**LAPORAN PRAKTIKUM POSTTEST 6**

**ALGORITMA PEMROGRAMAN LANJUT**

****

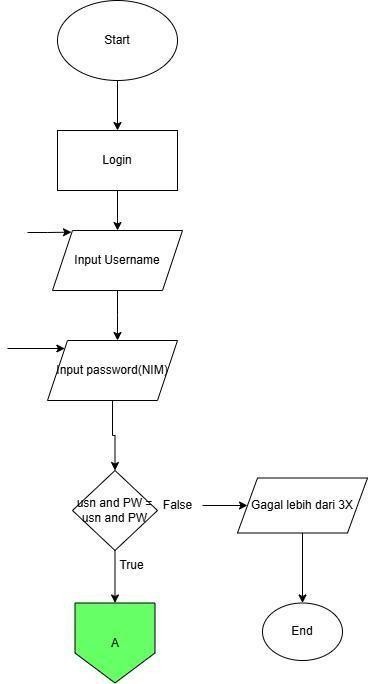
# Disusun oleh:

**Sayid Annashir Ikhwan (2409106097) Kelas (C ‘24)**

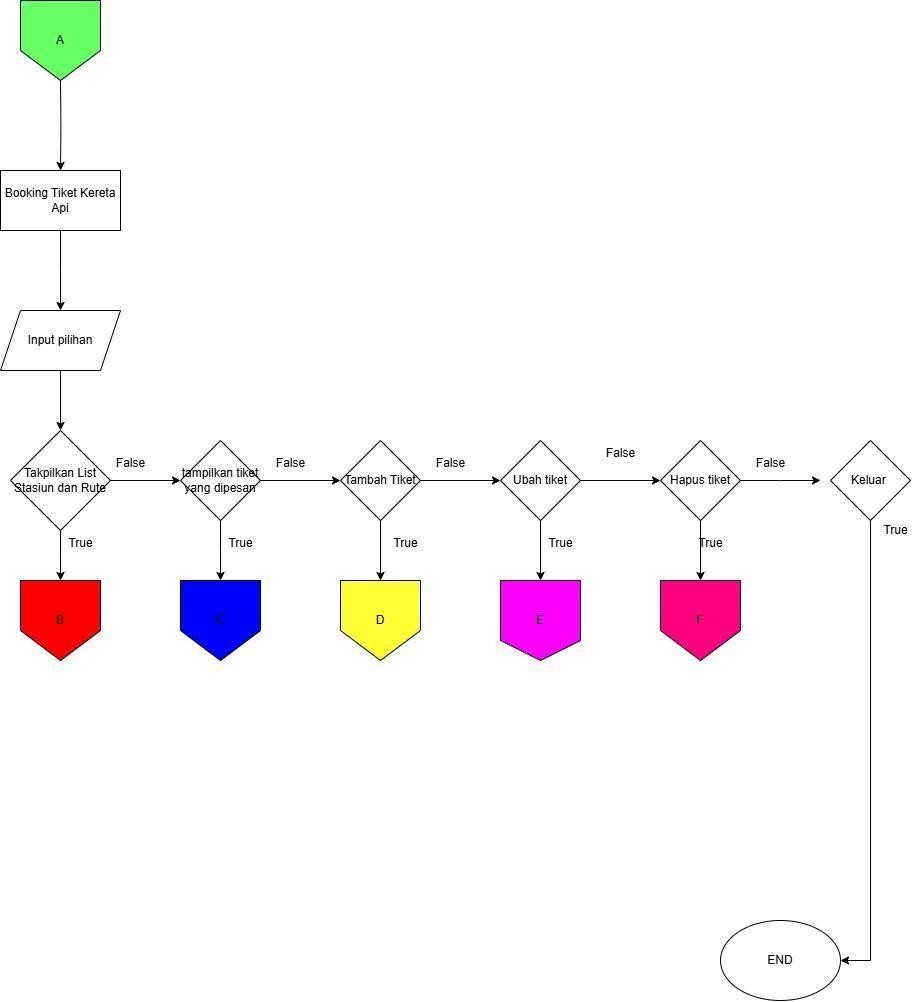
