**LAPORAN PRAKTIKUM**

**POSTTEST 4**

**ALGORITMA PEMROGRAMAN LANJUT**



**Disusun oleh:**

**Ade Pasiha Tangke Allo (2409106109)**

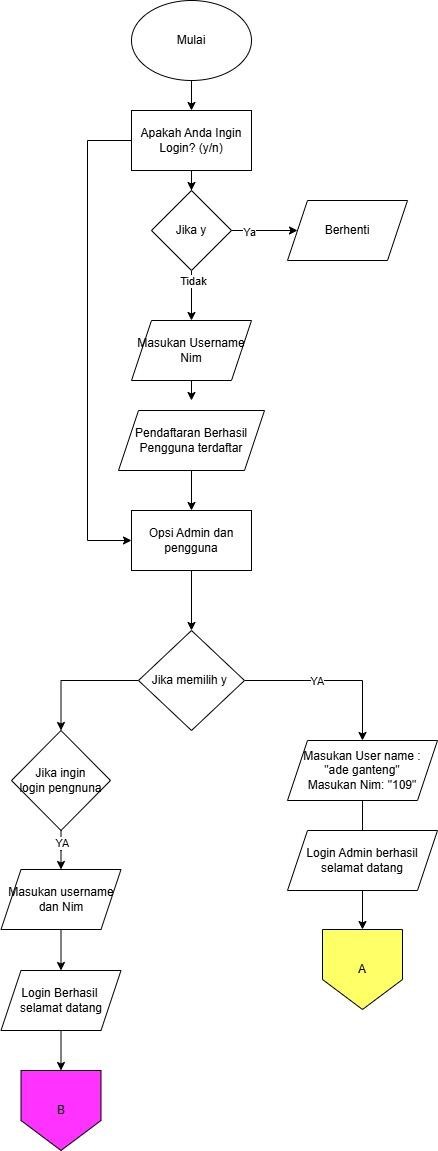
**Kelas (C2 ‘24)**

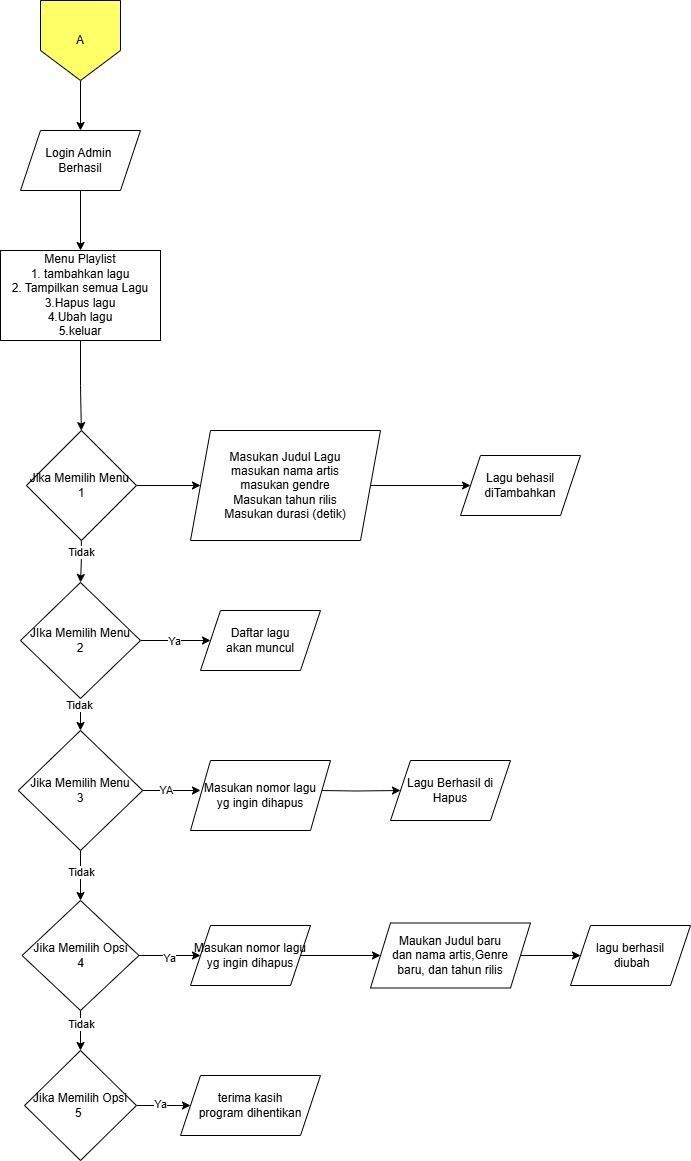
**PROGRAM STUDI INFORMATIKA**

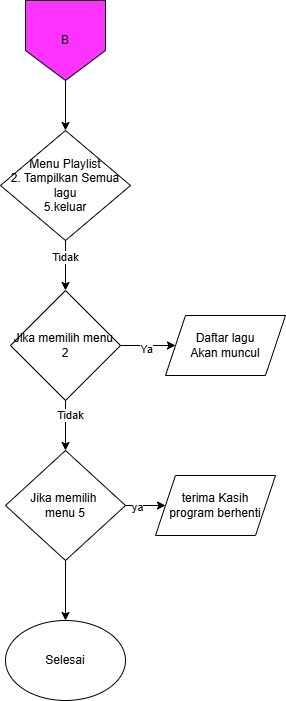
**UNIVERSITAS MULAWARMAN**

**SAMARINDA** **2025**

# 1. Flowchart







# 2. Analisis Program

## **2.1 Deskripsi Singkat Program**

Program ini saya buat untuk mengelola playlist lagu, di mana pengguna bisa melihat daftar lagu, dan kalau dia login sebagai admin, bisa juga menambah, menghapus, atau mengubah data lagu.Setiap user harus login dulu, dan kalau username-nya "adeganteng" dengan NIM "109", dia otomatis jadi admin.Data lagu disimpan dalam array, dan ditampilkan dalam bentuk tabel. Untuk menampilkan semua lagu, saya pakai rekursi biar lebih menarik.Program ini juga punya fitur registrasi user, pembatasan jumlah lagu dan user, serta validasi login maksimal 3 kali.Intinya, program ini semacam sistem playlist sederhana dengan perbedaan hak akses antara user biasa dan admin.

1. **Souce Code**
2. **Librari dan Konstanta**
3. #include <iostream>
4. #include <string>
5. #include <iomanip>

