

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VII  
QUEUE**



**Disusun Oleh :**

**NAMA : FAISAL KHOIRUDDIN**

**NIM : 2311102046**

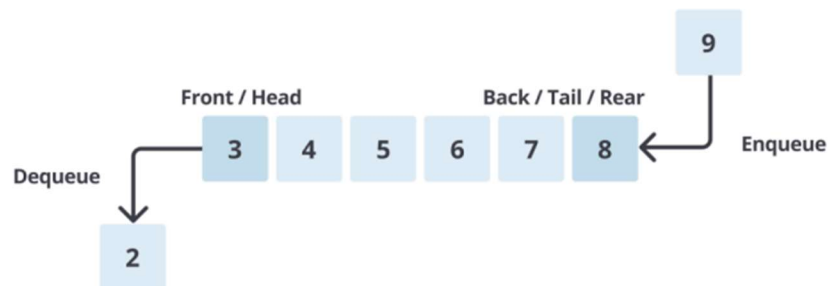
**Dosen**

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

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

## A. Dasar Teori

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Class pada Queue mendukung Queue first-in, first-out (FIFO) pada struktur data. Queue mirip dengan konsep antrian pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu. Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. Front/head adalah pointer ke elemen pertama dalam queue dan rear/tail/back adalah pointer ke elemen terakhir dalam queue.



### FIRST IN FIRST OUT (FIFO)

Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir diinputkan akan berada paling dengan dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.

Pada Queue, operasi tersebut dilakukan ditempat berbeda (melalui salah satu ujung) karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert. maupun delete. Prosedur ini sering

disebut Enqueue dan Dequeue pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya. Operasi pada Queue

- enqueue() : menambahkan data ke dalam queue.
- dequeue() : mengeluarkan data dari queue.
- peek() : mengambil data dari queue tanpa menghapusnya.
- isEmpty() : mengecek apakah queue kosong atau tidak.
- isFull() : mengecek apakah queue penuh atau tidak.
- size() : menghitung jumlah elemen dalam queue.

## Guided

### 1. Guided 1

#### Source Code

```
#include <iostream>
using namespace std;
const int maksimalQueue = 5; // Maksimal
antrian
int front = 0;                // Penanda
antrian
int back = 0;                 // Penanda
string queueTeller[5];       // Fungsi
pengecekan
bool isFull()
{ // Pengecekan antrian penuh atau tidak
    if (back == maksimalQueue)
    {
        return true; // =1
    }
    else
    {
        return false;
    }
}
bool isEmpty()
{ // Antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
void enqueueAntrian(string data)
{ // Fungsi menambahkan antrian
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        if (isEmpty())
        { // Kondisi ketika queue kosong
```

```

        queueTeller[0] = data;
        front++;
        back++;
    }
    else
    { // Antrianya ada isi
        queueTeller[back] = data;
        back++;
    }
}

void dequeueAntrian()
{ // Fungsi mengurangi antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i
+ 1];
        }
        back--;
    }
}

int countQueue()
{ // Fungsi menghitung banyak antrian
    return back;
}

void clearQueue()
{ // Fungsi menghapus semua antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

```

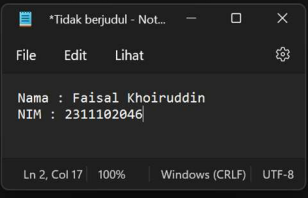
```

}
void viewQueue()
{ // Fungsi melihat antrian
    cout << "Data antrian 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()
{
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " <<
countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " <<
countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " <<
countQueue() << endl;
    return 0;
}

```

Screenshots Output

```
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-z0z0a4bc.jgy' '--stdout=Microsoft-MIEngine-Out-erfimpe.5rg' '--stderr=Microsoft-MIEngine-Error-osvnc5u.0sg' '--pid=Microsoft-MIEngine-Pid-1ztyqlt.2yd' '--dbgExe=c:\mingw32\bin\gdb.exe' '--interpreter=mi'
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2>
```



### Deskripsi:

Program tersebut merupakan Queue. Program tersebut melakukan operasi seperti enqueueAntrian (menambahkan data), dequeueAntrian (mengurangi data), countQueue (menghitung data), clearQueue(menghapus semua data antrian), dan viewQueue (melihat data antrian). Program tersebut menampilkan output secaraurut sesuai posisi.