**PROGRAM STUDI INFORMATIKA UNIVERSITAS MULAWARMAN SAMARINDA**

**2025**

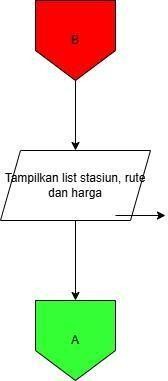
# Flowchart

****

Gambar 1.1. Flowchart



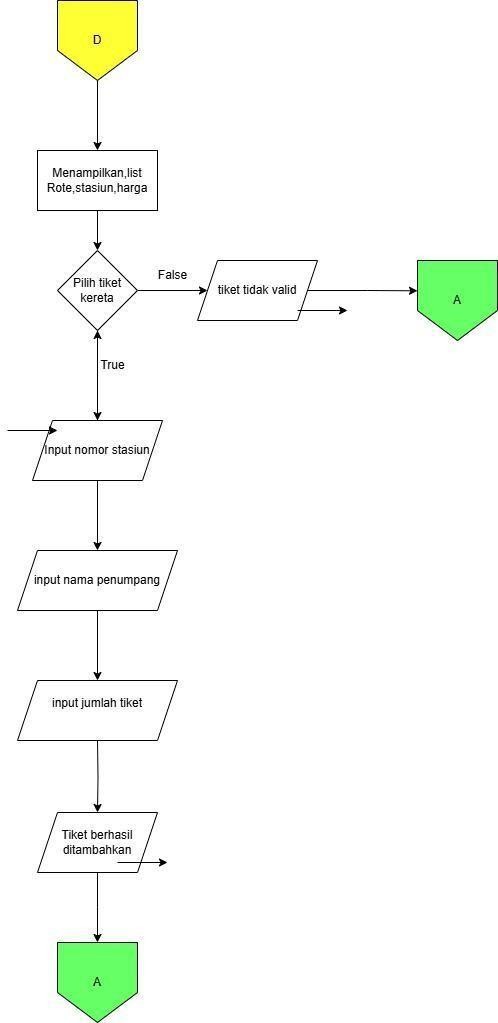
Gambar 1.2. <flowchart tampilan menu utama>