* # include untuk mengimpor Pustaka C++ standar.
* Iostream input/output (missal; cin, cout).
* String untuk mengunakan tipe data string
* Iomanip untuk mengatur format output ( setw untuk lebar kolom).

const int MAX\_SONGS = 100;

const int MAX\_USERS = 100;

* Konstanta maksimum jumlah lagu dan pengguna.

**2. Struct (Struktur Data)**

struct Song {

    string title;

    string artist;

};

* Struct Song untuk menyimpan data lagu (judul dan artis)

struct Song {

    string title;

    string artist;

};

* Sruct User menyimpan info pengguna, termasuk isAdmin (0 = biasa, 1 =admin).

**3. Rekursif untuk Menampilkan Lagu**

void tampilkanLaguRekursif(Song songs[], string genres[], int releaseYears[], int durations[], int index, int total) {

    if (index >= total) return;

    cout << left << setw(30) << songs[index].title

         << setw(30) << songs[index].artist

         << setw(20) << genres[index]

         << setw(15) << releaseYears[index]

         << setw(10) << durations[index] << endl;

    tampilkanLaguRekursif(songs, genres, releaseYears, durations, index + 1, total);

}

* Menggunakan daftar lagu secara rekurdif
* Digunakan untuk menampilkan lagu satu per satu berdasarkan inseks.