- `#include <iostream>` ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- `using namespace std;` ➔ digunakan untuk mendeklarasikan/memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- `const int maksimalQueue = 5;` ➔ maksimal antrian yaitu 5
- `int front = 0;` ➔ untuk menandai posisi awal antrian queue
- `int back = 0;` ➔ untuk menandai posisi akhir antrian queue
- `string queueTeller[5];` ➔ array untuk menyimpan data dalam antrian yaitu 5
- `bool isFull()` ➔ fungsi untuk mengecek apakah antrian sudah penuh atau belum
- `if (back == maksimalQueue)` ➔ percabangan if jika jumlah antrian sudah mencapai maksimal

- `return true;` ➔ mengembalikan true
- `else` ➔ jika jumlah antrian belum mencapai maksimal
- `return false;` ➔ mengembalikan false
- `bool isEmpty()` ➔ fungsi untuk mengecek apakah antrian kosong atau tidak
- `if (back == 0)` ➔ percabangan if jika tidak ada antrian
- `return true;` ➔ maka antrian kosong
- `else` ➔ jika ada antrian
- `return false;` ➔ maka antrian dianggap tidak kosong
- `void enqueueAntrian(string data)` ➔ fungsi untuk menambahkan data ke antrian
- `if (isFull())` ➔ percabangan if jika antrian penuh
- `cout << "Antrian penuh" << endl;` ➔ cout menampilkan statement antrian penuh
- `else` ➔ jika antrian belum penuh
- `if (isEmpty())` ➔ percabangan if jika antrian kosong
- `queueTeller[0] = data;` ➔ maka data akan ditambahkan ke antrian pertama
- `front++;` ➔ `front++` (increment)
- `back++;` ➔ `back++` (increment)
- `else` ➔ jika antrian tidak kosong
- `queueTeller[back] = data;` ➔ maka data akan ditambahkan ke antrian terakhir
- `back++;` ➔ `back++` (increment)
- `void dequeueAntrian()` ➔ fungsi untuk menghapus data dari antrian
- `if (isEmpty())` ➔ percabangan if jika antrian kosong
- `cout << "Antrian kosong" << endl;` ➔ cout menampilkan statement antrian kosong



- else → jika antrian tidak kosong
- for (int i = 0; i < back; i++) → perulangan for untuk menggeser antrian ke depan
- queueTeller[i] = queueTeller[i + 1]; → menggeser semua elemen antrian ke depan setelah elemen di hapus
- back--; → jumlah antrian dikurangi satu
- int countQueue() → fungsi untuk menghitung jumlah antrian
- return back; → mengembalikan jumlah antrian
- void clearQueue() → fungsi untuk menghapus semua antrian
- if (isEmpty()) → percabangan if jika antrian kosong
- cout << "Antrian kosong" << endl; → cout menampilkan statement antrian kosong
- else → jika antrian tidak kosong
- for (int i = 0; i < back; i++) → perulangan for untuk menghapus semua data antrian
- queueTeller[i] = ""; → menghapus data antrian pada indeks ke i
- back = 0; → jumlah antrian diatur menjadi 0
- front = 0; → front diatur menjadi 0
- void viewQueue() → fungsi untuk melihat antrian
- cout << "Data antrian teller:" << endl; → cout menampilkan statement data antrian teller
- for (int i = 0; i < maksimalQueue; i++) → perulangan for sebanyak maksimal antrian
- if (queueTeller[i] != "") → percabangan if jika data antrian tidak kosong
- cout << i + 1 << ". " << queueTeller[i] << endl; → cout menampilkan data antrian
- else → jika data antrian kosong
- cout << i + 1 << ". (kosong)" << endl; → cout menampilkan

