

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII  
“QUEUE”**



**Disusun Oleh :**

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas : IF 11 B

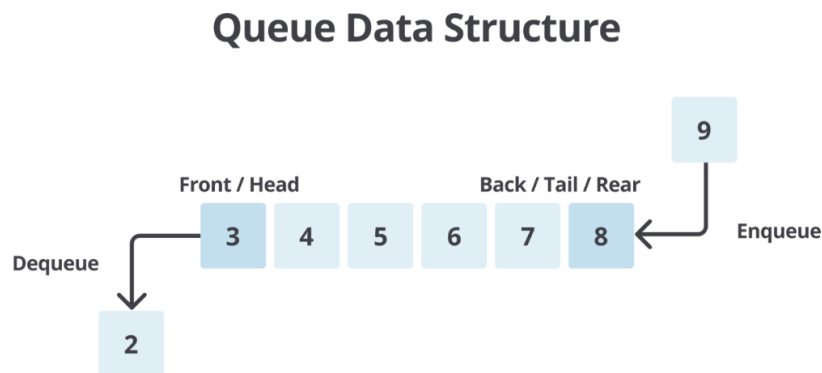
**DOSEN:**

WAHYU ANDI SAPUTRA, S.Pd. , M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## A. DASAR TEORI

Queue atau Antrian dalam bahasa Indonesia berarti struktur data yang menyusun elemen-elemen data secara linier. Prinsip dasar struktur data ini adalah “first in, first out” (FIFO). Artinya item data pertama yang dimasukkan ke dalam Queue adalah item data pertama yang dihapus. Prinsip FIFO dalam Queue Mekanismenya mirip dengan antrian orang di supermarket, dimana orang yang pertama datang juga merupakan orang pertama yang dilayani (first in, first out). Dalam struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut permulaan atau permulaan. Sebaliknya, urutan data terakhir (data yang baru saja ditambahkan) disebut Belakang, Belakang, atau Ekor. Proses penambahan data ke dalam Queue disebut enqueueing, dan proses mengeluarkan data dari Queue disebut dequeueing.



### Fungsi Queue

Queue memainkan peran penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrian tugas atau operasi secara efisien. Ini digunakan dalam sistem komputer untuk menangani tugas tugas seperti penjadwalan proses, antrian pesan, dan manajemen sumber daya.

### Jenis Queue

Jenis struktur data ini dapat diklasifikasikan berdasarkan implementasi atau penggunaannya. Jenis-jenis tersebut antara lain:

1. Berdasarkan implementasi
  - Linear/Simple Queue: Elemen data disusun dalam barisan linear, penambahan dan penghapusan elemen hanya terjadi pada kedua ujung barisan.
  - Antrian Melingkar : Mirip dengan tipe linier , namun kedua ujung antrian terhubung satu sama lain, menciptakan struktur antrian berputar.
2. Berdasarkan Penggunaan
  - Antrean Prioritas Berbasis Penggunaan: Setiap item memiliki prioritas tertentu. Item dengan prioritas tertinggi diterima terlebih dahulu.

- Antrean berujung ganda (dihapus dari antrean): Item dapat ditambahkan atau dihapus dari salah satu ujung antrean.

#### Kelebihan dan Keterbatasan

Meskipun struktur data antrian memiliki banyak kegunaan, Anda juga harus menyadari beberapa kelebihan dan keterbatasan yang harus Anda pertimbangkan sebelum menggunakannya.

##### 1. Manfaat

- Mengelola data dalam jumlah besar dengan mudah dan efisien.
- Proses memasukkan dan menghapus data mudah karena mengikuti aturan FIFO (first in, first out).
- Memproses tugas secara efisien berdasarkan siapa yang datang lebih dulu dilayani.

##### 2. Keterbatasan

- Mencari item tertentu dalam antrian tidak efisien.
- Alokasi memori yang cukup diperlukan untuk menyimpan antrian

## B. GUIDED

### 1. Guided I

#### Source Code

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

const int maksimalQueue = 5; //maksimal Queue
int front = 0;
int back= 0;
string queueTeller [5];

bool isFull (){
    if (back== maksimalQueue){
        return true;
    } else {
        return false;
    }
}

bool isEmpty (){
    if (back ==0){
        return true;
    } else {
        return false;
    }
}

void enqueueQueue (string data ){
    if (isFull()){
        cout <<"Queue penuh"<< endl;
    } else {
        if (isEmpty (){
            queueTeller [0] = data;
            front++;
            back++;
        } else {
            queueTeller [back] = data;
            back++;
        }
    }
}

void dequeueQueue (){
    if (isEmpty (){
        cout << "Queue kosong "<< endl;
    } else {
        for (int i=0; i< back; i++){
            queueTeller [i] = queueTeller [i+1];
        }
    }
}
```

