LAPORAN AKHIR PRAKTIKUM
ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

Oleh:

Afwan Ayasyin/124240193

PROGRAM STUDI SISTEM INFORMASI

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2025

HALAMAN PENGESAHAN

LAPORAN AKHIR

Disusun oleh:

Afwan Ayasyin 124240193

Telah Diperiksa dan Disetujui oleh Asisten Praktikum

Algoritma Pemrograman dan Struktur Data

Pada Tanggal: 31 MEI 2025

Menyetujui,

Asisten Praktikum	**Asisten Praktikum**
Gradiva Arya Wicaksana	**Athaya Rizqia Fitriani**
123230089	**124210071**

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa yang senantiasa mencurahkan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan praktikum Dasar-Dasar Pemrograman serta laporan akhir praktikum Algoritma Pemrograman dan Struktur Data. Adapun laporan ini berisi tentang kumpulan tugas dan evaluasi dari hasil pembelajaran selama praktikum berlangsung.

Tidak lupa ucapan terima kasih kepada asisten dosen yang selalu membimbing dan mengajari kami dalam melaksanakan praktikum dan dalam menyusun laporan ini. Laporan ini masih sangat jauh dari kesempurnaan, oleh karena itu kritik serta saran yang membangun kami harapkan untuk menyempurnakan laporan akhir ini.

Atas perhatian dari semua pihak yang membantu penulisan ini, kami ucapkan terima kasih. Semoga laporan ini dapat dipergunakan seperlunya.

Yogyakarta, 29 Mei 2025

Penyusun

DAFTAR ISI

BAB I
SORTING

Sistem manajemen data Bioskop Double A dirancang untuk memudahkan bagian ticketing dalam mengelola data film yang terdiri dari judul, kode, dan rating. Data film dapat ditampilkan menggunakan pointer, dicari berdasarkan kode dengan sequential search, serta dicari berdasarkan judul dengan binary search setelah data diurutkan. Untuk pengurutan rating, sistem menyediakan quick sort untuk urutan naik (ascending) dan bubble sort untuk urutan turun (descending). Fitur-fitur ini membantu ticketing dalam pencarian dan pengurutan film secara efisien.

1.1 *Source Code* Program

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

struct Film
{
string judul;
string kode;
double rating;
};

void tampilkanFilm(Film films[], int n)
{
cout << setfill('=') << setw(40) << "=" << endl;
cout << "Judul            Kode    Rating\n";
cout << setfill('=') << setw(40) << "=" << endl;
for (Film *ptr = films; ptr < films + n; ptr++)
{
cout << ptr->judul << "\t\t" << ptr->kode << "\t" << ptr-
>rating << endl;
}
cout << setfill('=') << setw(40) << "=" << endl;
}

int cariFilmByKode(Film films[], int n, string kode)
{
for (int i = 0; i < n; i++)
{
if (films[i].kode == kode)
{
return i;
}
}
return -1;
```

```
}

int binarySearchByJudul(Film films[], int left, int right,
string judul)
{
while (left <= right)
{
int mid = left + (right - left) / 2;
if (films[mid].judul == judul)
return mid;
if (films[mid].judul < judul)
left = mid + 1;
else
right = mid - 1;
}
return -1;
}

void quickSort(Film films[], int low, int high)
{
if (low < high)
{
double pivot = films[high].rating;
int i = low - 1;
for (int j = low; j < high; j++)
{
if (films[j].rating < pivot)
{
i++;
swap(films[i], films[j]);
}
}
swap(films[i + 1], films[high]);
int pi = i + 1;

quickSort(films, low, pi - 1);
quickSort(films, pi + 1, high);
}
}

void bubbleSort(Film films[], int n)
{
for (int i = 0; i < n - 1; i++)
{
for (int j = 0; j < n - i - 1; j++)
{
if (films[j].rating < films[j + 1].rating)
{
swap(films[j], films[j + 1]);
}
}
}
```

```
}

void sortByJudul(Film films[], int n)
{
for (int i = 0; i < n - 1; i++)
{
for (int j = 0; j < n - i - 1; j++)
{
if (films[j].judul > films[j + 1].judul)
{
swap(films[j], films[j + 1]);
}
}
}
}

int main()
{
system("cls");
Film films[5] = {
{"Avengers", "F001", 8.50},
{"Titanic", "F002", 9.00},
{"Inception", "F003", 8.80},
{"Interstellar", "F004", 9.20},
{"Joker", "F005", 8.70}};

int pilihan;
do
{
cout << setfill('=') << setw(40) << "=" << endl;
cout << setfill(' ') << setw(25) << "Bioskop Double A" << "
" << endl;
cout << setfill('=') << setw(40) << "=" << endl;
cout << "1. Tampilkan Film\n";
cout << "2. Cari film berdasarkan Kode\n";
cout << "3. Cari film berdasarkan Judul\n";
cout << "4. Urutkan film berdasarkan Rating (Asc)\n";
cout << "5. Urutkan film berdasarkan Rating (Desc)\n";
cout << "6. Keluar\n";
cout << "Pilih opsi: ";
cin >> pilihan;
system("cls");

if (pilihan == 1)
{
tampilkanFilm(films, 5);
}
else if (pilihan == 2)
{
string kode;
cout << "Masukkan Kode Film: ";
cin >> kode;
```