## B. Unguided/Tugas

### 1. Unguided 1

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

### Source Code

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

using namespace std;

class Node
{
public:
    string nama;
    string nim;
    Node* next;

    Node (string nama, string nim)
    {
        this->nama = nama;
        this->nim = nim;
        this->next = nullptr;
    }
};

class Queue {
private:
    Node* front;
    Node* rear;
    int capacity;

public:
    Queue (int capacity){
        this->capacity = capacity;
        front = nullptr;
        rear = nullptr;
    }

    void enqueue (string nama, string nim) {
```

```

        if (isFull()){
            cout << "Antrian penuh. Tidak
dapat menambahkan lebih banyak elemen." <<
endl;
            return;
        }
        Node* newNode = new Node (nama, nim);
        if (isEmpty()){
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
    }

void dequeue(){
    if (isEmpty()){
        cout << "Antrian kosong. Tidak
dapat menghapus elemen." << endl;
        return;
    }
    Node* temp = front;
    front = front->next;
    string namaYangDihapus = temp->nama;
    delete temp;
    cout << "Data di depan yaitu (" <<
namaYangDihapus << ") telah dihapus." <<
endl;
}

string peek(){
    if (isEmpty()){
        cout << "Antrian kosong. Tidak
dapat melihat elemen." << endl;
        return"";
    }
    return front->nama + " (" + front-
>nim + ") ";
}

bool isEmpty(){
    return front == nullptr;
}

bool isFull(){
    int count = 0;
    Node* current = front;

```

```

        while (current != nullptr) {
            count++;
            current = current->next;
        }
        return count == capacity;
    }

    int size(){
        int count = 0;
        Node* current = front;
        while (current != nullptr) {
            count++;
            current = current->next;
        }
        return count;
    }

    void viewQueue (){
        cout << "Data antrian" << endl;
        cout << setfill('=') << setw(57)
<< "=" << setfill(' ') << endl;
        cout << "| " << setw(2) << left <<
"NO" << " | " << setw(15) << left <<
>Nama" << " | " << setw(30) << left <<
"NIM" << " |" << endl;
        cout << setfill('=') << setw(57)
<< "=" << setfill(' ') << endl;
        Node* current = front;
        int no = 1;
        while (current != nullptr) {
            cout << "| " << setw(2) <<
left << no << " | " << setw(15) << left <<
current->nama << " | " << setw(30) << left
<< current->nim << " |" << endl;
            current = current->next;
            no++;
        }
        cout << setfill('=') << setw(57)
<< "=" << setfill(' ') << endl;
    }
};

int main() {
    Queue queue (5);
    int pilihan;
    string nama, nim;

```

```

do {
    cout << "\nMenu Operasi
Antrian:\n";
    cout << setfill ('=') << setw (39)
<< "=" << setfill (' ') << endl;
    cout << "| " << setw (2) << left
<< "1" << " | " << "Tambahkan data" <<
"
    | \n";
    cout << "| " << setw (2) << left
<< "2" << " | " << "Hapus data" <<
"
    | \n";
    cout << "| " << setw (2) << left
<< "3" << " | " << "Melihat data antrian
di depan" << " | \n";
    cout << "| " << setw (2) << left
<< "4" << " | " << "Jumlah elemen antrian"
<< "
    | \n";
    cout << "| " << setw (2) << left
<< "5" << " | " << "Melihat seluruh
antrian" << "
    | \n";
    cout << "| " << setw (2) << left
<< "6" << " | " << "Keluar" <<
"
    | \n";
    cout << setfill ('=') << setw (39)
<< "=" << setfill (' ') << endl;
    cout << "Pilih menu dari 1 sampai
6: ";

    cin >> pilihan;
    cout << "\n";

    switch (pilihan){
        case 1:
            cout << "Masukkan nama :
";

            cin >> nama;
            cout << "Masukkan NIM :
";

            cin >> nim;
            queue.enqueue(nama, nim);
            break;
        case 2:
            queue.dequeue();
            cout << "Data di depan
telah dihapus." << endl;
            break;
    }
}

```