```

        }
        back--;
    }
}

int countQueue (){
    return back;
}

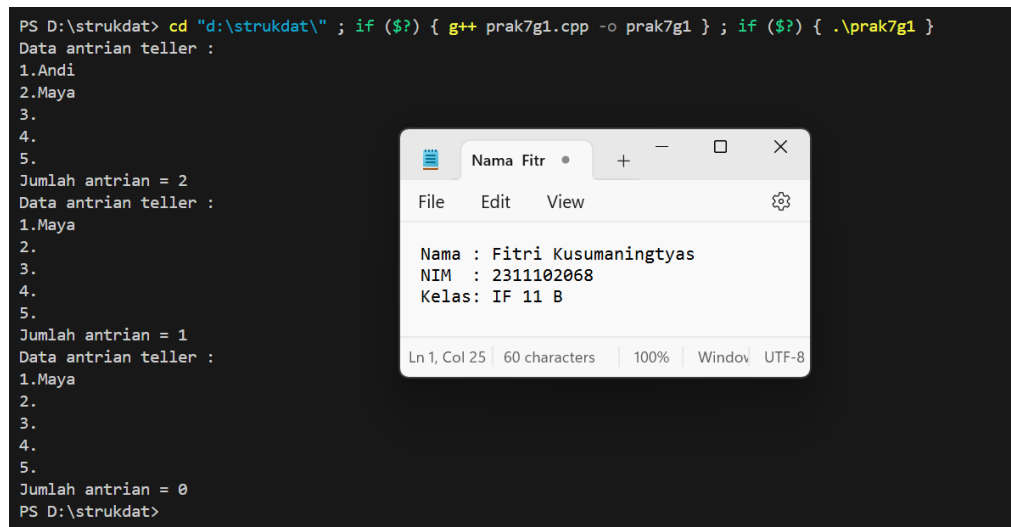
void clearQueue (){
    if (isEmpty()){
        cout << " Queue kosong" << endl;
    } else {
        for (int i=0;i<back; i++){
        }
        back = 0;
        front = 0;
    }
}

void viewQueue () {
    cout << "Data Queue teller :" << endl;
    for (int i= 0 ; i<maksimalQueue; i++){
        if (queueTeller [i] != " ") {
            cout << i+1 << "."<< queueTeller[i] <<endl;
        } else {
            cout << i+1 << ". (kosong)"<< endl;
        }
    }
}

int main (){
    enqueueQueue ("Andi");
    enqueueQueue ("Maya");
    viewQueue();
    cout << "Jumlah Queue = " << countQueue() << endl;
    dequeueQueue ();
    viewQueue();
    cout << "Jumlah Queue = " << countQueue() << endl;
    clearQueue ();
    viewQueue();
    cout << "Jumlah Queue = " << countQueue() << endl;
    return 0;
}

```

## Ouput Program :



The screenshot shows a terminal window on the left and a small application window titled 'Nama Fitr' on the right. The terminal displays the execution of a C++ program that simulates a queue. It starts with 5 people in the queue (Andi, Maya, 3 unnamed), then Maya is served, leaving 4 people. Then another Maya is served, leaving 3 people. The queue becomes empty, and the program ends. The application window displays the details of the person being served: Fitri Kusumaningtyas, NIM 2311102068, and Class IF 11 B.

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak7g1.cpp -o prak7g1 } ; if ($?) { .\prak7g1 }
Data antrian teller :
1.Andi
2.Maya
3.
4.
5.
Jumlah antrian = 2
Data antrian teller :
1.Maya
2.
3.
4.
5.
Jumlah antrian = 1
Data antrian teller :
1.Maya
2.
3.
4.
5.
Jumlah antrian = 0
PS D:\strukdat>
```

Nama Fitr

File Edit View

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window UTF-8

## Deskripsi Program :

Kode di atas merupakan implementasi dari struktur data Queue di C++.

