**LAPORAN PRAKTIKUM**

**POSTTEST 6**

**ALGORITMA PEMROGRAMAN LANJUT**



**Disusun oleh:**

**Ade Pasiha Tangke Allo (2409106109)**

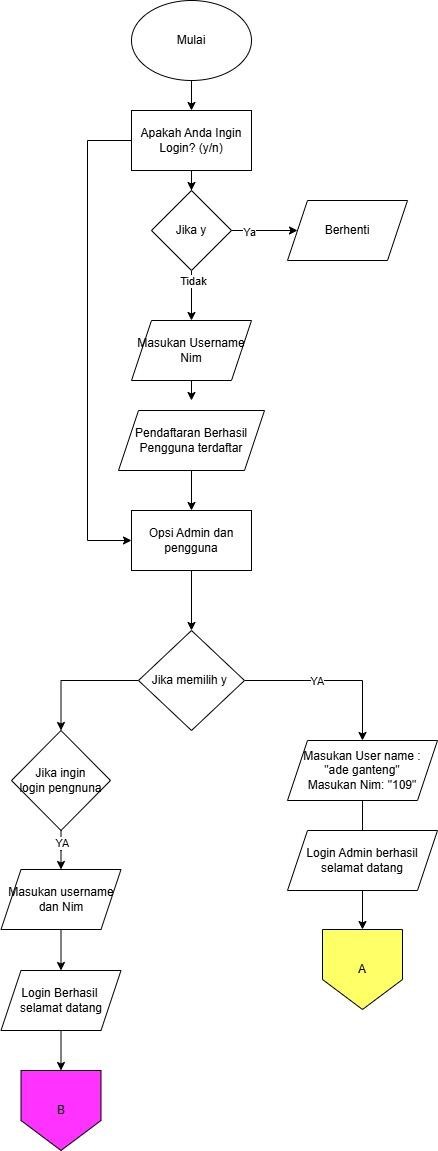
**Kelas (C2 ‘24)**

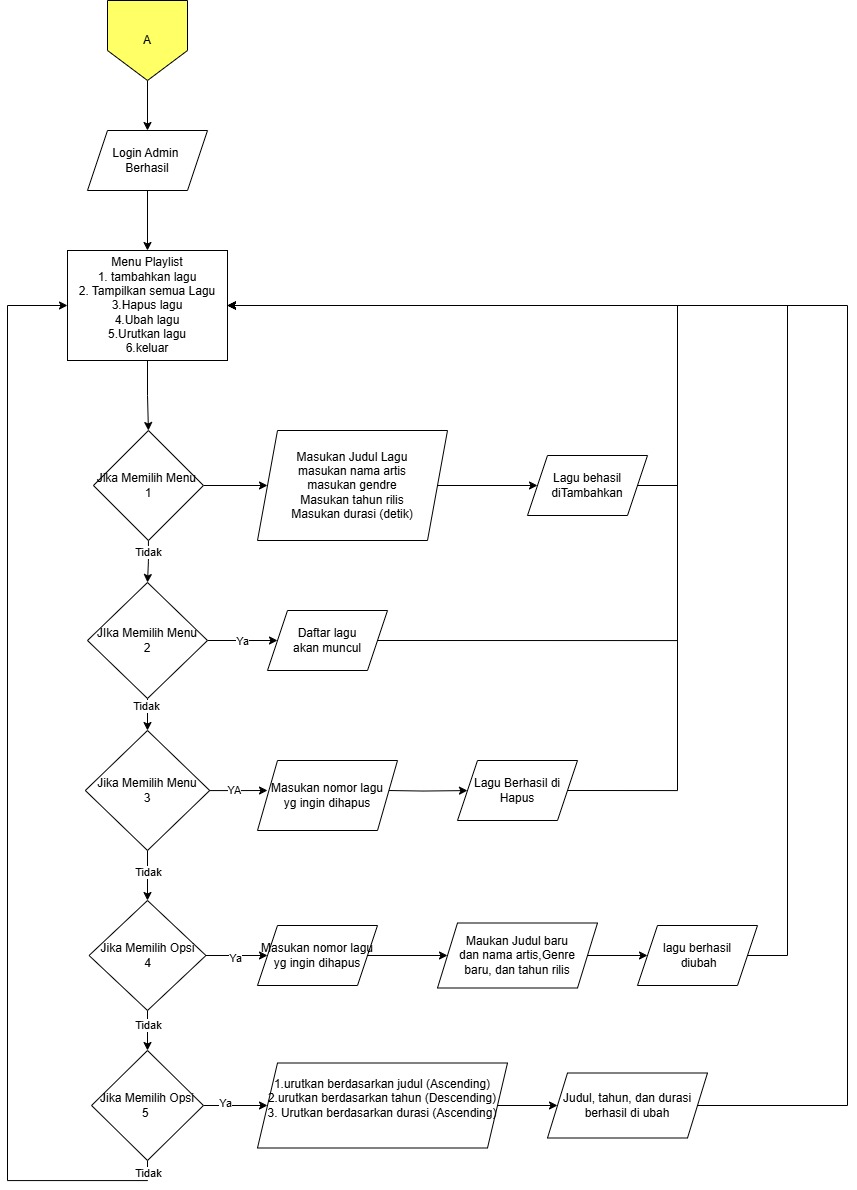
**PROGRAM STUDI INFORMATIKA**

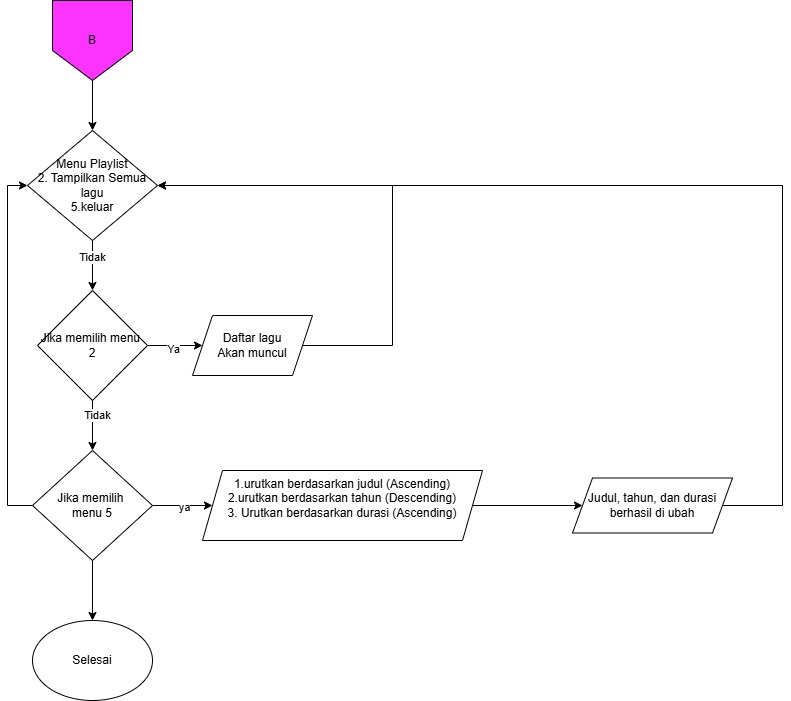
**UNIVERSITAS MULAWARMAN**

**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. **Sturct Program**
3. struct Song {
4. string title;
5. string artist;
6. };

* Mewakili data lagu: Judul dan Astis.

struct User {

    string username;

    string nim;

    int isAdmin;

};

* Mewakili data user dengan:
* Username
* Nim (sebagai password atau ID)
* isAdmin (1 untuk admin, 0 untuk user biasa)

1. **Fungsi dan Prosedur**
2. **TampilkanLaguRekursif**
3. void tampilkanLaguRekursif(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int index, int total) {
4. if (index >= total) return;
5. cout << left << setw(30) << songs[index].title
6. << setw(30) << songs[index].artist
7. << setw(20) << genres[index]
8. << setw(15) << releaseYears[index]
9. << setw(10) << durations[index] << endl;
10. tampilkanLaguRekursif(songs, genres, releaseYears, durations, index + 1, total);
11. }