```
int idx = cariFilmByKode(films, 5, kode);
if (idx != -1)
cout << "Film ditemukan: " << films[idx].judul << " dengan
Rating: " << films[idx].rating << endl;
else
cout << "Film tidak ditemukan.\n";
}
else if (pilihan == 3)
{
sortByJudul(films, 5);
string judul;
cout << "Masukkan Judul Film: ";
cin.ignore();
getline(cin, judul);
int index = binarySearchByJudul(films, 0, 4, judul);
if (index != -1)
cout << "Film ditemukan: " << films[index].judul << "
dengan Rating: " << films[index].rating << endl;
else
cout << "Film tidak ditemukan.\n";
}
else if (pilihan == 4)
{
quickSort(films, 0, 4);
cout << "Film telah diurutkan berdasarkan Rating
(Asc).\n";
tampilkanFilm(films, 5);
}
else if (pilihan == 5)
{
bubbleSort(films, 5);
cout << "Film telah diurutkan berdasarkan Rating
(Desc).\n";
tampilkanFilm(films, 5);
}
else if (pilihan == 6)
{
system("cls");
cout << "Keluar dari program.\n";
}
else
{
cout << "Pilihan tidak valid. Coba lagi.\n";
}
} while (pilihan != 6);

return 0;
}
```

1.2 Implementasi Materi

1. Struct dan Array

```
struct Film {
string judul;
string kode;
double rating;
};

Film films[5] = {
{"Avengers", "F001", 8.50},
{"Titanic", "F002", 9.00},
{"Inception", "F003", 8.80},
{"Interstellar", "F004", 9.20},
{"Joker", "F005", 8.70}
};
```

Program menggunakan struct untuk menyimpan data film yang terdiri dari judul, kode, dan rating. Array digunakan untuk menampung kumpulan data film.

2. Pointer

```
void tampilkanFilm(Film films[], int n) {
for (Film *ptr = films; ptr < films + n; ptr++) {
cout << ptr->judul << "\t\t" << ptr->kode << "\t" << ptr->rating
<< endl;
}
}
```

Pointer digunakan untuk menelusuri array films saat menampilkan daftar film. Ini memberikan fleksibilitas dalam mengakses data.

3. Pencarian Linier (Linear Search)

```
int cariFilmByKode(Film films[], int n, string kode) {
    for (int i = 0; i < n; i++) {
        if (films[i].kode == kode) {
            return i;
        }
    }
    return -1;
}
```

Pencarian linier digunakan untuk mencari film berdasarkan kode. Pendekatan ini cocok karena data tidak harus diurutkan.

4. Pencarian Biner (Binary Search)

```
int binarySearchByJudul(Film films[], int left, int right, string
judul) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (films[mid].judul == judul)
            return mid;
        if (films[mid].judul < judul)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
```

Digunakan untuk mencari film berdasarkan judul, dengan syarat data sudah diurutkan terlebih dahulu. Ini membuat pencarian lebih efisien.

5. Pengurutan Quick Sort (Rating Ascending)

```
void quickSort(Film films[], int low, int high) {
    if (low < high) {
        double pivot = films[high].rating;
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (films[j].rating < pivot) {
                i++;
                swap(films[i], films[j]);
            }
        }
        swap(films[i + 1], films[high]);
        int pi = i + 1;
        quickSort(films, low, pi - 1);
        quickSort(films, pi + 1, high);
    }
}
```

QuickSort digunakan untuk mengurutkan data film berdasarkan rating secara naik (ascending). Metode ini efisien dan cepat untuk dataset besar.

6. Pengurutan Bubble Sort (Rating Descending)

```
void bubbleSort(Film films[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (films[j].rating < films[j + 1].rating) {
                swap(films[j], films[j + 1]);
            }
        }
    }
}
```

BubbleSort digunakan untuk mengurutkan data film berdasarkan rating secara turun (descending). Meskipun lebih lambat dari QuickSort, metode ini sederhana dan mudah dipahami.

7. Pengurutan Berdasarkan Judul

```
void sortByJudul(Film films[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (films[j].judul > films[j + 1].judul) {
                swap(films[j], films[j + 1]);
            }
        }
    }
}
```

Untuk dapat menggunakan pencarian biner berdasarkan judul, data harus terlebih dahulu diurutkan berdasarkan urutan alfabet judul film

8. Menu dan Interaksi Pengguna

```
int pilihan;
do {
    cout << "1. Tampilkan Film\n";
    cout << "2. Cari film berdasarkan Kode\n";
    cout << "3. Cari film berdasarkan Judul\n";
    cout << "4. Urutkan film berdasarkan Rating (Asc)\n";
    cout << "5. Urutkan film berdasarkan Rating (Desc)\n";
    cout << "6. Keluar\n";
    cout << "Pilih opsi: ";
    cin >> pilihan;

    if (pilihan == 1) {
```

```
        tampilkanFilm(films, 5);
    }
    else if (pilihan == 2) {
        string kode;
        cout << "Masukkan Kode Film: ";
        cin >> kode;
        int idx = cariFilmByKode(films, 5, kode);
    }
} while (pilihan != 6);
```

Program menyediakan menu interaktif yang memudahkan pengguna memilih fitur-fitur yang diimplementasikan. Ini menunjukkan penggunaan struktur kontrol if-else dan do-while.

1.3 Catatan Revisi

No revision

1.4 Revisi Program

```
-
```

1.5 *Screenshot* Program

1. Menu

```
========================================
         Bioskop Double A
========================================
1. Tampilkan Film
2. Cari film berdasarkan Kode
3. Cari film berdasarkan Judul
4. Urutkan film berdasarkan Rating (Asc)
5. Urutkan film berdasarkan Rating (Desc)
6. Keluar
Pilih opsi:
```

2. Tampilkan film

```
========================================
Judul            Kode    Rating
========================================
Avengers                 F001    8.5
Titanic          F002    9
Inception                F003    8.8
Interstellar             F004    9.2
Joker            F005    8.7
========================================
```

3. Cari film berdasarkan kode