- Const int maksimalQueue : Konstanta bilangan bulat dengan nilai 5 menunjukkan ukuran maksimum antrian.
- Int front dan int back : variabel integer diinisialisasi dengan nilai 0. Digunakan untuk melacak posisi isyarat maju dan mundur.
- queueTeller: Array string ukuran 5 digunakan untuk menyimpan item dalam antrian.
- Fungsi isFull(): Fungsi Boolean yang memeriksa apakah antrian sudah penuh dengan membandingkan nilai kembali ke MaximumQueue. Fungsi ini mengembalikan nilai true jika kembali adalah MaximumQueue. Artinya antrian sudah penuh.
- isEmpty(): Fungsi Boolean yang memeriksa apakah antrian kosong dengan membandingkan nilai back dengan 0. Jika kembali adalah 0, fungsi ini mengembalikan nilai true. Artinya antriannya kosong.
- enqueueQueue(data string): Fungsi yang menambahkan elemen ke antrian. Jika antrian penuh, fungsi ini mengembalikan pesan kesalahan. Jika tidak, fungsi akan menambahkan elemen ke antrian dan menambah nilai kembalian.
- dequeueQueue(): Berfungsi untuk menghapus elemen dari kepala antrian. Jika antrian kosong, fungsi ini mengembalikan pesan kesalahan. Jika tidak, fungsi akan memindahkan semua elemen dalam antrian ke kiri sebanyak satu dan mengurangi nilai yang dikembalikan.
- countQueue(): Fungsi yang mengembalikan jumlah elemen dalam antrian, diwakili oleh nilai kembalian .
- clearQueue(): Fungsi yang menghapus antrian dengan menyetel nilai sebelumnya dan berikutnya ke 0.

- `viewQueue()`: Berfungsi untuk menampilkan item dalam antrian. Fungsi ini mengulang array `queueTeller` dan mencetak setiap elemen beserta indeksinya.
- Fungsi Utama Fungsi `main()` mendemonstrasikan penggunaan fungsi antrian di atas.
  - Gunakan fungsi `enqueueQueue()` untuk menambahkan dua elemen 'Andi' dan 'Maya' ke antrian.
  - Lihat antrian menggunakan fungsi `viewQueue()`.
  - Gunakan fungsi `countQueue()` untuk mencetak jumlah elemen dalam antrian.
  - Gunakan fungsi `dequeueQueue()` untuk menghapus elemen dari depan antrian.
  - Gunakan fungsi `viewQueue()` untuk melihat antrean yang diperbarui.
  - Mengembalikan jumlah item dalam antrian yang diperbarui menggunakan fungsi `countQueue()`.
  - Hapus antrian menggunakan fungsi `clearQueue()`.
  - Lihat antrian kosong menggunakan fungsi `viewQueue()`.
  - Gunakan fungsi `countQueue()` untuk mengembalikan jumlah elemen dalam antrian yang kosong.

## C. UNGUIDED

### 1. Unguided 1 Source Code

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

const int maximalQueue = 5;
int length = 0;

struct Node {
    string data;
    Node *next;
};

Node *head;
Node *tail;

void init(){
    head = NULL;
    tail = NULL;
}

bool isFull(){
    return (length == maximalQueue);
}

bool isEmpty(){
    return head == NULL;
}

void enqueueAntrian(string nilai){
    if (isFull()) {
        cout << "Antrian Penuh" << endl;
    } else {
        Node *baru = new Node;
        baru->data = nilai;
        baru->next = NULL;
        if (isEmpty()) {
            head = tail = baru;
        } else {
            tail->next = baru;
            tail = baru;
        }
        length++;
    }
}
```



```

void dequeueAntrian() {
    if (!isEmpty()) {
        Node *hapus = head;
        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue() {
    Node *bantu = head;
    while (bantu != NULL) {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "Semua antrian dihapus" << endl;
}

void viewQueue() {
    if (!isEmpty()) {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL) {
            cout << index << ". " << bantu->data << " " <<
endl;

            bantu = bantu->next;
            index++;
        }
        cout << endl;
    } else {
        cout << "Tidak ada antrian" << endl;
    }
}