1. **Prosedur menampilkan lagu**

void tampilkanSemuaLagu(Song songs[], string genres[], int releaseYears[], int durations[], int songCount) {

    cout << "\n=== Daftar Lagu ===" << endl;

    cout << left << setw(30) << "Judul" << setw(30) << "Artis" << setw(20) << "Genre" << setw(15) << "Tahun" << setw(10) << "Durasi" << endl;

    cout << string(105, '-') << endl;

    tampilkanLaguRekursif(songs, genres, releaseYears, durations, 0, songCount); // Gunakan rekursi

}

* Menampilkan header table lagu..
* Memanggil TampilkaLaguRekursif

1. **Prosedur menambahkan lagu**
2. void tambahLagu(Song songs[], string genres[], int releaseYears[], int durations[], int &songCount) {
3. if (songCount < MAX\_SONGS) {
4. cout << "Masukkan Judul Lagu: ";
5. cin.ignore();
6. getline(cin, songs[songCount].title);
7. cout << "Masukkan Nama Artis: ";
8. getline(cin, songs[songCount].artist);
9. cout << "Masukkan Genre: ";
10. getline(cin, genres[songCount]);
11. cout << "Masukkan Tahun Rilis: ";
12. cin >> releaseYears[songCount];
13. cout << "Masukkan Durasi (dalam detik): ";
14. cin >> durations[songCount];
15. songCount++;
16. cout << "Lagu berhasil ditambahkan!" << endl;
17. } else {
18. cout << "Playlist sudah penuh!" << endl;
19. }
20. }

* Menambahkan lagu baruke array songs, beserta genre, tahun, dan durasinya
* Cek batas maksumim dengan songCout.

1. **Prosedur Menghapus lagu**
2. void hapusLagu(Song songs[], string genres[], int releaseYears[], int durations[], int &songCount) {
3. int indexToDelete;
4. cout << "Masukkan nomor lagu yang ingin dihapus (1-" << songCount << "): ";
5. cin >> indexToDelete;
6. if (indexToDelete > 0 && indexToDelete <= songCount) {
7. for (int i = indexToDelete - 1; i < songCount - 1; i++) {
8. songs[i] = songs[i + 1];
9. genres[i] = genres[i + 1];
10. releaseYears[i] = releaseYears[i + 1];
11. durations[i] = durations[i + 1];
12. }
13. songCount--;
14. cout << "Lagu berhasil dihapus!" << endl;
15. } else {
16. cout << "Nomor lagu tidak valid!" << endl;
17. }
18. }

* Menghapus lagu berdasarkan nomor urut (input user)
* Menggeser elemen setelah lagu yang dihapus agar array tetap rapat.
* Update songCout

**7. Prosedur Mengubah lagu**

void ubahLagu(Song songs[], string genres[], int releaseYears[], int durations[], int songCount) {

    int indexToEdit;

    cout << "Masukkan nomor lagu yang ingin diubah (1-" << songCount << "): ";

    cin >> indexToEdit;

    if (indexToEdit > 0 && indexToEdit <= songCount) {

        cin.ignore();

        cout << "Masukkan Judul Baru: ";

        getline(cin, songs[indexToEdit - 1].title);

        cout << "Masukkan Nama Artis Baru: ";

        getline(cin, songs[indexToEdit - 1].artist);

        cout << "Masukkan Genre Baru: ";

        getline(cin, genres[indexToEdit - 1]);

        cout << "Masukkan Tahun Rilis Baru: ";

        cin >> releaseYears[indexToEdit - 1];

        cout << "Masukkan Durasi Baru (dalam detik): ";

        cin >> durations[indexToEdit - 1];

        cout << "Lagu berhasil diubah!" << endl;

    } else {

        cout << "Nomor lagu tidak valid!" << endl;

    }

}

* Mengubah data lagu berdasarkan nomor lagu.
* Update data langung pada array sesuai input baru dari user.

**8. fungsi Login**

bool login(User users[], int userCount, string username, string nim, int &isAdmin) {

    for (int i = 0; i < userCount; i++) {

        if (users[i].username == username && users[i].nim == nim) {

            isAdmin = users[i].isAdmin;

            return true;

        }

    }

    return false;

}

* Mengecaek apakah Username dan Nim cocok dengan yang ada di array user.
* Jika cocok isAdmin diisi dengan user dan fungsi mengembalikan true.

9**. Inisialisasi Lagu**

int releaseYears[MAX\_SONGS] = {

        2020, 1999, 2000, 2001, 1999, 2017, 2004, 2000, 2000

    };

    int durations[MAX\_SONGS] = {

        210, 180, 240, 200, 220, 210, 240, 180, 210

    };

    int songCount = 9;

* Data awal 9 lagu dimasukan langsung ke array.