```

        case 3:
            cout << "Data di depan: "
<< queue.peek() << endl;
            break;
        case 4:
            cout << "Jumlah data dalam
antrian : " << queue.size() << endl;
            break;
        case 5:
            queue.viewQueue();
            break;
        case 6:
            cout << "Keluar dari
program." << endl;
            break;
        default:
            cout << "Pilihan tidak
valid. Silahkan coba lagi." << endl;
            }
        } while (pilihan != 6);

        return 0;
    }

```

## Screenshots Output

```

PS C:\Users\ASUS\OneDrive\Dokumen\semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\deb
gAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-5wziwcob.2iz' '--stdout=Microsoft-MIEngine-Out-mgz4t
b5a.1w' '--stderr=Microsoft-MIEngine-Error-1e0xbbm.1bo' '--pid=Microsoft-MIEngine-Pid-32bayybx.15d' '--dbgExe=c:\mingw32\
bin\gdb.exe' '--interpreter=mi'

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 1

Masukkan nama : Andi
Masukkan NIM : 2311102098

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 1

```

```
Masukkan nama : Adi
Masukkan NIM : 2311102097

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 1

Masukkan nama : Doni
Masukkan NIM : 2311102096

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 1
```

\*Tidak berjudul - Not...

File Edit Lihat

Nama : Faisal Khoiruddin  
NIM : 2311102046

Ln 2, Col 17 100% Windows (CRLF) UTF-8

```
Masukkan nama : Toni
Masukkan NIM : 2311102095

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 1

Masukkan nama : Roni
Masukkan NIM : 2311102094

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 5
```

\*Tidak berjudul - Not...

File Edit Lihat

Nama : Faisal Khoiruddin  
NIM : 2311102046

Ln 2, Col 17 100% Windows (CRLF) UTF-8

```
Data antrian
=====
| NO | Nama | NIM |
=====
| 1 | Andi | 2311102098 |
| 2 | Adi | 2311102097 |
| 3 | Doni | 2311102096 |
| 4 | Toni | 2311102095 |
| 5 | Roni | 2311102094 |
=====

Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar |
=====
Pilih menu dari 1 sampai 6: 2

Data di depan yaitu (Andi) telah dihapus.
Data di depan telah dihapus.
```

\*Tidak berjudul - Not...

File Edit Lihat

Nama : Faisal Khoiruddin  
NIM : 2311102046

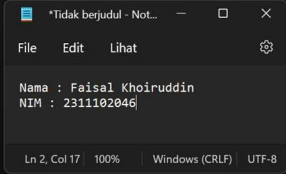
Ln 2, Col 17 100% Windows (CRLF) UTF-8

```
Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data    |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar       |
=====
Pilih menu dari 1 sampai 6: 3

Data di depan: Adi (2311102097)

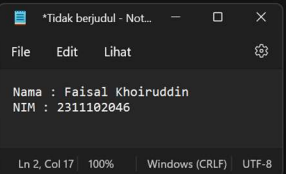
Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data    |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar       |
=====
Pilih menu dari 1 sampai 6: 4

Jumlah data dalam antrian : 4
```



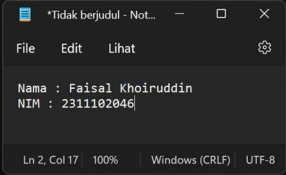
```
Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data    |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar       |
=====
Pilih menu dari 1 sampai 6: 5

Data antrian
=====
| NO | Nama      | NIM      |
=====
| 1  | Adi       | 2311102097 |
| 2  | Doni      | 2311102096 |
| 3  | Toni      | 2311102095 |
| 4  | Roni      | 2311102094 |
=====
```