int countQueue() {
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
}

```

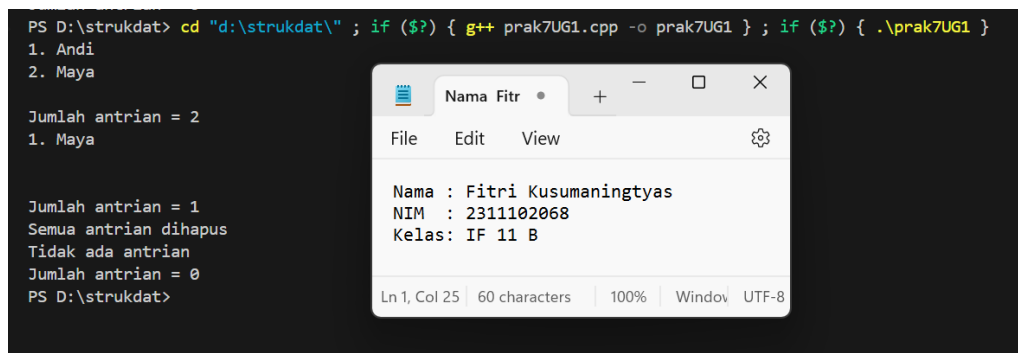
```

    }
    return jumlah;
}

int main() {
    init();
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "\nJumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "\nJumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Output Program :



The screenshot shows a terminal window with the following output:

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak7UG1.cpp -o prak7UG1 } ; if ($?) { .\prak7UG1 }
1. Andi
2. Maya

Jumlah antrian = 2
1. Maya

Jumlah antrian = 1
Semua antrian dihapus
Tidak ada antrian
Jumlah antrian = 0
PS D:\strukdat>

```

Overlaid on the terminal is a Notepad window titled "Nama Fitr" containing the following text:

```

Nama : Fitri Kusumaningtyas
NIM  : 2311102068
Kelas: IF 11 B

```

The Notepad window also shows a status bar at the bottom: "Ln 1, Col 25 | 60 characters | 100% | Window | UTF-8".

Deskripsi Program :

- Kode di atas merupakan implementasi antrian menggunakan struktur data daftar tertaut. Di bawah ini adalah deskripsi kodenya:
- Const int maximumQueue untuk menunjukkan ukuran maksimum antrian.
- struktur node yang berisi data dan pointer next.
- Fungsi init() digunakan untuk menginisialisasi awal dan akhir ke NULL.
- Fungsi isFull() dan isEmpty() memeriksa apakah antrian penuh atau kosong.
- Fungsi enqueueQueue(value string) digunakan untuk menambahkan data ke antrian. Ketika antrian sudah penuh, program menampilkan pesan "Antrian sudah penuh".

- Fungsi dequeueQueue() digunakan untuk menghapus data dari antrian. Jika antrian kosong, program akan menampilkan pesan “Tidak Ada Antrian”.
- Fungsi clearQueue() digunakan untuk menghapus semua data dari antrian.
- Fungsi viewQueue() digunakan untuk menampilkan data dalam antrian.
- Fungsi countQueue() digunakan untuk menghitung jumlah data dalam antrian.
- Pada fungsi main(), buat antrian dengan menginisialisasi awal dan akhir ke NULL menggunakan fungsi init().
  - Gunakan fungsi enqueueQueue() untuk menambahkan data 'Andi' dan 'Maya' ke antrian.
  - Gunakan fungsi viewQueue() untuk menampilkan data dalam antrian.
  - Gunakan fungsi countQueue() untuk menghitung jumlah data dalam antrian dan menampilkan hasilnya.
  - Gunakan fungsi dequeueQueue() untuk menghapus data dari antrian.
  - Gunakan fungsi viewQueue() untuk menampilkan data dalam antrian,
  - Gunakan fungsi countQueue() untuk menghitung jumlah data dalam antrian, dan tampilkan hasilnya.
  - Langkah Terakhir, hapus semua data dari antrian menggunakan fungsi clearQueue() dan tampilkan data dalam antrian menggunakan fungsi viewQueue().
  - Hitung jumlah data dalam antrian menggunakan fungsi countQueue() dan program akan menampilkan hasilnya.

## 2. Unguided 2

### Source Code

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

const int maximalQueue = 5;
int length = 0;

struct Node
{
    string nama;
    string nim;
    Node *next;
};

Node *head;
```

```

Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nama, string nim)
{
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
        baru->nama = nama;
        baru->nim = nim;
        baru->next = NULL;
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
        cout << endl << "Berhasil Masuk Antrian";
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {

```

```

        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->nama << " ( " <<
bantu->nim << " ) " << endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Antrian masih kosong" << endl;
    }
}