```
Masukkan Kode Film: F003
Film ditemukan: Inception dengan Rating: 8.8
```

4. Cari film berdasarkan judul

```
Masukkan Judul Film: Avengers
Film ditemukan: Avengers dengan Rating: 8.5
```

5. Urutkan film berdasarkan Rating (Asc)

```
Film telah diurutkan berdasarkan Rating (Asc).
========================================
Judul            Kode    Rating
========================================
Avengers                 F001    8.5
Joker            F005    8.7
Inception                F003    8.8
Titanic          F002    9
Interstellar             F004    9.2
========================================
```

6. Urutkan film berdasarkan Rating (Desc)

```
Film telah diurutkan berdasarkan Rating (Desc).
========================================
Judul            Kode    Rating
========================================
Interstellar             F004    9.2
Titanic          F002    9
Inception                F003    8.8
Joker            F005    8.7
Avengers                 F001    8.5
========================================
```

7. Keluar

```
Keluar dari program.
PS D:\prak algo\tugasAlgo>
```

BAB II
ARRAY DAN STRING

Sistem ini dirancang untuk membantu toko roti "Manis Lezat" dalam mengatur antrean pesanan secara digital. Sistem memungkinkan penambahan pesanan baru ke antrean dengan data berupa nama pembeli, jenis roti, dan total harga. Pesanan yang paling depan dapat diproses dan dipindahkan ke riwayat pesanan yang sudah dilayani. Selain itu, sistem menyediakan fitur untuk menampilkan daftar pesanan yang sedang antre, membatalkan pesanan terakhir dalam antrean, serta menampilkan riwayat pesanan yang sudah selesai dilayani.

2.1 *Source Code* Program

```
#include <iostream>
using namespace std;

struct Pesanan
{
    char nama[50];
    char jenisRoti[50];
    int totalHarga;
    Pesanan *next;
    Pesanan *prev;
};

Pesanan *head = nullptr;
Pesanan *tail = nullptr;

void simpanRiwayat(Pesanan *p)
{
    FILE *file = fopen("riwayat.dat", "ab");
    if (!file)
    {
        cout << "Gagal membuka file riwayat!\n";
        return;
    }
    fwrite(p, sizeof(Pesanan) - 2 * sizeof(Pesanan *), 1,
file);
    fclose(file);
}

void simpanPesanan(Pesanan *p)
{
    FILE *file = fopen("pesanan.dat", "ab");
    if (!file)
    {
        cout << "Gagal membuka file pesanan!\n";
        return;
    }
    fwrite(p, sizeof(Pesanan) - 2 * sizeof(Pesanan *), 1,
file);
    fclose(file);
}
```

```
void ambilAntrean()
{
    Pesanan *baru = new Pesanan;
    cout << "Masukkan nama pelanggan: ";
    cin.ignore();
    cin.getline(baru->nama, 50);
    cout << "Jenis roti: ";
    cin.getline(baru->jenisRoti, 50);
    cout << "Total harga: ";
    cin >> baru->totalHarga;

    baru->next = nullptr;
    baru->prev = tail;

    if (!head)
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }

    simpanPesanan(baru);
    cout << "Pesanan ditambahkan ke antrean dan disimpan ke
file.\n";
}

void layaniPembeli()
{
    if (!head)
    {
        cout << "Antrean kosong.\n";
        return;
    }

    Pesanan *dilayani = head;
    cout << "Melayani: " << dilayani->nama << ", Roti: " <<
dilayani->jenisRoti << ", Harga: " << dilayani->totalHarga <<
"\n";

    simpanRiwayat(dilayani);

    head = head->next;
    if (head)
        head->prev = nullptr;
    else
        tail = nullptr;

    delete dilayani;
    cout << "Pesanan telah dilayani dan disimpan ke
riwayat.\n";
}
```

```
void tampilkanAntrean()
{
    if (!head)
    {
        cout << "Antrean kosong.\n";
        return;
    }

    Pesanan *temp = head;
    cout << "\n--- Daftar Pesanan dalam Antrean ---\n";
    while (temp)
    {
        cout << "Nama: " << temp->nama << ", Roti: " << temp-
>jenisRoti << ", Harga: " << temp->totalHarga << "\n";
        temp = temp->next;
    }
}

void batalkanPesanan()
{
    if (!tail)
    {
        cout << "Tidak ada pesanan untuk dibatalkan.\n";
        return;
    }

    Pesanan *dibatalkan = tail;
    cout << "Pesanan oleh " << dibatalkan->nama << "
dibatalkan.\n";

    tail = tail->prev;
    if (tail)
        tail->next = nullptr;
    else
        head = nullptr;

    delete dibatalkan;
}

void tampilkanRiwayat()
{
    FILE *file = fopen("riwayat.dat", "rb");
    if (!file)
    {
        cout << "Belum ada riwayat tersimpan.\n";
        return;
    }

    Pesanan temp;
    cout << "\n--- Riwayat Pesanan ---\n";
    while (fread(&temp, sizeof(Pesanan) - 2 * sizeof(Pesanan
*), 1, file))
    {
        cout << "Nama: " << temp.nama << ", Roti: " <<
temp.jenisRoti << ", Harga: " << temp.totalHarga << "\n";
```

```
    }
    fclose(file);
}

int main()
{
    int pilihan;
    do
    {
        cout << "\n=== Sistem Antrian Toko Roti 'Manis Lezat'
===\n";
        cout << "1. Ambil Antrean\n";
        cout << "2. Layani Pembeli\n";
        cout << "3. Tampilkan Antrean\n";
        cout << "4. Batalkan Pesanan Terakhir\n";
        cout << "5. Tampilkan Riwayat Pesanan\n";
        cout << "0. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            ambilAntrean();
            break;
        case 2:
            layaniPembeli();
            break;
        case 3:
            tampilkanAntrean();
            break;
        case 4:
            batalkanPesanan();
            break;
        case 5:
            tampilkanRiwayat();
            break;
        case 0:
            cout << "Terima kasih!\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 0);

    return 0;
}
```

2.2 Implementasi Materi

Program ini menggunakan struct untuk mendefinisikan tipe data Pesanan, yang merepresentasikan satu data pesanan pelanggan. Struktur ini berisi informasi nama pelanggan, jenis roti, total harga pesanan, serta dua pointer next dan prev untuk membentuk double linked list.