```
Menu Operasi Antrian:
=====
| 1 | Tambahkan data |
| 2 | Hapus data    |
| 3 | Lihat data antrian di depan |
| 4 | Jumlah elemen antrian |
| 5 | Lihat seluruh antrian |
| 6 | Keluar       |
=====
Pilih menu dari 1 sampai 6: 6

Keluar dari program.
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2>
```



## Deskripsi:

Program tersebut program implementasi dari queue. Program tersebut yaitu menginput nama dan nim kemudian mengurutkan sesuai urutan antrian. Pada program tersebut terdapat fungsi enqueue (menambahkan), dequeue(menghapus), peek(melihat data di antrian), size(mendapatkan jumlah elemen), viewqueue(menampilkan seluruh antrian). Pada fungsi enqueue program tersebut menambahkan sesuai antrian. Pada fungsi dequeue program tersebut menghapus data mulai dari yang paling atas. Pada fungsi peek melihat data pada antrian. Pada fungsi size untuk menghitung jumlah elemen yang ada. Pada fungsi



viewqueue menampilkan keseluruhan data.

- `#include <iostream>` → merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- `#include <iomanip>` → standar untuk menentukan beberapa manipulator yang masing-masing mengambil satu argumen
- `using namespace std;` → digunakan untuk mendeklarasikan/memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- `class Node` → mendefinisikan kelas node
- `public:` → mengatur ke publik
- `string nama;` → deklarasi variabel untuk menyimpan nama
- `string nim;` → deklarasi variabel untuk menyimpan nim
- `Node* next;` → pointer ke node berikutnya dalam antrian
- `Node (string nama, string nim)` → konstruktor node terdiri dari string nama dan string nim
- `this->nama = nama;` → menginisialisasi nama
- `this->nim = nim;` → menginisialisasi nim
- `this->next = nullptr;` → menginisialisasi next sebagai nullptr
- `class Queue {` → mendefinisikan kelas queue
- `private:` → mengatur ke private
- `Node* front;` → pointer ke node depan antrian
- `Node* rear;` → pointer ke node belakang antrian
- `int capacity;` → kapasitas maksimum antrian
- `public:` → mengatur ke publik
- `Queue (int capacity){` → konstruktor queue
- `this->capacity = capacity;` → menginisialisasi kapasitas
- `front = nullptr;` → menginisialisasi front sebagai nullptr

- `rear = nullptr;` ➔ menginisialisasi rear sebagai nullptr
- `void enqueue (string nama, string nim) {` ➔ fungsi untuk menambahkan elemen ke antrian
- `if (isFull()) {` ➔ percabangan if jika antrian penuh
- `cout << "Antrian penuh. Tidak dapat menambahkan lebih banyak elemen." << endl;` ➔ cout menampilkan antrian penuh
- `return;` ➔ mengembalikan
- `Node* newNode = new Node (nama, nim);` ➔ memuat node baru
- `if (isEmpty()) {` ➔ percabangan if jika antrian kosong
- `front = rear = newNode;` ➔ node baru menjadi satu-satunya node dalam antrian
- `} else {` ➔ jika antrian tidak kosong
- `rear->next = newNode;` ➔ menambahkan node baru ke belakang antrian
- `rear = newNode;` ➔ memperbarui rear
- `void dequeue() {` ➔ fungsi untuk menghapus elemen dari antrian
- `if (isEmpty()) {` ➔ percabangan if jika antrian kosong
- `cout << "Antrian kosong. Tidak dapat menghapus elemen." << endl;` ➔ cout menampilkan statement antrian kosong
- `return;` ➔ mengembalikan
- `Node* temp = front;` ➔ menyimpan node depan sementara