**10. Registrasi Pengguna**

char registerChoice;

    cout << "Apakah Anda ingin mendaftar? (y/n): ";

    cin >> registerChoice;

    cin.ignore();

    while (registerChoice == 'y' && userCount < MAX\_USERS) {

        cout << "Masukkan Username: ";

        getline(cin, users[userCount].username);

        cout << "Masukkan NIM: ";

        getline(cin, users[userCount].nim);

        users[userCount].isAdmin = (users[userCount].username == "adeganteng" && users[userCount].nim == "109") ? 1 : 0;

        userCount++;

        cout << "Pendaftaran berhasil! Pengguna terdaftar: " << users[userCount - 1].username << endl;

        cout << "Apakah Anda ingin mendaftar pengguna lain? (y/n): ";

        cin >> registerChoice;

        cin.ignore();

    }

* Mentediakan fitur pendaftaran User.
* Jika username ‘adegantang” dan nim “109”, otomatis jadi admin.

**11. Login Pengguna**

 while (loginAttempts < MAX\_ATTEMPTS) {

        cout << "\n=== Login ===" << endl;

        cout << "Masukkan Username: ";

        getline(cin, username);

        cout << "Masukkan NIM: ";

        getline(cin, nim);

        if (login(users, userCount, username, nim, isAdmin)) {

            cout << "Login berhasil! Selamat datang, " << username << "!" << endl;

            isLoggedIn = true;

            break;

        } else {

            cout << "Username atau NIM salah. Coba lagi." << endl;

            loginAttempts++;

        }

    }

* Maksimum 3 Percobaan login.
* Jika barhasil, isloggedIn = true.

**12. Menu Interaktif**

 while (opsi != 5) {

        cout << "\n=== Menu Playlist ===" << endl;

        if (isAdmin == 1) {

            cout << "1. Tambah Lagu" << endl;

            cout << "2. Tampilkan Semua Lagu" << endl;

            cout << "3. Hapus Lagu" << endl;

            cout << "4. Ubah Lagu" << endl;

        } else {

            cout << "2. Tampilkan Semua Lagu" << endl;

        }

* Tampilan Menu sesuai role:
* Admin bisa menambah, menghapus, mengubah, dan melihat lagu.
* User bisa hanya bisa melihat lagu
* Input pilihan menu 🡪 progress berdasarkan opsi

**13. Keluar Program**

while (opsi != 5) {

        cout << "\n=== Menu Playlist ===" << endl;

        if (isAdmin == 1) {

            cout << "1. Tambah Lagu" << endl;

            cout << "2. Tampilkan Semua Lagu" << endl;

            cout << "3. Hapus Lagu" << endl;

            cout << "4. Ubah Lagu" << endl;

        } else {

            cout << "2. Tampilkan Semua Lagu" << endl;

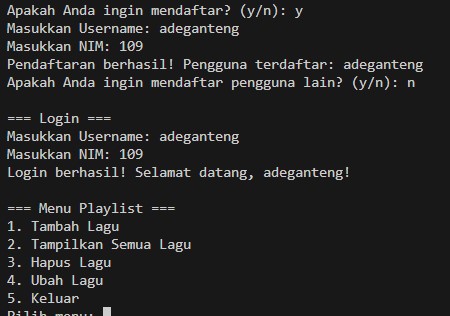
        }

* Keluar Program

# 4. Hasil Output

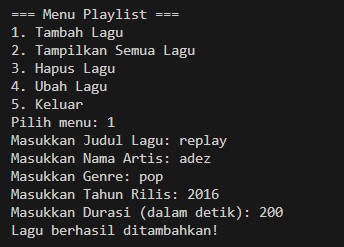
**4.1 Hasil Output**

# 1. Login Admin dan Menu Playlist



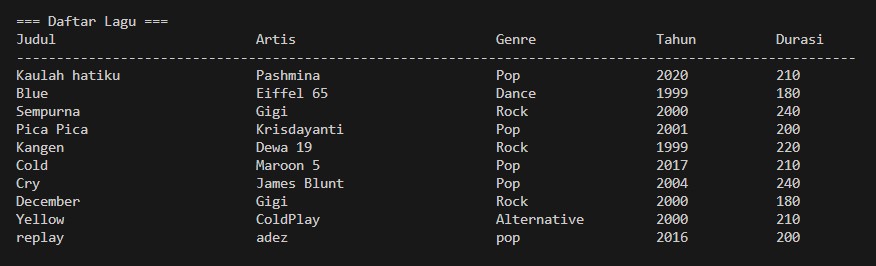
Gambar 1 Tampilan Admin dan Menu Playlist