```
struct Pesanan {
            char nama[50];
            char jenisRoti[50];
            int totalHarga;
            Pesanan *next;
            Pesanan *prev;
};
```

Dua pointer global head dan tail dideklarasikan untuk menunjuk ke awal dan akhir antrean. Inisialisasi pointer sebagai nullptr berarti antrean awalnya kosong.

```
Pesanan *head = nullptr;
Pesanan *tail = nullptr;
```

Untuk menyimpan data ke file, program menyediakan dua fungsi, yaitu simpanPesanan() dan simpanRiwayat(). Masing-masing menyimpan data pesanan ke file pesanan.dat dan riwayat.dat. Data yang disimpan hanya bagian utama struct (tanpa pointer), sehingga file menyimpan informasi pelanggan secara efisien.

```
void simpanPesanan(Pesanan *p) {
FILE *file = fopen("pesanan.dat", "ab");
if (!file) {
   cout << "Gagal membuka file pesanan!\n";
    return;
}
fwrite(p, sizeof(Pesanan) - 2 * sizeof(Pesanan *), 1, file);
fclose(file);
}
```

Selanjutnya, saat pelanggan ingin memesan, fungsi ambilAntrean() dipanggil. Program meminta input nama pelanggan, jenis roti, dan total harga. Setelah itu, node baru dibuat dan disambungkan ke belakang antrean. Jika antrean kosong, node akan menjadi head dan tail sekaligus.

```
Pesanan *baru = new Pesanan;
cin.getline(baru->nama, 50);
cin.getline(baru->jenisRoti, 50);
cin >> baru->totalHarga;
baru->next = nullptr;
```

```
baru->prev = tail;
if (!head) {
    head = tail = baru;
} else {
   tail->next = baru;
   tail = baru;
}
```

Untuk melayani pelanggan, digunakan fungsi layaniPembeli(). Program akan mengambil node paling depan (head), menampilkannya, menyimpannya ke file riwayat, lalu menghapus node tersebut dari antrean. Jika setelah itu antrean menjadi kosong, maka tail diatur ke nullptr.

```
Pesanan *dilayani = head;
head = head->next;
if (head)
      head->prev = nullptr;
else
      tail = nullptr;
      delete dilayani;
```

Antrean yang sedang berlangsung bisa ditampilkan menggunakan fungsi tampilkanAntrean(). Dengan menggunakan pointer sementara (temp), program akan mencetak seluruh isi antrean mulai dari head ke tail.

```
Pesanan *temp = head;
while (temp) {
      cout << "Nama: " << temp->nama << ", Roti: " << temp-
>jenisRoti << ", Harga: " << temp->totalHarga << "\n";
      temp = temp->next;
}
```

Fungsi batalkanPesanan() disediakan jika pelanggan terakhir ingin membatalkan pesanan. Node terakhir (tail) akan dihapus dan pointer tail akan dipindahkan ke node sebelumnya. Jika antrean menjadi kosong, maka head juga akan diset ke nullptr.

```
Pesanan *dibatalkan = tail;
tail = tail->prev;
if (tail)
      tail->next = nullptr;
else
      head = nullptr;
      delete dibatalkan;
```

Untuk menampilkan semua pesanan yang sudah dilayani, digunakan fungsi tampilkanRiwayat(). Program akan membaca file riwayat.dat dan mencetak isinya satu per satu hingga selesai.

```
while (fread(&temp, sizeof(Pesanan) - 2 * sizeof(Pesanan *), 1, file)) {
      cout << "Nama: " << temp.nama << ", Roti: " << temp.jenisRoti << ", Harga: " << temp.totalHarga << "\n";
}
```

Semua fitur di atas dikendalikan dari fungsi main(), yang menyediakan tampilan menu interaktif untuk memilih aksi seperti mengambil antrean, melayani, membatalkan, atau melihat riwayat. Program akan terus berjalan dalam loop selama pengguna belum memilih keluar.

```
do {
      cout << "1. Ambil Antrean\n";
      cout << "2. Layani Pembeli\n";
      …
      cin >> pilihan;
      switch (pilihan) {
      case 1: ambilAntrean(); break;
      ...
      }
} while (pilihan != 0);
```

2.3 Catatan Revisi

wuidih pake file segala nih, good job:)

2.4 Revisi Program

```
-
```

2.5 *Screenshot* Program

1. Menu

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu:
```

2. Ambil antrean

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu: 1
Masukkan nama pelanggan: shiin
Jenis roti: croissant
Total harga: 11600
Pesanan ditambahkan ke antrean dan disimpan ke file.
```

3. Layani pembeli

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu: 2
Melayani: shiin, Roti: croissant, Harga: 11600
Pesanan telah dilayani dan disimpan ke riwayat.
```

4. Tampilkan antrean

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu: 3

--- Daftar Pesanan dalam Antrean ---
Nama: shiin, Roti: croissant, Harga: 11600
```

5. Batalkan pesanan terakhir

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu: 4
Tidak ada pesanan untuk dibatalkan.
```