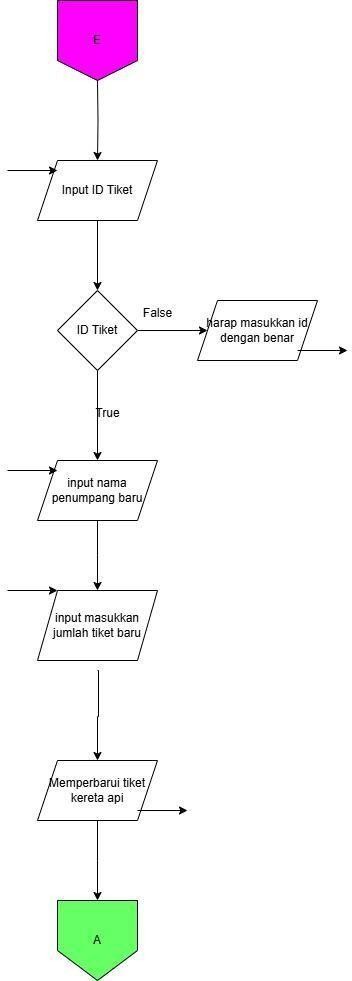
Gambar 1.3. <manampilkan list stasiun>



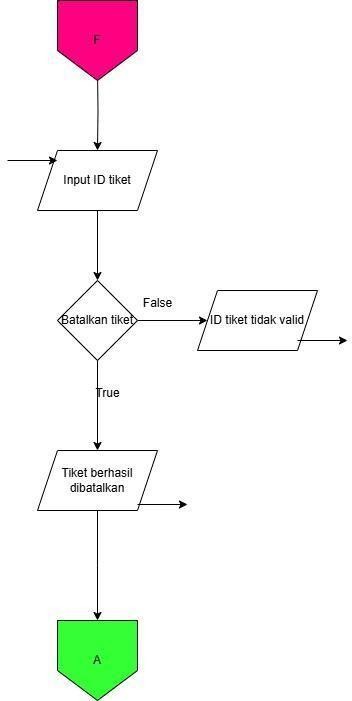
Gambar 1.4 menampilkan tiket yang sudah dipesan



Gambar 1.5 Tambah tiket



Gambar 1.6 ubah tiket



Gambar 1.7 Hapus tiket

# Analisis Program

## Deskripsi Singkat Program

Program ini adalah sebuah sistem pemesanan tiket kereta api berbasis konsol yang memungkinkan pengguna untuk melakukan pemesanan tiket, melihat daftar stasiun dan rute, serta mengelola tiket yang telah dipesan. Program ini dirancang untuk memenuhi kebutuhan pengguna dalam memesan tiket kereta api dengan antarmuka yang sederhana dan mudah digunakan. Dalam program ini, struct digunakan untuk menyimpan informasi tentang tiket yang dipesan oleh user. Kode ini dirancang untuk memenuhi tiga persyaratan utama:

menggunakan metode **sorting** huruf secara **ascending**, metode sorting angka secara **descending**, dan untuk satu metode sorting lainnya dibebaskan.

## Penjelasan Alur & Algoritma Alur Program:

1. **Login :**
   * User diminta memasukkan nama dan NIM.
   * Jika nama dan NIM sesuai, user berhasil login dan masuk ke menu utama.
   * Jika salah, user diberi 3 kesempatan. Jika gagal 3 kali, program berhenti.

## Menu Utama:

* + Setelah login berhasil, program menampilkan menu utama dengan 6 opsi:
    1. Tampilkan List Stasiun dan Rute.
    2. Tampilkan Tiket yang Dipesan.
    3. Tambah Tiket.
    4. Ubah Tiket.
    5. Hapus Tiket.
    6. Keluar.
  + Program akan berulang sampai user memilih opsi "Keluar".

## CRUD:

* + Create (Tambah Tiket): User dapat menambahkan tiket baru dengan memilih stasiun, memasukkan nama penumpang, dan jumlah tiket.
  + Read (Tampilkan Tiket): Menampilkan semua tiket yang sudah dipesan.
  + Update (Ubah Tiket): User dapat mengubah informasi tiket yang sudah dipesan.
  + Delete (Hapus Tiket): User dapat menghapus tiket yang sudah dipesan.

## Keluar:

* + Jika user memilih opsi "Keluar", program akan menampilkan pesan terima kasih dan berhenti.

# Source Code

case 7:

                sortingTiket(tiket, panjang, stasiun, rute);

                break;

Gambar 3.1 (fitur Urutkan Data Tiket)

*void* sortingTiket(Tiket\* *tiket*, *int* *panjang*, const string\* *stasiun*, const string\* *rute*) {

*int* pilihan;

    cout << "Pilihan sorting:" << endl;

    cout << "1. Nama Penumpang (Ascending)" << endl;

    cout << "2. Jumlah Tiket (Descending)" << endl;

    cout << "3. Stasiun (Ascending)" << endl;

    cout << "Masukkan pilihan: ";

    cin >> pilihan;

    cin.ignore();

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

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

*bool* tukar = false;

            if (pilihan == 1 && (*tiket* + i)->namaPenumpang > (*tiket* + j)->namaPenumpang) {

                tukar = true;

            } else if (pilihan == 2 && (*tiket* + i)->jumlahTiket < (*tiket* + j)->jumlahTiket) {

                tukar = true;