* Menampilkan lagu secara rekursif (dipanggil berulang dengan index+1)
* Menggunakan pointer untuk mengakses data dari array struct dan array lainnya

**2. TampilkanSemuaLagu**

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

}

* Header tabel + garis pemisah
* Panggil tampilkanLaguRekursif() untuk menampilkan semua lagu satu per satu

**3. tambahlagu**

void tambahLagu(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int\* songCount) {

    if (\*songCount < MAX\_SONGS) {

        cout << "Masukkan Judul Lagu: ";

        cin.ignore();

        getline(cin, songs[\*songCount].title);

        cout << "Masukkan Nama Artis: ";

        getline(cin, songs[\*songCount].artist);

        cout << "Masukkan Genre: ";

        getline(cin, genres[\*songCount]);

        cout << "Masukkan Tahun Rilis: ";

        cin >> releaseYears[\*songCount];

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

        cin >> durations[\*songCount];

        \*songCount = \*songCount + 1;

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

    } else {

        cout << "Playlist sudah penuh!!!" << endl;

    }

}

* Input data Baru dari pengguna
* Disimpan dari Index terakhir ( songCount )
* Menambahkan Jumlah lagu dengan pointer \*songCount += 1
* Cek apakah Playlist penuh ( MAX\_SONGS)

**4.HapusLagu**

void hapusLagu(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int\* songCount) {

    int indexToDelete;

    cout << "Masukkan nomor lagu yang ingin dihapus (1-" << \*songCount << "): ";

    cin >> indexToDelete;

    if (indexToDelete > 0 && indexToDelete <= \*songCount) {

        for (int i = indexToDelete - 1; i < \*songCount - 1; i++) {

            songs[i] = songs[i + 1];

            genres[i] = genres[i + 1];

            releaseYears[i] = releaseYears[i + 1];

            durations[i] = durations[i + 1];

        }

        \*songCount = \*songCount - 1;

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

    } else {

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

    }

}

* Menghapus lagu berdasarkan nomor lagu yang dihapus kedepan (1-based index
* Geser semua data setelah lagu yang dihapus kedepan
* Kurangi jumlah lagu \*songCount -= 1

1. ubahLagu
2. void ubahLagu(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int songCount) {
3. int indexToEdit;
4. cout << "Masukkan nomor lagu yang ingin diubah (1-" << songCount << "): ";
5. cin >> indexToEdit;
6. if (indexToEdit > 0 && indexToEdit <= songCount) {
7. cin.ignore();
8. cout << "Masukkan Judul Baru: ";
9. getline(cin, songs[indexToEdit - 1].title);
10. cout << "Masukkan Nama Artis Baru: ";
11. getline(cin, songs[indexToEdit - 1].artist);
12. cout << "Masukkan Genre Baru: ";
13. getline(cin, genres[indexToEdit - 1]);
14. cout << "Masukkan Tahun Rilis Baru: ";
15. cin >> releaseYears[indexToEdit - 1];
16. cout << "Masukkan Durasi Baru (dalam detik): ";
17. cin >> durations[indexToEdit - 1];
18. cout << "Lagu berhasil diubah!" << endl;
19. } else {
20. cout << "Nomor lagu tidak valid!" << endl;
21. }
22. }

* Edit data lagu tertentu Berdasarkan nomor
* Tidak pakai pointer Untuk songCount karena hanya membaca

1. **Login**
2. int login(User\* users, int userCount, string username, string nim, int\* isAdmin) {
3. int i;
4. for (i = 0; i < userCount; i = i + 1) {
5. if (users[i].username == username && users[i].nim == nim) {
6. \*isAdmin = users[i].isAdmin;
7. return 1; // berhasil login
8. }
9. }
10. return 0; // gagal login
11. }

* Cek Username dana Nim
* Jika cocok, set\*Isadmin dari data user
* Return 1 (berhasil )atau 0 (gagal)

