LAPORAN PRAKTIKUM

POSTTEST 6

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Nama

(2409106093)

Kelas (C1‘24)

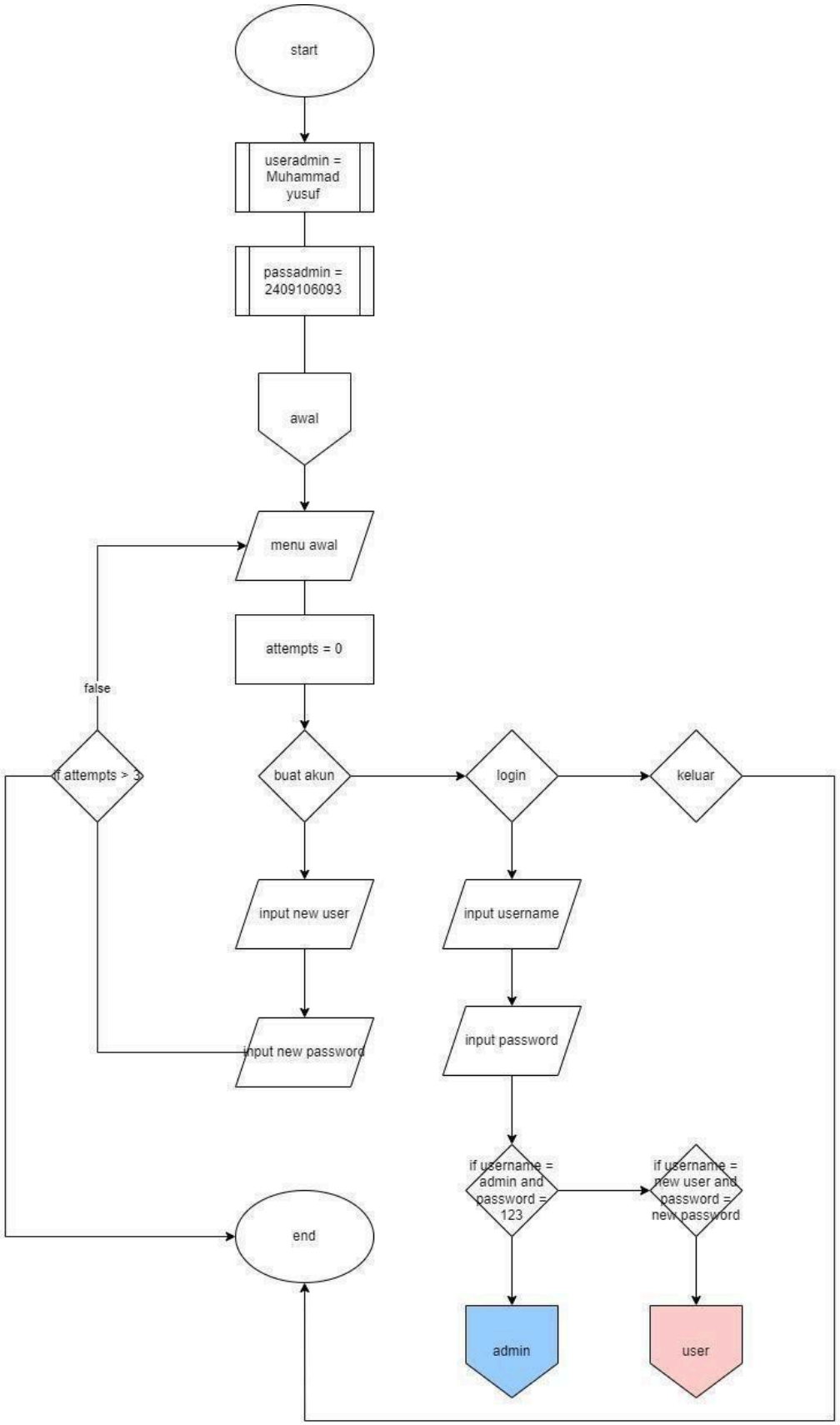
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