            } else if (pilihan == 3 && \*(*stasiun* + (*tiket* + i)->stasiunDipilih) > \*(*stasiun* + (*tiket* + j)->stasiunDipilih)) {

                tukar = true;

            }

            if (tukar) {

                Tiket temp = \*(*tiket* + i);

                \*(*tiket* + i) = \*(*tiket* + j);

                \*(*tiket* + j) = temp;

            }

        }

    }

    cout << "Data tiket berhasil diurutkan!" << endl;

    tampilkanTiketDipesan(*tiket*, *stasiun*, *rute*, *panjang*);

}

Gambar 3.2 (Fitur fungsi sortingTiket)

*void* tampilkanMenu() {

cout << "=============================================================" << endl;

cout << "Selamat datang di Program Booking Tiket Kereta Api" << endl;

cout << "=============================================================" << endl;

cout << "1. Tampilkan List Stasiun dan Rute" << endl; cout << "2. Tampilkan Tiket yang Dipesan" << endl; cout << "3. Tambah Tiket" << endl;

cout << "4. Ubah Tiket" << endl; cout << "5. Hapus Tiket" << endl; cout << "6. Keluar" << endl;

}

Gambar 3.3 (menu utama)

*void* tampilkanStasiunRute(const string\* *stasiun*, const string\* *rute*, const *int*\* *jumlahTiketStasiun*, const *int*\* *harga*, *int* *JUMLAH\_STASIUN*) {

    cout << "Daftar Stasiun dan Rute:" << endl;

    cout << "-----------------------------" << endl;

    for (*int* i = 0; i < *JUMLAH\_STASIUN*; i++) {

        cout << "Stasiun: " << \*(*stasiun* + i) << endl;

        cout << "Rute: " << \*(*rute* + i) << endl;

        cout << "Jumlah Tiket Tersedia: " << \*(*jumlahTiketStasiun* + i) << endl;

        cout << "Harga: Rp " << \*(*harga* + i) << endl;

        cout << "-----------------------------" << endl;

    }

}

Gambar 3.4 (daftar list stasiun dan rute)

*void* tampilkanTiketDipesan(const Tiket\* *tiket*, const string\* *stasiun*, const string\* *rute*, *int* *panjang*) {

    if (*panjang* == 0) {

        cout << "Belum ada tiket yang dipesan." << endl;

    } else {

        cout << "Daftar Tiket yang Dipesan:" << endl;

        cout << "-----------------------------" << endl;

        for (*int* i = 0; i < *panjang*; i++) {

            cout << "Tiket ke-" << i + 1 << ":" << endl;

            cout << "Nama Penumpang: " << (*tiket* + i)->namaPenumpang << endl;

            cout << "Stasiun: " << \*(*stasiun* + (*tiket* + i)->stasiunDipilih) << endl;

            cout << "Rute: " << \*(*rute* + (*tiket* + i)->stasiunDipilih) << endl;

            cout << "Jumlah Tiket: " << (*tiket* + i)->jumlahTiket << endl;

            cout << "Total Harga: Rp " << (*tiket* + i)->totalHarga << endl;

            cout << "-----------------------------" << endl;

        }

    }

}

Gambar 3.5 (tampilan tiket yang dipesan)

*void* tambahTiket(Tiket\* *tiket*, *int*\* *jumlahTiketStasiun*, const *int*\* *harga*, *int*\* *panjang*, const string\* *stasiun*, const string\* *rute*, *int* *MAX\_TIKET*, *int* *JUMLAH\_STASIUN*) {

    if (\**panjang* < *MAX\_TIKET*) {

*int* index;

        cout << "Pilih Stasiun:" << endl;

        for (*int* i = 0; i < *JUMLAH\_STASIUN*; i++) {

            cout << i + 1 << ". " << \*(*stasiun* + i) << " (" << \*(*rute* + i) << ")" << endl;