6. Tampilkan riwayat pesanan

```
=== Sistem Antrian Toko Roti 'Manis Lezat' ===
1. Ambil Antrean
2. Layani Pembeli
3. Tampilkan Antrean
4. Batalkan Pesanan Terakhir
5. Tampilkan Riwayat Pesanan
0. Keluar
Pilih menu: 5

--- Riwayat Pesanan ---
Nama: shiin, Roti: croissant, Harga: 11600
Nama: shiin, Roti: croissant, Harga: 11600
```

7. Keluar

```
Terima kasih!
PS D:\prak algo\tugasAlgo>
```

BAB III
PROYEK AKHIR

Proyek akhir ini merupakan pengembangan sebuah program sistem informasi klinik yang dirancang untuk mendukung kegiatan operasional di lingkungan pelayanan kesehatan. Program ini bertujuan untuk mempermudah proses administrasi, meningkatkan efisiensi pelayanan, dan mengurangi kesalahan pencatatan yang sering terjadi pada sistem manual. Program ini mencakup fitur-fitur utama seperti pendaftaran pasien, pemeriksaan medis, tampilan antrean pasien, pencarian data pasien dan urutan pasien. Dengan adanya sistem ini, diharapkan pihak klinik dapat mengelola data pasien dan operasional harian secara lebih terstruktur, cepat, dan akurat. Dikembangkan menggunakan teknologi yang sesuai dengan kebutuhan skala klinik, program ini dirancang dengan antarmuka yang sederhana dan ramah pengguna sehingga dapat dioperasikan oleh staf administrasi tanpa memerlukan pelatihan teknis yang mendalam. Proyek ini juga menjadi bentuk penerapan nyata dari ilmu yang telah diperoleh selama masa studi, serta sebagai kontribusi nyata dalam mendukung digitalisasi layanan kesehatan di tingkat masyarakat.

3.1 Dasar Teori

Dalam merancang dan mengembangkan program layanan kesehatan untuk klinik, penggunaan struktur data dan algoritma tertentu sangat berperan penting dalam memastikan program berjalan secara efisien dan terorganisir. Salah satu struktur data yang digunakan adalah *linked list* ganda *(double linked list)*. Struktur ini memungkinkan setiap elemen data (node) saling terhubung dua arah, yaitu ke elemen sebelumnya dan sesudahnya. Dengan cara ini, proses penelusuran data, baik ke depan maupun ke belakang, menjadi lebih fleksibel. Dalam konteks klinik, ini sangat bermanfaat, misalnya untuk melihat riwayat kunjungan pasien secara berurutan berdasarkan waktu. Program ini juga memanfaatkan konsep *linked list* dengan kepala *(head)* dan ekor *(tail).* Node kepala digunakan untuk menunjuk ke data pertama, sedangkan node ekor menunjuk ke data terakhir. Pendekatan ini sangat cocok untuk mengelola antrean pasien, di mana pasien baru dimasukkan di bagian akhir *(tail)* dan pelayanan dilakukan dari bagian awal *(head)*. Hal ini menciptakan sistem antrean yang dinamis dan efisien. Untuk penyimpanan data secara permanen, program menggunakan file sebagai media utama.. Penggunaan file ini memastikan bahwa data tidak hilang meskipun program ditutup dan dapat diakses kembali saat dibutuhkan. Agar data dalam program dapat ditampilkan dengan lebih rapi dan mudah dicari, digunakan teknik *sorting* (pengurutan). Dengan data yang sudah terurut, pencarian informasi menjadi lebih cepat dan terstruktur. Selain itu, program dilengkapi dengan fitur *search* (pencarian) untuk mempermudah pengguna dalam menemukan data tertentu. Metode pencarian yang digunakan dapat disesuaikan, mulai dari pencarian linier (linear *search*) untuk data kecil

hingga pencarian biner (*binary search*) untuk data yang telah terurut. Keseluruhan proses dalam struktur data tersebut sangat bergantung pada penggunaan pointer, yaitu variabel yang menyimpan alamat memori dari variabel lain. Dalam program ini, pointer berfungsi untuk menghubungkan node dalam *linked list*, mengatur alokasi memori secara dinamis, serta mempermudah pengaksesan data dalam file. Dengan pointer, pengelolaan data menjadi lebih fleksibel dan efisien, terutama saat jumlah data bersifat dinamis dan terus bertambah. Melalui penerapan konsep-konsep tersebut, program layanan kesehatan klinik dapat memberikan kinerja yang optimal dalam menangani berbagai kebutuhan administratif dan informasi medis secara cepat, akurat, dan terstruktur.