1. **Fungsi**
2. Inisialisasi Data
3. int main() {
4. Song songs[MAX\_SONGS] = {
5. {"Kaulah hatiku", "Pashmina"},
6. {"Blue", "Eiffel 65"},
7. {"Sempurna", "Gigi"},
8. {"Pica Pica", "Krisdayanti"},
9. {"Kangen", "Dewa 19"},
10. {"Cold", "Maroon 5"},
11. {"Cry", "James Blunt"},
12. {"December", "Gigi"},
13. {"Yellow", "ColdPlay"},
14. };
15. string genres[MAX\_SONGS] = {
16. "Pop", "Dance", "Rock", "Pop", "Rock", "Pop", "Pop", "Rock", "Alternative"
17. };
18. int releaseYears[MAX\_SONGS] = {
19. 2020, 1999, 2000, 2001, 1999, 2017, 2004, 2000, 2000
20. };
21. int durations[MAX\_SONGS] = {
22. 210, 180, 240, 200, 220, 210, 240, 180, 210
23. };

* Data awal playlist lagu sudah tersedia
* Total lagu awal: songCount = 9

1. **Pendaftaran User**
2. while (registerChoice == 'y' && userCount < MAX\_USERS) {
3. cout << "Masukkan Username: ";
4. getline(cin, users[userCount].username);
5. cout << "Masukkan NIM: ";
6. getline(cin, users[userCount].nim);
7. if (users[userCount].username == "adeganteng" && users[userCount].nim == "109") {
8. users[userCount].isAdmin = 1;
9. } else {
10. users[userCount].isAdmin = 0;
11. }
12. userCount = userCount + 1;
13. cout << "Pendaftaran berhasil! Pengguna terdaftar: " << users[userCount - 1].username << endl;
14. cout << "Apakah Anda ingin mendaftar pengguna lain? (y/n): ";
15. cin >> registerChoice;
16. cin.ignore();
17. }

* User bisa mendaftar (maks MAX\_USERS)
* Jika username adeganteng dan nim: 109, akan di anggab admin
* Selain itu dianggab User biasa

1. **Login**
2. while (loginAttempts < MAX\_ATTEMPTS) {
3. cout << "\n=== Login ===" << endl;
4. cout << "Masukkan Username: ";
5. getline(cin, username);
6. cout << "Masukkan NIM: ";
7. getline(cin, nim);
8. if (login(users, userCount, username, nim, &isAdmin) == 1) {
9. cout << "Login berhasil! Selamat datang, " << username << "!" << endl;
10. isLoggedIn = 1;
11. break;
12. } else {
13. cout << "Username atau NIM salah. Coba lagi." << endl;
14. loginAttempts = loginAttempts + 1;
15. }
16. }
17. if (isLoggedIn == 0) {
18. cout << "Terlalu banyak percobaan. Program dihentikan." << endl;
19. return 0;
20. }

* Max tiga kali Login
* Setelah berhasil login, disimpan isAdmin untuk control hak akses

**4. Menu Playlist**

 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;

        }

        cout << "5. Keluar" << endl;

        cout << "Pilih menu: ";

        cin >> opsi;

* Admin: bisa tambah, ubah, hapus, lihat lagu
* User bisa: hanya bisa melihat lagu
* Validasi akses berdasarkan isAdmin
* Keluar jika memilih opsi 5

1. **Bubble Sort: berdasarkan judul (Ascending)**
2. void bubbleSortTitleAsc(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int songCount) {
3. for (int i = 0; i < songCount - 1; i++) {
4. for (int j = 0; j < songCount - i - 1; j++) {
5. if (songs[j].title > songs[j + 1].title) {
6. swapSongs(songs, genres, releaseYears, durations, j, 1);
7. }
8. }
9. }
10. }

* Mengurutkan lagu bersadarkan Songs[i]. Title (judul lagu) dari A-Z (Ascending)

 if (songs[j].title > songs[j + 1].title) {

                swapSongs(songs, genres, releaseYears, durations, j, j + 1);

* Dibandingkan dua judul lagu yang berurutan
* Jika title [j] lebih besar dari title [j+1] (secara alfabet) maka dikukar sebagai swapSong().
* Proses ini dilakukan berulang kali sehingga lagu dengan judul terkecil naik ke atas (mirip gelembung naik-buble).