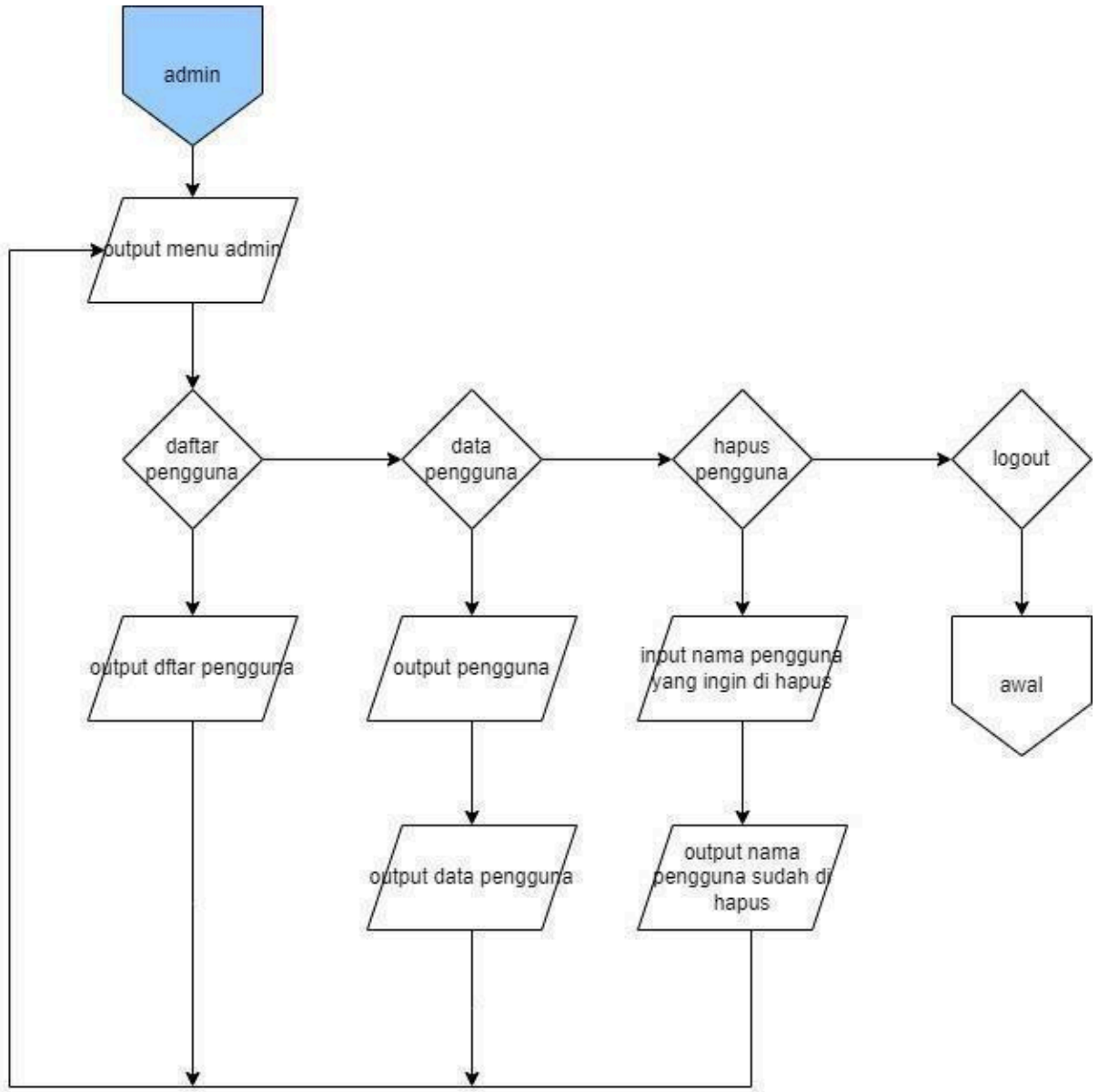
SAMARINDA

2025

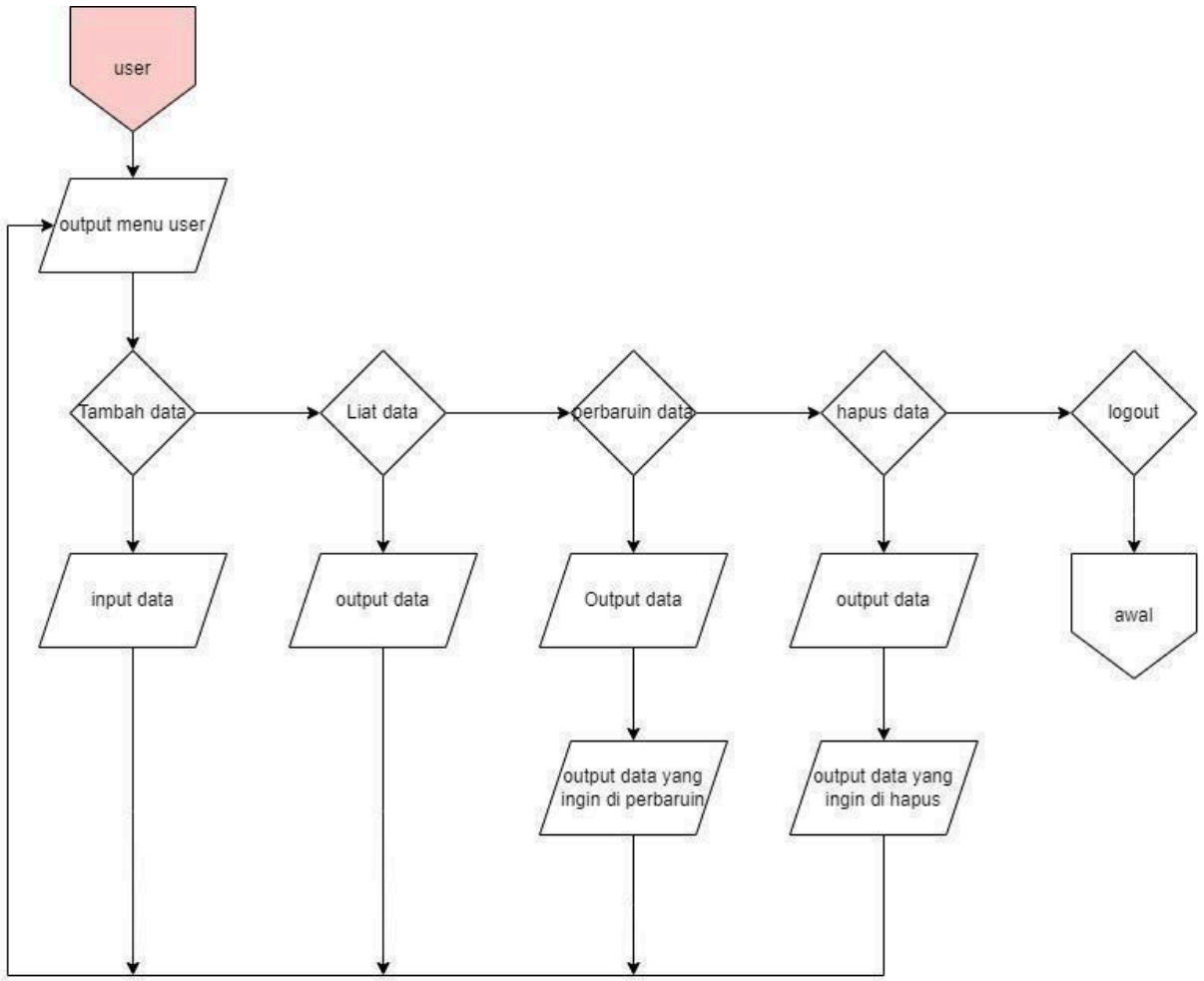
1. Flowchart


start
useradmin = Muhammad yusuf
passadmin = 2409106093
awal
menu awal
attempts = 0
false
if attempts > 3
buat akun
login
keluar
input new user
input username
input new password
input password
if username = admin and password = 123
if username = new user and password = new password
end
admin
user