## **2. Tambahkan Lagu**



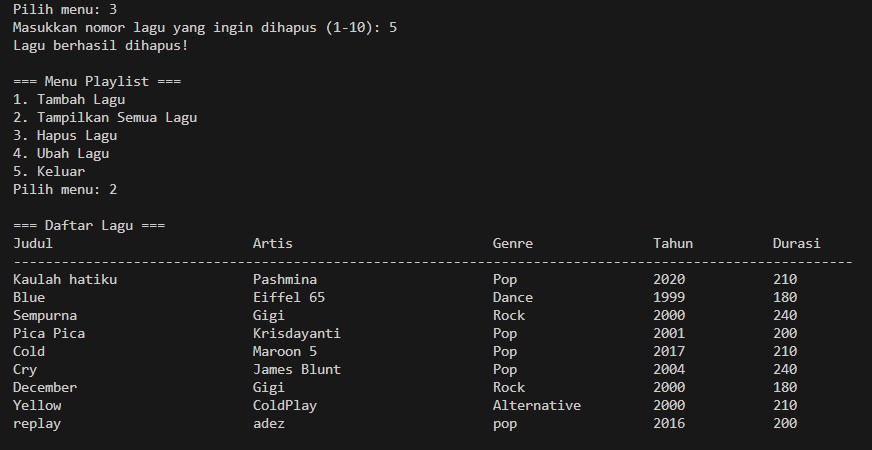
Gambar 2 Tambahkan Lagu

## **3. Tampilan semua lagu**



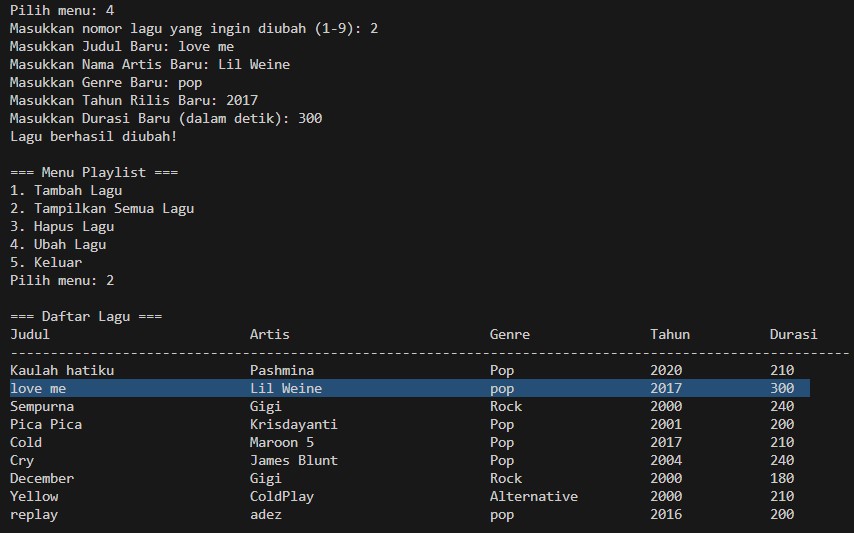
## Gambar 3 Tampilan Semua Lagu

### 4. Hapus Lagu



## Gambar 4 Tampilan Hapus Lagu

### 5. Ubah Lagu



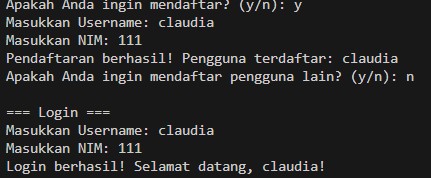
Gambar 5 Tampilan Ubah lagu

## **6.Keluar Program**



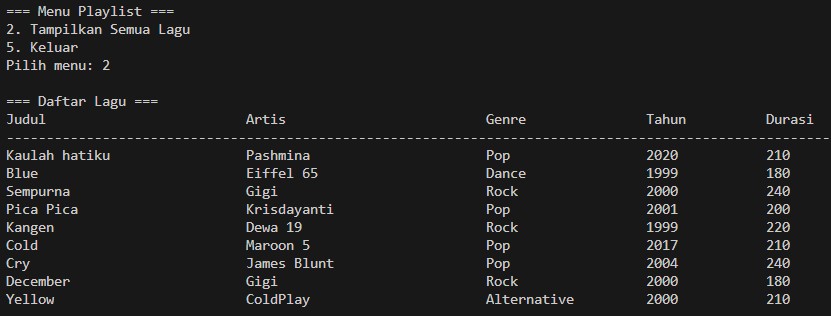
Gambar 6 Tampilan Keluar Program

### 7. Login Pengguna



Gambar 7 Tampilan Pengguna

### 8. Tampilan semua Lagu



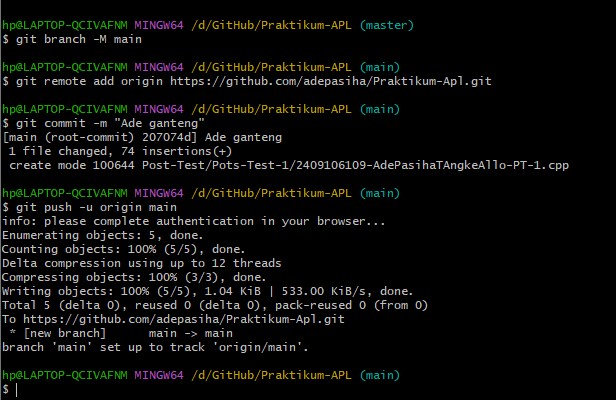
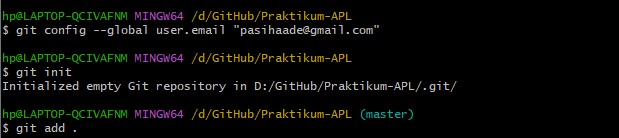
Gambar 8 Tampilan semua Lagu

9. Keluar Program



Gambar 9 Keluar dari Pragram

# 5. Git



## Gambar 5 Tampilan Git Bash

1. Mengatur Email Global Untuk Git

$ git config –global user “pasihaade@gmail.com”

-Perintah ini digunakan untuk mengatur email global Git, yang akan digunakan dalam setiap commit yang dibuat oleh pengguna.

1. Menginstal Repository Git

$ Git init

-Peintah ini menginisialisasi (membuat) Repository Git dalam folder Praktukum APL Jika sudah ada repository Git folder tersebut maka Git hanya akan menginginkan bahwa repository sudah ada

1. Menambahkan File ke Staging Area $ git add.