- `front = front->next;` ➔ memperbarui front
- `string namaYangDihapus = temp->nama;` ➔ menyimpan nama yang dihapus
- `delete temp;` ➔ menghapus node depan
- `cout << "Data di depan yaitu (" << namaYangDihapus << ") telah dihapus." << endl;` ➔ cout menampilkan statement data di depan yaitu

- `string peek(){` ➔ fungsi untuk melihat elemen depan antrian
- `if (isEmpty()){` ➔ percabangan if jika antrian kosong
- `cout << "Antrian kosong. Tidak dapat melihat elemen." << endl;` ➔ `cout` menampilkan statement antrian kosong. Tidak dapat melihat antrian
- `return"";` ➔ mengembalikan
- `return front->nama + " (" + front->nim + ") ";` ➔ mengembalikan nama dan nim dari elemen depan
- `bool isEmpty(){` ➔ fungsi untuk memeriksa apakah antrian kosong
- `return front == nullptr;` ➔ mengembalikan true jika front adalah nullptr
- `bool isFull(){` ➔ fungsi untuk memeriksa apakah antrian penuh
- `int count = 0;` ➔ menginisialisasi penghitung
- `Node* current = front;` ➔ mulai dari node depan
- `while (current != nullptr) {` ➔ selama current belum mencapai akhir antrian
- `count++;` ➔ `count++` (increment) menaik
- `current = current->next;` ➔ pindah ke node berikutnya
- `return count == capacity;` ➔ mengembalikan true jika jumlah node sama dengan kapasitas
- `int size(){` ➔ fungsi untuk mendapatkan jumlah elemen dalam antrian
- `int count = 0;` ➔ menginisialisasi penghitung
- `Node* current = front;` ➔ mulai dari node depan
- `while (current != nullptr) {` ➔ selama belum mencapai akhir antrian
- `count++;` ➔ `count++` (increment) menaik
- `current = current->next;` ➔ pindah ke node berikutnya
- `return count;` ➔ mengembalikan jumlah elemen dalam antrian

- `void viewQueue ()` { ➔ fungsi untuk menampilkan seluruh antrian
- `cout << "Data antrian" << endl;` ➔ `cout` menampilkan statement data antrian
- `cout << setfill('=') << setw(57) << "=" << setfill(' ') << endl;` ➔ menampilkan garis batas atas
- `cout << "| " << setw(2) << left << "NO" << " | " << setw(15) << left << "Nama" << " | " << setw(30) << left << "NIM" << " | " << endl;` ➔ menampilkan no, nama, nim disertai garis pembatas
- `cout << setfill('=') << setw(57) << "=" << setfill(' ') << endl;` ➔ menampilkan garis batas bawah
- `Node* current = front;` ➔ mulai dari node depan
- `int no = 1;` ➔ mengisialisasi nomor urut
- `while (current != nullptr) {` ➔ selama belum mencapai akhir antrian
- `cout << "| " << setw(2) << left << no << " | " << setw(15) << left << current->nama << " | " << setw(30) << left << current->nim << " | " << endl;` ➔ mencetak dan memanggil no, nama, dan nim
- `current = current->next;` ➔ pindah ke node berikutnya
- `no++;` ➔ `no++(increment)` menaik
- `cout << setfill('=') << setw(57) << "=" << setfill(' ') << endl;` ➔ `cout` menampilkan garis batas
- `int main() {` ➔ merupakan fungsi utama
- `Queue queue (5);` ➔ membuat antrian dengan batas 5
- `int pilihan;` ➔ deklarasi variabel untuk menyimpan nama pengguna
- `string nama, nim;` ➔ deklarasi variabel untuk menyimpan nama dan nim
- `do {` ➔ perulangan do while
- `cout << "\nMenu Operasi Antrian:\n";`

- `cout << setfill ('=') << setw (39) << "=" << setfill (' ') << endl;`  
➔ `cout` menampilkan garis pembatas