•


admin
output menu admin
daftar pengguna
data pengguna
hapus pengguna
logout
output dftar pengguna
output pengguna
input nama pengguna yang ingin di hapus
awal
output data pengguna
output nama pengguna sudah di hapus

-


user
output menu user
Tambah data
Liat data
perbaruin data
hapus data
logout
input data
output data
Output data
output data
awal
output data yang ingin di perbaruin
output data yang ingin di hapus

2. Analisis Program
2.1 Deskripsi Singkat Program

1. Manajemen Pengguna

- Program menyediakan fitur untuk **membuat akun** baru dengan memasukkan *username* dan *password* .
- Program memeriksa apakah username sudah ada atau belum sebelum membuat akun baru (menghindari duplikasi).
- Terdapat **akun admin** (username: *Muhammad Yusuf*, password: *2409106093*) yang memiliki kontrol penuh untuk:
    1. Melihat daftar semua pengguna.
    2. Melihat data semua pengguna.
    3. Menghapus akun pengguna tertentu.
- Pengguna biasa hanya bisa mengakses dan mengelola data mereka sendiri.

2. Penyimpanan Data Pengguna

- Setiap akun pengguna memiliki array data (string data[MAX_DATA]) untuk menyimpan informasi pribadi.
- Pengguna dapat melakukan:
    1. **Menambahkan** data baru ke dalam akun mereka.
    2. **Membaca** seluruh data yang telah mereka input.
    3. **Memperbarui** data tertentu dengan memilih berdasarkan nomor urut.
    4. **Menghapus** data yang tidak diperlukan lagi.
- Admin dapat **melihat semua data** milik semua pengguna, tetapi tidak bisa mengedit data pengguna lain (hanya menghapus akun jika diperlukan).

3. Keamanan dan Validasi

- **Login sistem** dibatasi maksimal **3 kali percobaan**. Jika gagal, pengguna harus keluar dari program.
- Sistem membedakan login:
    1. Jika login sebagai **admin**, maka masuk ke menu admin.
    2. Jika login sebagai **user biasa**, maka masuk ke menu user pribadi.
- Validasi dilakukan saat:
    1. Membuat akun baru (cek duplikasi username).
    2. Login (cek kecocokan username dan password).
    3. Menginput pilihan menu (penanganan input tidak valid).

4. Interaksi yang Mudah

- Program menggunakan **menu berbasis teks** yang sederhana dan terstruktur:
    1. Menu utama: Membuat akun, Login, Keluar.
    2. Menu admin: Lihat semua pengguna, lihat semua data, hapus pengguna, logout.
    3. Menu user: Tambah data, baca data, perbarui data, hapus data, sortir data, logout.
- Setiap menu memiliki nomor opsi yang jelas untuk navigasi cepat.
- Setelah logout, pengguna kembali ke menu utama.
- Terdapat **fitur sorting data**:
    1. **Bubble Sort** untuk mengurutkan berdasarkan huruf (ascending).
    2. **Selection Sort** untuk mengurutkan data angka (descending).
    3. **Merge Sort** untuk mengurutkan huruf dengan cara lebih efisien.

2.2. Penjelasan Alur & Algoritma

1. Deklarasi

```
#include <iostream>
#include <string>
#include <iomanip>
#include <sstream>
using namespace std;
```

2. Main (menu)

```
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";
        }
    }
}
```

- **users adalah array berisi semua akun pengguna.**
- **total_users menyimpan berapa banyak akun yang telah dibuat.**
- **Perulangan while (true) membuat program terus berjalan sampai user memilih keluar.**
- **Saat memanggil createAccount(users, &total_users), kita mengirimkan alamat dari total_users.**
- **Kenapa pakai &?**
- **Karena kalau jumlah user bertambah, kita mau nilai aslinya di main() ikut berubah.**
- **➔ Inilah prinsip parameter dengan address-of (&).**

3. **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";
    }
}
```

- **Pointer digunakan di sini untuk:**

  Mengakses dan mengubah total_users.
  Menyimpan data baru ke array users.

- ***total_users artinya mengambil nilai dari alamat total_users.**

- **users[*total_users] adalah tempat untuk menyimpan data akun baru.**

4. **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 << "| 6  | Sorting Data        |\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;
            case 6:
                sortUserData(user);
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    }
}
```

1. Saat memilih menu, fungsi akan memanggil sub-fungsi lain.
2. user dioper dengan pointer (User* user) supaya perubahan yang dilakukan langsung mengubah data aslinya.
3. Pada viewUserData(*user), pointer di-dereference untuk membaca nilai struct User secara penuh.