        }

        cout << "Masukkan nomor stasiun: ";

        cin >> index;

        cin.ignore();

        if (index > 0 && index <= *JUMLAH\_STASIUN*) {

            if (\*(*jumlahTiketStasiun* + index - 1) > 0) {

                cout << "Masukkan Nama Penumpang: ";

                getline(cin, (*tiket* + \**panjang*)->namaPenumpang);

                cout << "Masukkan Jumlah Tiket: ";

                cin >> (*tiket* + \**panjang*)->jumlahTiket;

                cin.ignore();

                if ((*tiket* + \**panjang*)->jumlahTiket <= \*(*jumlahTiketStasiun* + index - 1)) {

                    (*tiket* + \**panjang*)->stasiunDipilih = index - 1;

                    hitungTotalHarga(*tiket* + \**panjang*, *harga* + index - 1); // Menggunakan fungsi dereference

                    \*(*jumlahTiketStasiun* + index - 1) -= (*tiket* + \**panjang*)->jumlahTiket;

                    (\**panjang*)++;

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

                } else {

                    cout << "Jumlah tiket tidak mencukupi." << endl;

                }

            } else {

                cout << "Tiket untuk stasiun ini sudah habis." << endl;

            }

        } else {

            cout << "Nomor stasiun tidak benar. Harap masukkan yang betul." << endl;

        }

    } else {

        cout << "Kapasitas tiket penuh. Tidak bisa menambah tiket lagi." << endl;

Gambar 3.6 (tambah tiket)

*void* ubahTiket(Tiket\* *tiket*, const *int*\* *harga*, *int* *panjang*, const string\* *stasiun*, const string\* *rute*) {

    if (*panjang* == 0) {

        cout << "Belum ada tiket untuk diubah." << endl;

        return;

    }

*int* index;

    tampilkanTiketDipesan(*tiket*, *stasiun*, *rute*, *panjang*);

    cout << "Masukkan nomor tiket yang akan diubah: ";

    cin >> index;

    cin.ignore();

    if (index > 0 && index <= *panjang*) {

        cout << "Masukkan Nama Penumpang baru: ";

        getline(cin, (*tiket* + index - 1)->namaPenumpang);

        cout << "Masukkan Jumlah Tiket baru: ";

        cin >> (*tiket* + index - 1)->jumlahTiket;

        cin.ignore();

        hitungTotalHarga(*tiket* + index - 1, *harga* + (*tiket* + index - 1)->stasiunDipilih); // Menggunakan fungsi dereference

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

    } else {

        cout << "Nomor tiket tidak valid." << endl;

Gambar 3.7 (ubah tiket)

*void* hapusTiket(Tiket\* *tiket*, *int*\* *jumlahTiketStasiun*, *int*\* *panjang*, const string\* *stasiun*, const string\* *rute*) {

    if (\**panjang* == 0) {

        cout << "Belum ada tiket untuk dihapus." << endl;

        return;

    }

*int* index;

    tampilkanTiketDipesan(*tiket*, *stasiun*, *rute*, \**panjang*);

    cout << "Masukkan nomor tiket yang akan dihapus: ";

    cin >> index;

    cin.ignore();

    if (index > 0 && index <= \**panjang*) {

        \*(*jumlahTiketStasiun* + (*tiket* + index - 1)->stasiunDipilih) += (*tiket* + index - 1)->jumlahTiket;

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

            \*(*tiket* + i) = \*(*tiket* + i + 1);

        }

        (\**panjang*)--;

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

    } else {

        cout << "Nomor tiket tidak valid. Harap masukkan dengan benar." << endl;

Gambar 3.8 (Hapus tiket)

*void* hitungTotalHarga(Tiket\* *tiket*, const *int*\* *harga*) {

*tiket*->totalHarga = *tiket*->jumlahTiket \* (\**harga*);