1. **Selection Sort: Berdasarkan Tahun (Descending)**

Fungsi:

void selectionSortYearDesc(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int songCount) {

    for (int i = 0; i < songCount - 1; i++) {

        int maxIdx = i;

        for (int j = i + 1; j < songCount; j++) {

            if (releaseYears[j] > releaseYears[maxIdx]) {

                maxIdx = j;

            }

        }

* Mengurutkan berdasarkan reslaseYears[i] dari baru ke lama (Descending)

Baris Penting:

   if (maxIdx != i) {

            swapSongs(songs, genres, releaseYears, durations, i, maxIdx);

        }

    }

}

* Untuk setiap posisi i, cari indeks lagu dengan tahun rilis terbesar (maxIdx) di sisa array.
* Tukar lagu di indeks i dengan lagu tahun terbesar (maxIdx).
* Ini adalah ciri khas **selection sort**, yaitu selalu memilih nilai maksimum dan menempatkannya ke depan

1. **Insertion Sort: Berdasarkan Durasi (Ascending**

**Fungsi:**

void insertionSortDurationAsc(Song\* songs, string\* genres, int\* releaseYears, int\* durations, int songCount) {

    for (int i = 1; i < songCount; i++) {

        Song keySong = songs[i];

        string keyGenre = genres[i];

        int keyYear = releaseYears[i];

        int keyDuration = durations[i];

        int j = i - 1;

* Mengurutkan berdasarkan durations[i] dari terpendek ke terpanjang (ascending).

Baris Penting:

while (j >= 0 && durations[j] > keyDuration) {

            songs[j + 1] = songs[j];

            genres[j + 1] = genres[j];

            releaseYears[j + 1] = releaseYears[j];

            durations[j + 1] = durations[j];

            j--;

        }

        songs[j + 1] = keySong;

        genres[j + 1] = keyGenre;

        releaseYears[j + 1] = keyYear;

        durations[j + 1] = keyDuration;

    }

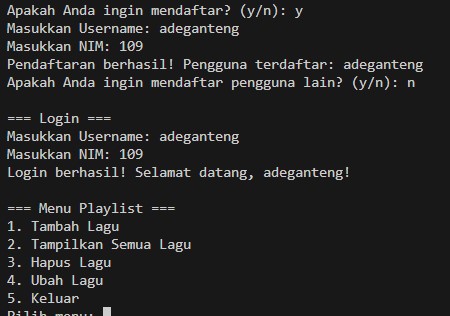
}

* Ambil lagu ke-i dan simpan sebagai "key".
* Bandingkan durasinya ke kiri, geser lagu yang durasinya lebih besar ke kanan.
* Setelah menemukan posisi yang sesuai, tempatkan key di sana.
* Proses ini mirip dengan menyisipkan kartu dalam urutan (insertion).

# 4. Hasil Output

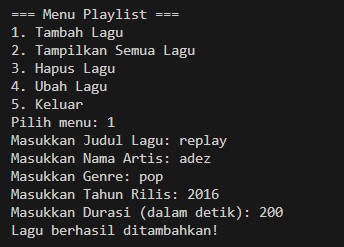
**4.1 Hasil Output**

# 1. Login Admin dan Menu Playlist



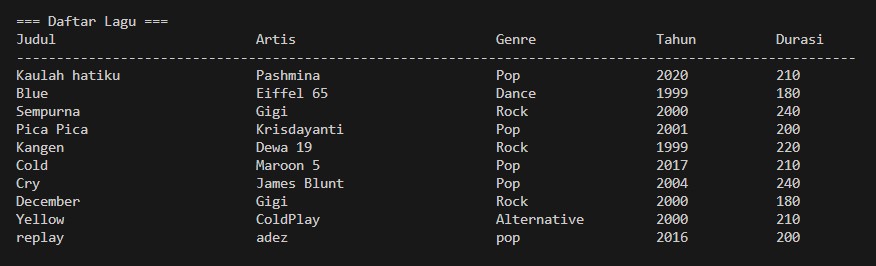
Gambar 1 Tampilan Admin dan Menu Playlist