5. Fungsu 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";
}
```

- **Fungsi ini untuk menambah data baru ke akun user.**

- **user->data_count menunjukkan jumlah data saat ini.**
- **user->data[user->data_count++] = newData;**
  Simpan newData di posisi saat ini.
  Lalu increment (++) jumlah data.
- **user-> artinya kita mengakses anggota struct melalui pointer.**

2.3. 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);

// Fungsi Sorting
float extractSize(const string& data);
void bubbleSortByDescription(User* user);
void selectionSortBySizeDescending(User* user);
void merge(string arr[], int l, int m, int r);
void mergeSortBySizeAscending(string arr[], int l, int r);
void applyMergeSort(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 << "| 6  | Sorting Data        |\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;
            case 6: {
                int metode;
                cout << "\nPilih metode sorting:\n";
                cout << "1. Huruf (Ascending - Bubble Sort)\n";
                cout << "2. Ukuran (Descending - Selection Sort)\n";
                cout << "3. Ukuran (Ascending - Merge Sort)\n";
                cout << "Pilihan: ";
                cin >> metode;
                cin.ignore();
                switch (metode) {
                    case 1: bubbleSortByDescription(user); break;
                    case 2: selectionSortBySizeDescending(user); break;
                    case 3: applyMergeSort(user); break;
                    default: cout << "Pilihan tidak valid.\n";
                }
                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;
    }

    string description;
    float sizeInMB;

    cout << "Masukkan deskripsi data baru: ";
    getline(cin, description);
```

```
    cout << "Masukkan ukuran data/MB: ";
    cin >> sizeInMB;
    cin.ignore();

    if (sizeInMB <= 0) {
        cout << "Ukuran data tidak valid! Harus lebih besar dari 0.\n";
        return;
    }

    user->data[user->data_count] = description + " (Ukuran: " +
to_string(sizeInMB) + " MB)";
    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++) {
            string data = user.data[i];
            size_t start = data.find("(Ukuran: ");
            size_t end = data.find(" MB)");

            if (start != string::npos && end != string::npos) {
                string sizeStr = data.substr(start + 9, end - (start + 9));
                cout << i + 1 << ". " << data.substr(0, start) << " (Ukuran: "
<< sizeStr << " MB)" << endl;
            } else {
                cout << i + 1 << ". " << data << 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";
    }
}

// Fungsi Sorting

float extractSize(const string& data) {
    size_t start = data.find("(Ukuran: ");
    size_t end = data.find(" MB)");
    if (start != string::npos && end != string::npos) {
        string sizeStr = data.substr(start + 9, end - (start + 9));
        return stof(sizeStr);
    }
    return 0;
}

void bubbleSortByDescription(User* user) {
    for (int i = 0; i < user->data_count - 1; i++) {
        for (int j = 0; j < user->data_count - i - 1; j++) {
            string desc1 = user->data[j].substr(0,
user->data[j].find("(Ukuran:"));
            string desc2 = user->data[j + 1].substr(0, user->data[j +
1].find("(Ukuran:"));

            if (desc1 > desc2) {
                swap(user->data[j], user->data[j + 1]);
```

```
            }
        }
    }
    cout << "Data berhasil diurutkan berdasarkan deskripsi (ascending).\n";
}

void selectionSortBySizeDescending(User* user) {
    for (int i = 0; i < user->data_count - 1; i++) {
        int maxIdx = i;
        for (int j = i + 1; j < user->data_count; j++) {
            if (extractSize(user->data[j]) > extractSize(user->data[maxIdx])) {
                maxIdx = j;
            }
        }
        swap(user->data[i], user->data[maxIdx]);
    }
    cout << "Data berhasil diurutkan berdasarkan ukuran (descending).\n";
}

void merge(string arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    string L[n1], R[n2];

    for (int i = 0; i < n1; i++) {
        L[i] = arr[l + i];
    }
    for (int i = 0; i < n2; i++) {
        R[i] = arr[m + 1 + i];
    }

    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (extractSize(L[i]) <= extractSize(R[j])) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
```

```
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSortBySizeAscending(string arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSortBySizeAscending(arr, l, m);
        mergeSortBySizeAscending(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

void applyMergeSort(User* user) {
    mergeSortBySizeAscending(user->data, 0, user->data_count - 1);
    cout << "Data berhasil diurutkan berdasarkan ukuran (ascending).\n";
}
```