3.2 *Source Code* Program

```
#include <iostream>
using namespace std;

struct Pasien
{
    char nama[50];
    int usia;
    char keluhan[100];
    Pasien *next;
    Pasien *prev;
};

Pasien *head = nullptr;
Pasien *tail = nullptr;

void simpanKeFile(Pasien *p)
{
    FILE *file = fopen("riwayat_pasien.dat", "ab");
    if (!file)
    {
        cout << "Gagal membuka file.\n";
        return;
    }
    fwrite(p, sizeof(Pasien) - 2 * sizeof(Pasien *), 1, file);
    fclose(file);
}

void tambahPasien()
{
    Pasien *baru = new Pasien;
    cout << "Nama: ";
    cin.ignore();
    cin.getline(baru->nama, 50);
    cout << "Usia: ";
    cin >> baru->usia;
    cin.ignore();
    cout << "Keluhan: ";
    cin.getline(baru->keluhan, 100);

    baru->next = nullptr;
```

```
    baru->prev = tail;

    if (!head)
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }

    cout << "Pasien berhasil ditambahkan.\n";
}

void periksaPasien()
{
    if (!head)
    {
        cout << "Tidak ada pasien dalam antrian.\n";
        return;
    }

    Pasien *p = head;
    cout << "\nMemeriksa pasien:\n";
    cout << "Nama: " << p->nama << ", Usia: " << p->usia << ",
Keluhan: " << p->keluhan << endl;

    simpanKeFile(p);

    head = head->next;
    if (head)
        head->prev = nullptr;
    else
        tail = nullptr;

    delete p;
    cout << "Pasien selesai diperiksa dan disimpan ke
riwayat.\n";
}

void tampilkanPasien()
{
    if (!head)
    {
        cout << "Antrian kosong.\n";
        return;
    }

    Pasien *temp = head;
    cout << "\n=== Daftar Pasien dalam Antrian ===\n";
    while (temp)
    {
        cout << "Nama: " << temp->nama << ", Usia: " << temp-
>usia << ", Keluhan: " << temp->keluhan << endl;
        temp = temp->next;
    }
```

```
}

void cariPasien()
{
    if (!head)
    {
        cout << "Antrian kosong.\n";
        return;
    }

    char namaCari[50];
    cout << "Masukkan nama pasien: ";
    cin.ignore();
    cin.getline(namaCari, 50);

    Pasien *temp = head;
    while (temp)
    {
        bool cocok = true;
        for (int i = 0; i < 50; i++)
        {
            if (namaCari[i] != temp->nama[i])
            {
                cocok = false;
                break;
            }
            if (namaCari[i] == '\0' && temp->nama[i] == '\0')
                break;
        }
        if (cocok)
        {
            cout << "Pasien ditemukan:\n";
            cout << "Nama: " << temp->nama << ", Usia: " <<
temp->usia << ", Keluhan: " << temp->keluhan << endl;
            return;
        }
        temp = temp->next;
    }

    cout << "Pasien tidak ditemukan.\n";
}

void tukarNode(Pasien *a, Pasien *b)
{
    if (a == b)
        return;

    // Tukar posisi pointer a dan b
    Pasien *prevA = a->prev;
    Pasien *nextB = b->next;

    if (a->next != b)
    {
        Pasien *nextA = a->next;
        Pasien *prevB = b->prev;

        if (prevA)
```

```
            prevA->next = b;
        if (nextA)
            nextA->prev = b;
        if (prevB)
            prevB->next = a;
        if (nextB)
            nextB->prev = a;

        a->next = nextB;
        a->prev = prevB;
        b->next = nextA;
        b->prev = prevA;
    }
    else
    {
        if (prevA)
            prevA->next = b;
        if (nextB)
            nextB->prev = a;

        a->next = nextB;
        a->prev = b;
        b->next = a;
        b->prev = prevA;
    }

    // Perbarui head dan tail jika perlu
    if (head == a)
        head = b;
    else if (head == b)
        head = a;

    if (tail == a)
        tail = b;
    else if (tail == b)
        tail = a;
}

void urutkanUsia()
{
    if (!head || !head->next)
        return;

    bool swapped;
    do
    {
        swapped = false;
        Pasien *curr = head;

        while (curr && curr->next)
        {
            if (curr->usia > curr->next->usia)
            {
                tukarNode(curr, curr->next);
                swapped = true;
                break; // restart dari head setelah tukar
            }
```

```
            else
            {
                curr = curr->next;
            }
        }
    } while (swapped);

    cout << "Antrian berhasil diurutkan berdasarkan usia.\n";
}

int main()
{
    int pilihan;
    do
    {
        cout << "\n=== Sistem Antrian Klinik ===\n";
        cout << "1. Tambah Pasien\n";
        cout << "2. Periksa Pasien\n";
        cout << "3. Tampilkan Antrian\n";
        cout << "4. Cari Pasien\n";
        cout << "5. Urutkan Berdasarkan Usia\n";
        cout << "0. Keluar\n";
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            tambahPasien();
            break;
        case 2:
            periksaPasien();
            break;
        case 3:
            tampilkanPasien();
            break;
        case 4:
            cariPasien();
            break;
        case 5:
            urutkanUsia();
            break;
        case 0:
            cout << "Terima kasih!\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 0);

    return 0;
}
```

3.3 Implementasi Materi

```
struct Pasien {
    char nama[50];
    int usia;
    char keluhan[100];
    Pasien *next; // menunjuk ke pasien berikutnya
    Pasien *prev; // menunjuk ke pasien sebelumnya
};
```

Tabel 3.3.1 Linked List Ganda (*Double Linked List*)

Linked list ganda memungkinkan navigasi dua arah antar node pasien. Ini berguna jika suatu saat program ingin menelusuri data baik dari awal ke akhir maupun sebaliknya. Hal ini juga membuat operasi seperti penghapusan, pencarian, dan pertukaran data menjadi lebih fleksibel dan efisien.

```
Pasien *head = nullptr;
Pasien *tail = nullptr;

void tambahPasien() {
    Pasien *baru = new Pasien;
    ...
    baru->prev = tail;
    baru->next = nullptr;

    if (!head) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}
```

Tabel 3.3.2 Linked List Kepala & Ekor (*Head & Tail*)

Menggunakan pointer head dan tail memungkinkan penambahan pasien baru di akhir antrian (tail) dan pemeriksaan pasien dari awal antrian (head). Ini mencerminkan mekanisme antrian (queue) seperti dalam pelayanan klinik yang berjalan dari depan ke belakang.

```
void simpanKeFile(Pasien *p) {
    FILE *file = fopen("riwayat_pasien.dat", "ab");
    ...
    fwrite(p, sizeof(Pasien) - 2 * sizeof(Pasien *), 1, file);
```

```
    fclose(file);
}
```