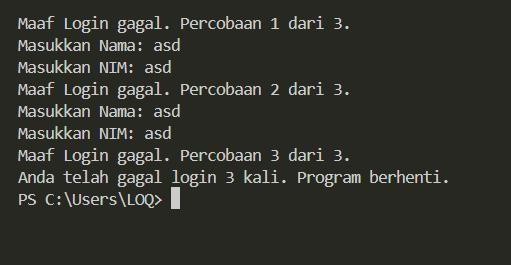
Gambar 3.9 (hitung Total Harga)

case 6:

cout << "Terima kasih sudah menggunakan program, sampai berjumpa kembali!" << endl; break;

Gambar 3.10 (logout/keluar)

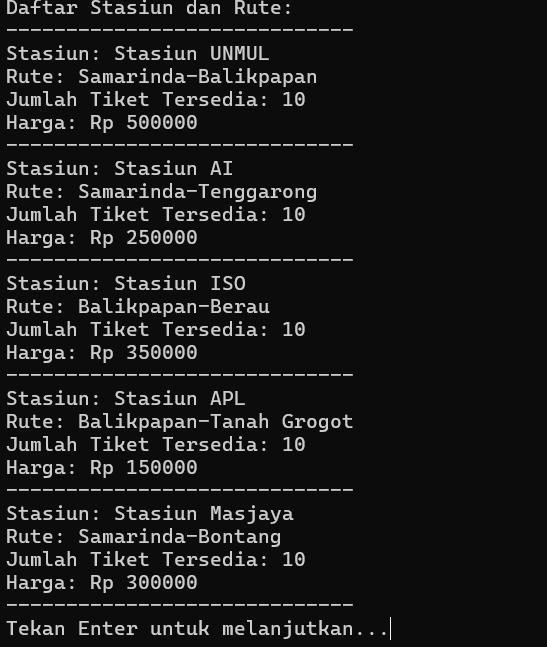
# Uji Coba dan Hasil Output

**4.1**

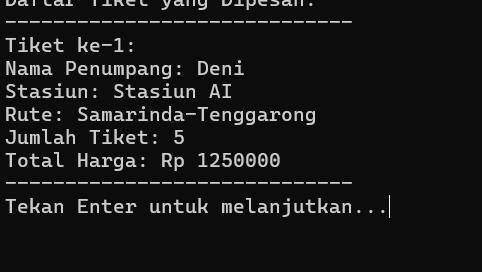
Gambar 4.1



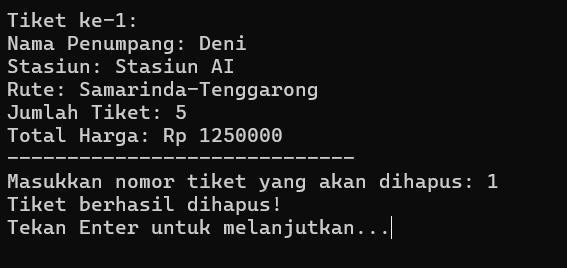
Gambar 4.2



Gambar 4.3



Gambar 4.4



Gambar 4.5

## GIT

1. **git add .**
   * Perintah ini digunakan untuk menambahkan semua perubahan (file yang baru dibuat, diubah, atau dihapus) ke dalam staging area.
   * Titik (`.`) menunjukkan bahwa semua file yang ada dalam direktori kerja akan ditambahkan.

## git commit -m "Posttest 6"

* + Perintah ini membuat commit dengan pesan `"Posttest 6"`, yang mendokumentasikan perubahan yang telah

ditambahkan ke staging area.

* + Commit ini menyimpan snapshot dari perubahan dalam repository lokal.

## git push

* + Perintah ini mengunggah commit yang telah dibuat ke repository di GitHub pada branch `main`.
  + `-u` (atau `--set-upstream`) digunakan untuk menghubungkan branch lokal `main` dengan branch `main` di repository remote (`origin`).
  + `origin` adalah alias default untuk repository remote yang dikloning atau diatur sebelumnya.