3. Uji Coba dan Hasil Output

3.1 Uji Coba

1. Pengguna memilih membuat akun
   Pengguna di minta mamasukan new user dan new password

2. Pengguna memilih login
   Jika kombinasi nama dan password benar, login berhasil.
   Jika salah, program memberikan **3 kesempatan** sebelum gagal login.

3. 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.

4. 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

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

3.2 Hasil Output

```
PS D:\GITHUB> cd "d:\GITHUB\Praktikum-Apl\post-test\post-test-apl-3\" ; if ($?)

====== AI Archive ======
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Buat Akun          |
| 2  | Login              |
| 3  | Keluar             |
+----+--------------------+
Pilih opsi: 1
Masukkan Nama: ali
Masukkan password: 123
Akun berhasil dibuat!
```

```
Pilih opsi: 2
Input Nama: ali
Input password: 123
Login berhasil!

=== AI Archive ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
+----+--------------------+
Pilih menu:
```

```
Pilih menu: 1
Masukkan data baru: catatan apl
Data berhasil ditambahkan.

=== AI Archive ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
+----+--------------------+
Pilih menu:
```

```
Pilih menu: 2

Data Anda:
1. catatan apl

=== AI Archive ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
+----+--------------------+
Pilih menu:
```

```
Pilih menu: 3

Data Anda:
1. catatan apl
2. rangkuman praktikum
Masukkan nomor data yang ingin diperbarui: 2
Masukkan data baru: modul bahasa indonesia
Data berhasil diperbarui.

=== AI Archive ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
+----+--------------------+
Pilih menu:
```

```
Pilih menu: 4

Data Anda:
1. catatan apl
2. modul bahasa indonesia
Masukkan nomor data yang ingin dihapus: 2
Data berhasil dihapus.

=== AI Archive ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
+----+--------------------+
Pilih menu: 2

Data Anda:
1. catatan apl
```

```
====== AI Archive ======
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Buat Akun          |
| 2  | Login              |
| 3  | Keluar             |
+----+--------------------+
Pilih opsi: 2
Input Nama: Muhammad Yusuf
Input password: 2409106093
Login Admin Berhasil!
```

```
Pilih menu: 1

Daftar Pengguna:
- ali

=== Admin Mode ===
+----+--------------------+
| No | Opsi               |
+----+--------------------+
| 1  | Lihat Semua Pengguna|
| 2  | Lihat Data Pengguna |
| 3  | Hapus Pengguna     |
| 4  | Logout             |
+----+--------------------+
Pilih menu:
```

```
Pilih menu: 2

=== Data Seluruh Pengguna ===

 fö╣ Pengguna: ali
   1. rangkuman praktikum

=== Admin Mode ===
+----+---------------------+
| No | Opsi                |
+----+---------------------+
| 1  | Lihat Semua Pengguna|
| 2  | Lihat Data Pengguna |
| 3  | Hapus Pengguna      |
| 4  | Logout              |
+----+---------------------+
Pilih menu:
```

```
Pilih menu: 3
Masukkan nama pengguna yang ingin dihapus: ali
Pengguna ali berhasil dihapus.

=== Admin Mode ===
+----+---------------------+
| No | Opsi                |
+----+---------------------+
| 1  | Lihat Semua Pengguna|
| 2  | Lihat Data Pengguna |
| 3  | Hapus Pengguna      |
| 4  | Logout              |
+----+---------------------+
Pilih menu: 1

Daftar Pengguna:

=== Admin Mode ===
```

```
Data Anda:
1. catatan apl  (Ukuran: 50.000000 MB)
2. rangkuman praktikum  (Ukuran: 500.000000 MB)
3. jaringan  (Ukuran: 5000.000000 MB)

=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 6

Pilih metode sorting:
1. Huruf (Ascending - Bubble Sort)
2. Ukuran (Descending - Selection Sort)
3. Ukuran (Ascending - Merge Sort)
Pilihan: 1
Data berhasil diurutkan berdasarkan deskripsi (ascending).

=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 2

Data Anda:
1. catatan apl  (Ukuran: 50.000000 MB)
2. jaringan  (Ukuran: 5000.000000 MB)
3. rangkuman praktikum  (Ukuran: 500.000000 MB)
```

```
=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 6

Pilih metode sorting:
1. Huruf (Ascending - Bubble Sort)
2. Ukuran (Descending - Selection Sort)
3. Ukuran (Ascending - Merge Sort)
Pilihan: 2
Data berhasil diurutkan berdasarkan ukuran (descending).

=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 2

Data Anda:
1. jaringan  (Ukuran: 5000.000000 MB)
2. rangkuman praktikum  (Ukuran: 500.000000 MB)
3. catatan apl  (Ukuran: 50.000000 MB)
```