-Perintah ini menambahkan semua file yang ada didalam folder ke staging area

- Staging area adalah tempat sementara sebuah file dikomit kedalam reporitory

1. Menambahkan Remote Repository (Gagal Kerena Sudah ada)

$ git Remote add origin [https://Github.com/adepasiha/praltikum-apl](https://github.com/adepasiha/praltikum-apl)

-perintah ini digunakan untuk menambahkan repoditory remote dengan nama origin -Error: “remote origin alredy exists”, ini terjadi kafrena sebelumnya sudah ada repositoy remote yang sudah 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 dicommit:

* + Post-test/Post-test-1/2409106109-AdePasihaTangkeAllo-PT-2.cpp
  + Post-test/Post-test-1/2409106109-AdePasihaTangkeAllo-PT-2.exe

1. Mendorong (Push) Perubahan ke Repository Remote

$ git push -u origin main

-Perintah ini mengunggah (push) perubahan ke repository remote pada branch main

-Kerena ini adalah push pertama, flag -u figunakan untuk mengatur branch local main agar terhubung dengan branch main di remote repository.

-Proses ini terjadi:

* + Menghitung objek (Rnumering objects: 6).
  + Mengeompresi objek sebelum mengunggahnya.
  + Menulis (mengunggah) objek ke gothub.
  + Menampilkan informasi bahwa branch main sekarang dilacak oleh remote repository origin/main.