```

```

    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

int main()
{
    init();
    system("cls");
    do
    {
        int pilih;
        string nama, nim;
        cout << "\n=====ANTRIAN MAHASISWA===== "
<< endl;
        cout << "1. Tambah Antrian " << endl;
        cout << "2. Keluar Antrian " << endl;
        cout << "3. Jumlah Antrian " << endl;
        cout << "4. Lihat Antrian " << endl;
        cout << "5. Hapus Antrian " << endl;
        cout << "0. Keluar Program" << endl;
        cout << "Pilih menu: ";
        cin >> pilih;
        cout << endl;
        switch (pilih)
        {
            case 1:
                cout << "Masukkan Nama Mahasiswa : ";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan NIM : ";
                cin >> nim;
                enqueueAntrian(nama, nim);
                system("pause > nul");
                break;

            case 2:
                dequeueAntrian();

```

```

        cout << "Berhasil keluar" << endl;
        cout << endl;

        system("pause > nul");
        break;

    case 3:
        cout << "Jumlah Antrian : " << countQueue() <<
endl;
        cout << endl;

        system("pause > nul");
        break;

    case 4:
        viewQueue();
        cout << endl;

        system("pause > nul");
        break;

    case 5:
        clearQueue();
        cout << "Data berhasil dihapus" << endl;
        cout << endl;

        system("pause > nul");
        break;

    case 0:
        cout << "Anda Keluar dari Program" << endl;
        exit(0);

    default:
        break;
}

} while (true);

return 0;
}

```

## Output Program :

```
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 1

Masukkan Nama Mahasiswa : Anisa Yasaroh
Masukkan NIM : 2311102063

Berhasil Masuk Antrian
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 1

Masukkan Nama Mahasiswa : Fitri Kusumaningtyas
Masukkan NIM : 2311102068

Berhasil Masuk Antrian
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 1

Masukkan Nama Mahasiswa : Trie Nabilla Farhah
Masukkan NIM : 2311102071

Berhasil Masuk Antrian
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 1

Masukkan Nama Mahasiswa : Anisah Syifa M.R
Masukkan NIM : 2311102080

Berhasil Masuk Antrian
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 3

Jumlah Antrian : 4

=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 4

1. Anisa Yasaroh ( 2311102063 )
2. Fitri Kusumaningtyas ( 2311102068 )
3. Trie Nabilla Farhah ( 2311102071 )
4. Anisah Syifa M.R ( 2311102080 )

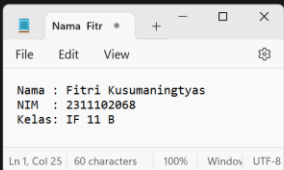
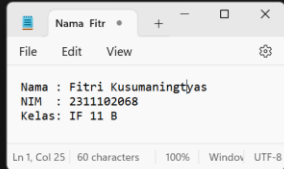
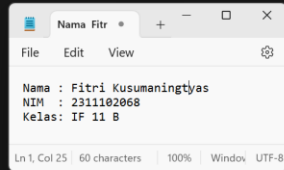
=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 2

Berhasil keluar

=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar Program
Pilih menu: 3

Jumlah Antrian : 3

=====ANTRIAN MAHASISWA=====
1. Tambah Antrian
2. Keluar Antrian
```





```
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
6. Keluar Program
Pilih menu: 4

1. Fitri Kusumaningtyas ( 2311102068 )
2. Trie Nabilla Farhah ( 2311102071 )
3. Anisah Syifa M.R ( 2311102080 )

=====ANTRIAN MAHASISMA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
6. Keluar Program
Pilih menu: 5

Data berhasil dihapus

=====ANTRIAN MAHASISMA=====
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
6. Keluar Program
Pilih menu: 6

Anda Keluar dari Program
PS D:\struktat>
```

### Deskripsi Program :

Kode di atas adalah program yang menggunakan struktur data linked list untuk mengelola antrian siswa. Berikut penjelasan kodenya.