## **2. Tambahkan Lagu**



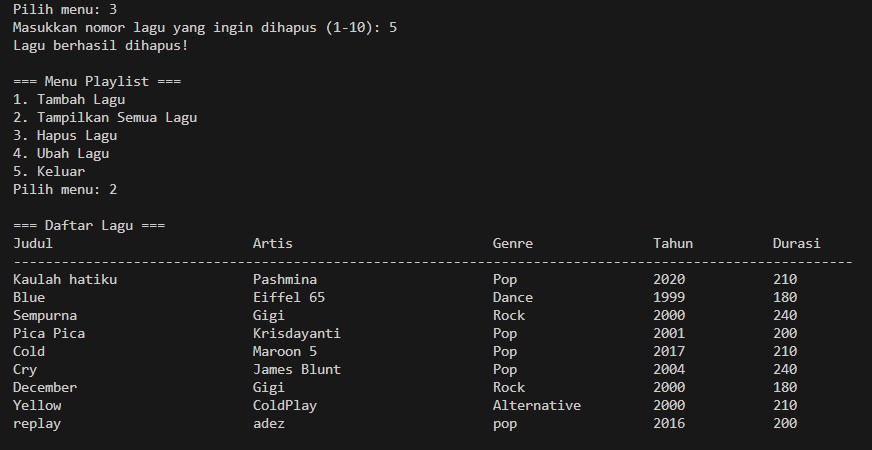
Gambar 2 Tambahkan Lagu

## **3. Tampilan semua lagu**



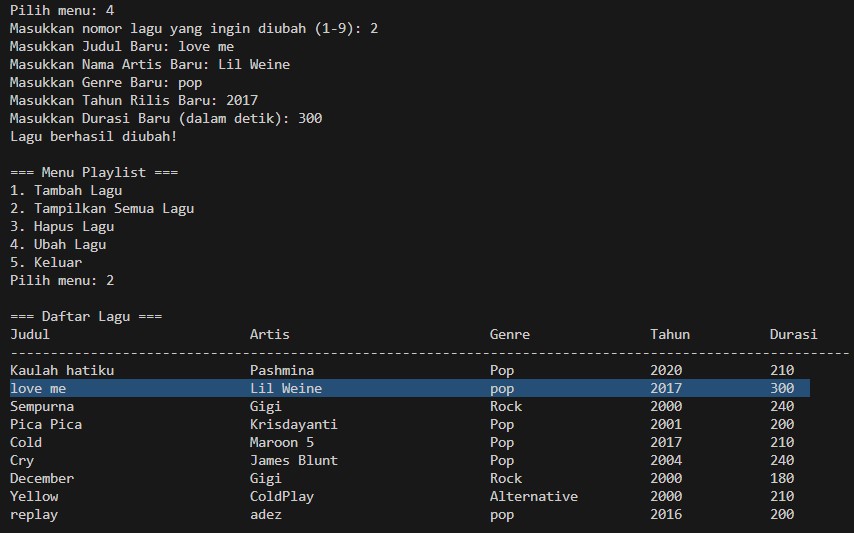
## Gambar 3 Tampilan Semua Lagu

### 4. Hapus Lagu



## Gambar 4 Tampilan Hapus Lagu

### 5. Ubah Lagu



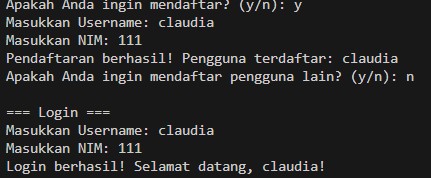
Gambar 5 Tampilan Ubah lagu

## **6.Keluar Program**



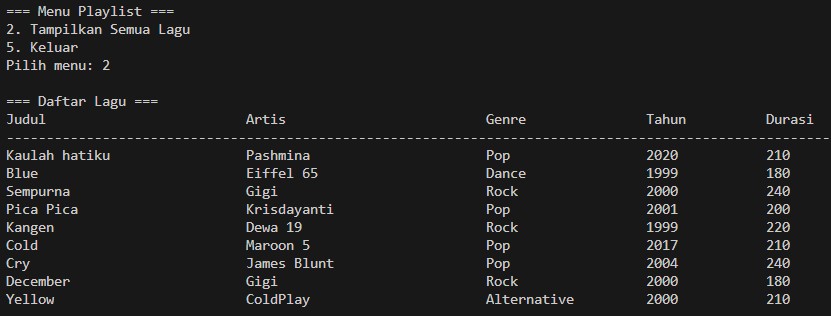
Gambar 6 Tampilan Keluar Program

### 7. Login Pengguna



Gambar 7 Tampilan Pengguna