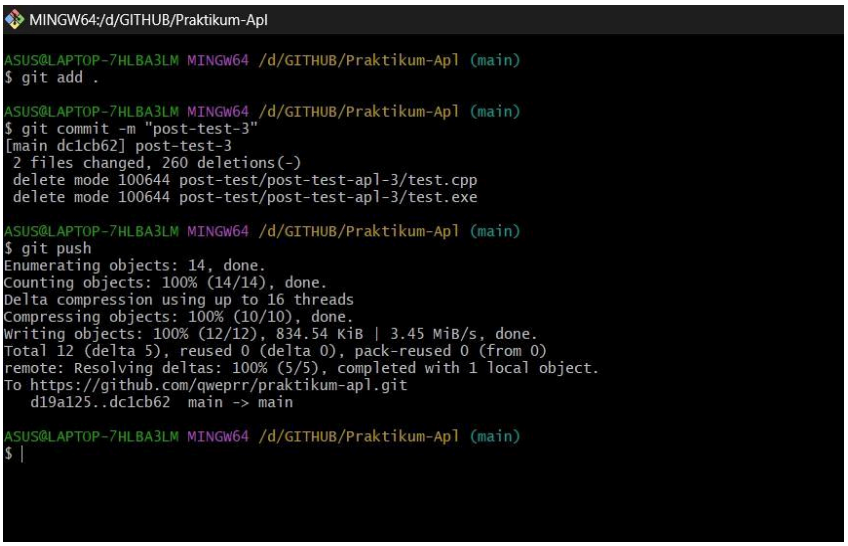
```
=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 6

Pilih metode sorting:
1. Huruf (Ascending - Bubble Sort)
2. Ukuran (Descending - Selection Sort)
3. Ukuran (Ascending - Merge Sort)
Pilihan: 3
Data berhasil diurutkan berdasarkan ukuran (ascending).

=== AI Archive ===
+----+--------------------+
| 1  | Tambah Data        |
| 2  | Baca Data          |
| 3  | Perbarui Data      |
| 4  | Hapus Data         |
| 5  | Logout             |
| 6  | Sorting Data       |
+----+--------------------+
Pilih menu: 2

Data Anda:
1. catatan apl  (Ukuran: 50.000000 MB)
2. rangkuman praktikum  (Ukuran: 500.000000 MB)
3. jaringan  (Ukuran: 5000.000000 MB)
```

4. 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.
2. 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.
3. Membuat Commit dengan Pesan "Post-test-6"
   $ 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-6.cpp
      - Post-test/Post-test-1/2409106093-Muhammadyusuf-PT-6.cpp
4. 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.