Tabel 3.3.3 File

File digunakan untuk menyimpan riwayat pasien yang telah diperiksa secara permanen. Hal ini penting karena data yang disimpan dalam memori akan hilang saat program ditutup. File memungkinkan data disimpan dan dibaca kembali di kemudian hari.

```
void urutkanUsia() {
    ...
    while (curr && curr->next) {
        if (curr->usia > curr->next->usia) {
            tukarNode(curr, curr->next);
            ...
        }
        ...
    }
}
```

Tabel 3.3.4 Sorting

Sorting digunakan untuk mengurutkan pasien berdasarkan usia agar dapat memberikan prioritas pelayanan. Dalam dunia nyata, pasien lansia atau anak-anak mungkin mendapat perhatian lebih cepat. Sorting dilakukan menggunakan pointer tanpa mengubah isi data pasien, hanya pertukaran posisi node.

```
void cariPasien() {
    ...
    char namaCari[50];
    cin.ignore();
    cin.getline(namaCari, 50);

    Pasien *temp = head;
    while (temp) {
        bool cocok = true;
        for (int i = 0; i < 50; i++) {
            if (namaCari[i] != temp->nama[i]) {
                cocok = false;
                break;
            }
            if (namaCari[i] == '\0' && temp->nama[i] == '\0')
                break;
        }
```

```
        if (cocok) {
            cout << "Pasien ditemukan: " << temp->nama << endl;
            return;
        }
        temp = temp->next;
    }
}
```

Tabel 3.3.5 *Search*

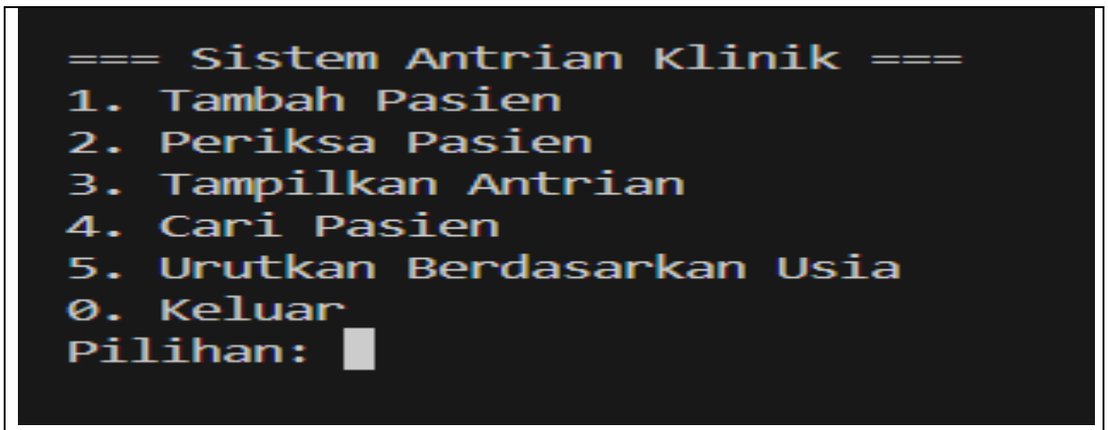
Fitur pencarian digunakan untuk menemukan informasi pasien tertentu berdasarkan nama. Hal ini sangat penting dalam aplikasi klinik untuk akses cepat terhadap data medis pasien.

```
Pasien *baru = new Pasien;
baru->next = nullptr;
baru->prev = tail;
```

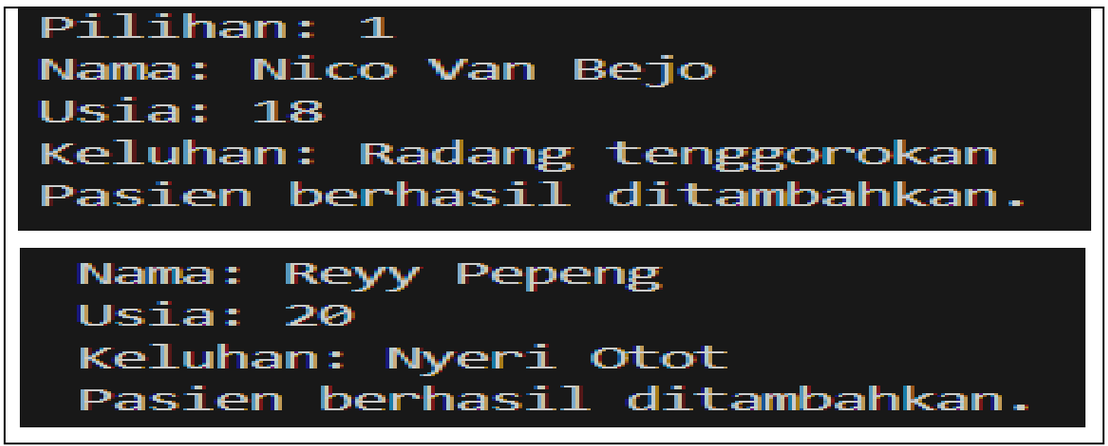
Tabel 3.3.6 Pointer

Pointer digunakan untuk membangun struktur linked list, mengatur alokasi memori dinamis, dan menghubungkan node satu dengan lainnya. Tanpa pointer, tidak mungkin membangun struktur data dinamis seperti linked list.