- Const int maximumQueue untuk menunjukkan ukuran maksimum antrian.
- struktur node yang berisi nama, nim, dan data pointer next.
- Fungsi init() digunakan untuk menginisialisasi awal dan akhir ke NULL.
- Fungsi isFull() dan isEmpty() memeriksa apakah antrian penuh atau kosong.
- Fungsi enqueueQueue(string name, string nim) digunakan untuk menambahkan data ke antrian. Ketika antrian sudah penuh, program menampilkan pesan "Antrian sudah penuh".
- Fungsi dequeueQueue() digunakan untuk menghapus data dari antrian. Jika antrian kosong, program akan menampilkan pesan "Tidak Ada Antrian".
- Fungsi clearQueue() digunakan untuk menghapus semua data dari antrian.
- Fungsi viewQueue() digunakan untuk menampilkan data dalam antrian.
- Fungsi countQueue() digunakan untuk menghitung jumlah data dalam antrian.
- Dalam fungsi main(), terdapat perulangan do- while yang dijalankan dengan nilai true.
- Pada setiap iterasi, program menampilkan menu untuk memilih tindakan yang akan dilakukan. Yaitu menambah antrian, meninggalkan antrian, menampilkan jumlah antrian, menampilkan antrian, menghapus antrian, atau keluar dari program.

- Jika pilihan yang dipilih adalah 1, program akan meminta Anda memasukkan nama siswa dan NIM dan menggunakan fungsi `enqueueQueue()` untuk menambahkan data ke antrian.
- Jika pilihan yang dipilih adalah 2, program menggunakan fungsi `dequeueQueue()` untuk menghapus data dari antrian.
- Jika pilihan yang dipilih adalah 3, program menggunakan fungsi `countQueue()` untuk menampilkan jumlah antrian.
- Jika pilihan yang dipilih adalah 4, program menggunakan fungsi `viewQueue()` untuk menampilkan data dalam antrian.
- Jika pilihan yang dipilih adalah 5, program akan menghapus semua data dari antrian menggunakan fungsi `clearQueue()`.
- Jika opsi yang dipilih adalah 0, program akan keluar dari loop dan keluar.
- Program kemudian menggunakan perintah `system("cls")` Kemudian kembali ke awal perulangan.

#### D. KESIMPULAN

Queue atau antrian merupakan struktur data yang mendasar dalam pemrograman, termasuk C++. Mempelajari hal ini mempunyai banyak manfaat, antara lain:

- Memahami alur kerja antrian : Antrian menyimulasikan antrian dunia nyata di mana item ditambahkan dan dihapus berdasarkan urutan kedatangannya. Memeriksa antrian dapat membantu Anda memahami cara kerja sistem antrian, seperti antrian checkout, antrian cetak, dan antrian pesan.
- Meningkatkan keterampilan pemecahan masalah : antrian dapat digunakan untuk menyelesaikan berbagai masalah pemrograman, termasuk: Contoh: manajemen memori, penjadwalan tugas, pemrosesan data. Mempelajari cara menangani isyarat meningkatkan kemampuan Anda untuk memecahkan masalah secara efisien dan terstruktur.
- Perluas pengetahuan Anda tentang struktur data : Antrian adalah salah satu struktur data linier yang paling penting. Penelitian memperluas pengetahuan Anda tentang struktur data secara keseluruhan dan membantu Anda mempelajari struktur data lain seperti tumpukan, daftar tertaut, dan pohon.
- Tingkatkan keterampilan pemrograman C++ : Mempelajari cara mengimplementasikan dan menggunakan antrian di C++ dapat meningkatkan pengetahuan Anda tentang bahasa pemrograman ini

## E. REFERESNSI JURNAL

- Goponenko, A., & Carroll, S. (2019). A C++ implementation of a lock-free priority queue based on Multi-Dimensional Linked List. *Link: [https://www.researchgate.net/publication/337020321\\_A\\_C\\_Implementation\\_of\\_a\\_Lock-Free\\_Priority\\_Queue\\_Based\\_on\\_Multi-Dimensional\\_Linked\\_List](https://www.researchgate.net/publication/337020321_A_C_Implementation_of_a_Lock-Free_Priority_Queue_Based_on_Multi-Dimensional_Linked_List)*.
- Jian, Y. C., Chung, M. S., Susanto, H., & Leu, F. Y. (2022, June). 5G Base Station Scheduling. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (pp. 315-324). Cham: Springer International Publishing.
- Sari, I. P., Batubara, I. H., Ramadhani, F., & Wardani, S. (2022). Perancangan Sistem Antrian pada Wahana Hiburan dengan Metode First In First Out (FIFO). *Sudo Jurnal Teknik Informatika*, 1(3), 116-123.
- Josipović, L., Guerrieri, A., & Ienne, P. (2021). From C/C++ code to high-performance dataflow circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(7), 2142-2155.
- Beard, J. C., Li, P., & Chamberlain, R. D. (2015, February). RaftLib: A C++ template library for high performance stream parallel processing. In *Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores* (pp. 96-105).