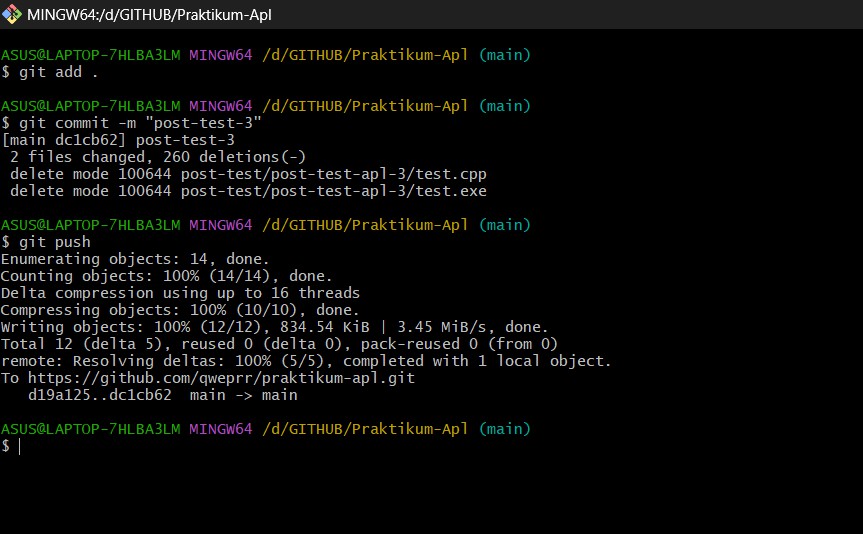


****

****



# Git



1. Menambahkan File ke Staging Area

$ git add .

* + Perintah ini menambahkan semua file yang ada di dalam folder ke staging area.
  + Staging area adalah tempat sementara sebelum file dikomit ke dalam repository.

1. Menambahkan Remote Repository (Gagal karena Sudah Ada)

$ git remote add origin https://github.com/qweprr/praktikum-apl

* + Perintah ini digunakan untuk menambahkan repository remote dengan nama origin.
  + Error: "remote origin already exists", ini terjadi karena sebelumnya sudah ada repository remote yang dikaitkan dengan nama origin.
  + ​

1. Membuat Commit dengan Pesan "Update"

$ git commit -m "Update"

* + Perintah ini menyimpan perubahan dalam repository dengan commit dan pesan"Update".
  + File yang dikomit:
    - Post-test/Post-test-1/2409106093-Muhammadyusuf-PT-1.cpp
    - Post-test/Post-test-1/2409106093-Muhammadyusuf-PT-1.cpp

1. Mendorong (Push) Perubahan ke Repository Remote

$ git push -u origin main

* + Perintah ini mengunggah (push) perubahan ke repository remote pada branch main.
  + Karena ini adalah push pertama, flag -u digunakan untuk mengatur branch lokal main agar terhubung dengan branch main di remote repository.
  + Proses yang terjadi:
    - Menghitung objek (Enumerating objects: 6).
    - Mengompresi objek sebelum mengunggahnya.
    - Menulis (mengunggah) objek ke GitHub.
    - Menampilkan informasi bahwa branch main sekarang dilacak oleh remote repository origin/main.