3.4 *Screenshot* Program



Gambar 3.3.1 Tampilan Awal



```
Nama: Krisna Menang
Usia: 21
Keluhan: Sakit gigi
Pasien berhasil ditambahkan.
```

```
Nama: Aslam Bintang
Usia: 35
Keluhan: Cidera ankle kaki
Pasien berhasil ditambahkan.
```

```
Nama: Han Marinir
Usia: 45
Keluhan: Luka tembak
Pasien berhasil ditambahkan.
```

Gambar 3.3.2 Menambah Pasien

```
Memeriksa pasien:
Nama: Nico Van Bejo, Usia: 18, Keluhan: Radang tenggorokan
Pasien selesai diperiksa dan disimpan ke riwayat.
```

```
Memeriksa pasien:
Nama: Reyy Pepeng, Usia: 20, Keluhan: Nyeri Otot
Pasien selesai diperiksa dan disimpan ke riwayat.
```

```
Memeriksa pasien:
Nama: Krisna Menang, Usia: 21, Keluhan: Sakit gigi
Pasien selesai diperiksa dan disimpan ke riwayat.
```

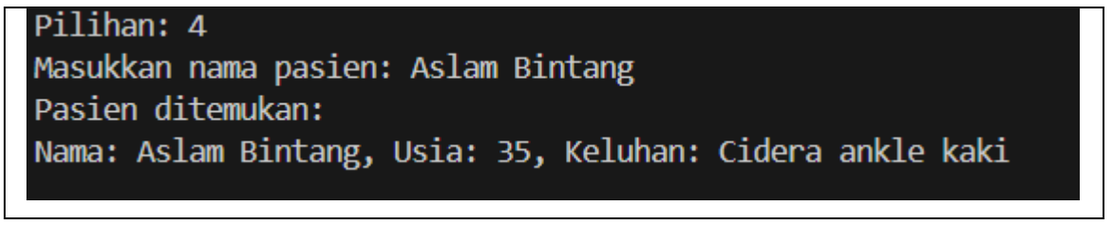
```
Memeriksa pasien:
Nama: Aslam Bintang, Usia: 35, Keluhan: Cidera ankle kaki
Pasien selesai diperiksa dan disimpan ke riwayat.
```

```
Memeriksa pasien:
Nama: Han Marinir, Usia: 45, Keluhan: Luka tembak
Pasien selesai diperiksa dan disimpan ke riwayat.
```

Gambar 3.3.3 Memeriksa Pasien

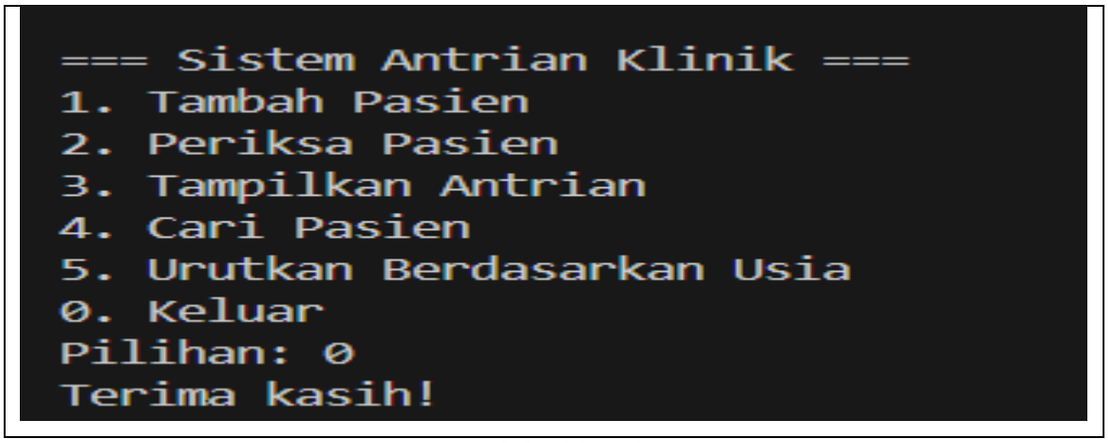
```
=== Daftar Pasien dalam Antrian ===
Nama: Nico Van Bejo, Usia: 18, Keluhan: Radang tenggorokan
Nama: Reyy Pepeng, Usia: 20, Keluhan: Nyeri otot
Nama: Krisna Menang, Usia: 21, Keluhan: Sakit gigi
Nama: Aslam Bintang, Usia: 35, Keluhan: Cidera ankle kaki
Nama: Han Marinir, Usia: 45, Keluhan: Luka tembak
```

Gambar 3.3.4 Menampilkan Antrean Pasien



Gambar 3.3.5 Mencari Pasien

Gambar 3.3.6 Urutan Pasien Berdasarkan Umur



Gambar 3.3.7 Menu Logout

3.5 Jadwal Pengerjaan

Berikut ini merupakan jadwal singkat dalam pengerjaan proyek akhir “Klinik”, yaitu:

No	Kegiatan	Pengerjaan				
		Mei				Juni
		16	18	22	30	5
1	Mengkaji Masalah					
2	Perancangan Sistem Program					
3	Pembuatan Program					
4	Pembuatan Laporan					
5	Presentasi					

Tabel 3.5.1 Jadwal Pengerjaan Proyek Akhir “Klinik”

3.6 Pembagian Tugas

Berikut ini merupakan pembagian tugas selama pengerjaan proyek akhir “Klinik”, yaitu:

No	Kegiatan	Penanggung Jawab	
		Satya Adi Wicaksana	Afwan Ayasyin
1	Mengkaji Masalah		
2	Perancangan Sistem Program		
3	Pembuatan Program		
4	Pembuatan Laporan		
5	Presentasi		

Tabel 3.6.1 Pembagian Tugas Proyek Akhir “Klinik”

LAMPIRAN