### 8. Tampilan semua Lagu



Gambar 8 Tampilan semua Lagu

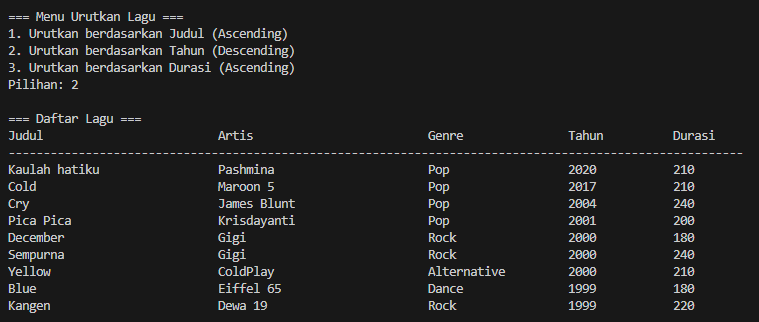
**9. Keluar Program**



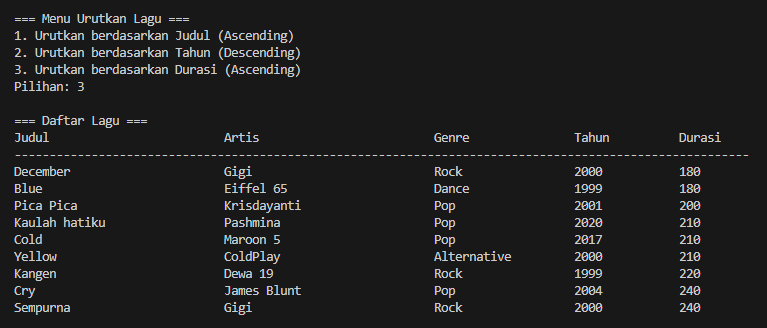
Gambar 9 Keluar dari Pragram

1. **Urutkan Lagu**

Gambar 10 urutkan judul lagu (Ascending)

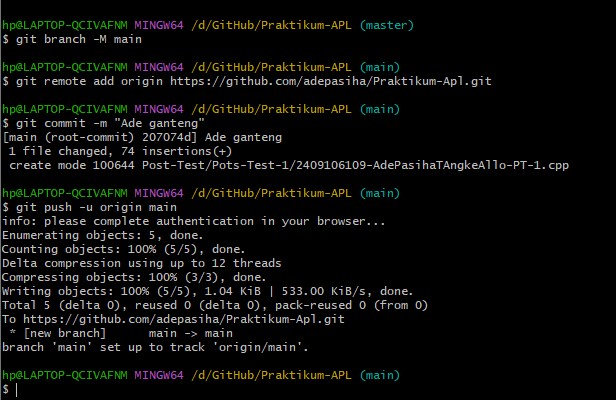
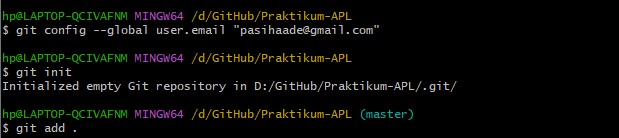


Gambar 10 Urutkan Berdasarkan tahun (Descending)



Gambar 10 Urutan berdasarkan durasi (Ascending)

# 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.