- `cout << "| " << setw (2) << left << "1" << " | " << "Tambahkan data" << "\n";` ➔ `cout` menampilkan pilihan menu tambahkan data
- `cout << "| " << setw (2) << left << "2" << " | " << "Hapus data" << "\n";` ➔ `cout` menampilkan pilihan menu Hapus data
- `cout << "| " << setw (2) << left << "3" << " | " << "Melihat data antrian di depan" << "\n";` ➔ `cout` menampilkan pilihan menu Hapus data Melihat data antrian di depan
- `cout << "| " << setw (2) << left << "4" << " | " << "Jumlah elemen antrian" << "\n";` ➔ `cout` menampilkan pilihan menu Jumlah elemen antrian
- `cout << "| " << setw (2) << left << "5" << " | " << "Melihat seluruh antrian" << "\n";` ➔ `cout` menampilkan Melihat seluruh antrian
- `cout << "| " << setw (2) << left << "6" << " | " << "Keluar" << "\n";` ➔ `cout` menampilkan Keluar
- `cout << setfill ('=') << setw (39) << "=" << setfill (' ') << endl;`  
➔ `cout` menampilkan garis pembatas
- `cout << "Pilih menu dari 1 sampai 6: ";` ➔ `cout` menampilkan pilih menu dari 1 sampai 6
- `cin >> pilihan;` ➔ menerima inputan dari pengguna dan menyimpan dalam variabel pilihan
- `cout << "\n";` ➔ `cout` mencetak baris baru
- `switch (pilihan){` ➔ switch case berdasarkan pilihan pengguna
- `case 1:` ➔ jika pengguna memilih 1
- `cout << "Masukkan nama : ";` ➔ `cout` menampilkan masukkan nama
- `cin >> nama;` ➔ menerima inputan dari pengguna dan menyimpan dalam variabel nama
- `cout << "Masukkan NIM : ";` ➔ `cout` menampilkan masukkan

nim

- cin >> nim; ➔ menerima inputan dari pengguna dan menyimpan dalam variabel nim
- queue.enqueue(nama, nim); ➔ menambahkan data ke antrian
- break; ➔ keluar dari case 1
- case 2: ➔ jika pengguna memilih 2
- queue.dequeue(); ➔ menghapus data dari antrian
- cout << "Data di depan telah dihapus." << endl; ➔ cout menampilkan data di depan telah dihapus
- break; ➔ keluar dari case 2
- case 3: ➔ jika pengguna memilih 3
- cout << "Data di depan: " << queue.peek() << endl; ➔ cout menampilkan data di depan dan dilanjutkan memanggil queue.peek()
- break; ➔ keluar dari case 3
- case 4: ➔ jika pengguna memilih 4
- cout << "Jumlah data dalam antrian : " << queue.size() << endl; ➔ cout menampilkan jumlah data dalam antrian
- break; ➔ keluar dari case 4
- case 5: ➔ jika pengguna memilih 5
- queue.viewQueue(); ➔ menampilkan seluruh antrian
- break; ➔ keluar dari case 5
- case 6: ➔ jika pengguna memilih 6
- cout << "Keluar dari program." << endl; ➔ cout menampilkan keluar dari program
- break; ➔ keluar dari case 5
- default: ➔ jika pengguna memilih selain pilihan menu 1-6
- cout << "Pilihan tidak valid. Silahkan coba lagi." << endl; ➔ cout menampilkan Pilihan tidak valid. Silahkan coba lagi

- `while (pilihan != 6);` ➔ melanjutkan perulangan selama tidak memilih pilihan menu 6
- `return 0;` ➔ program akan mengembalikan (return) nilai 0 ke

### **C. Kesimpulan**

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue.

Beberapa operasi umum pada queue:

- enqueue() : menambahkan data ke dalam queue.
- dequeue() : mengeluarkan data dari queue.
- peek() : mengambil data dari queue tanpa menghapusnya.
- isEmpty() : mengecek apakah queue kosong atau tidak.
- isFull() : mengecek apakah queue penuh atau tidak.
- size() : menghitung jumlah elemen dalam queue.



#### **D. Referensi**

[1] Asisten Praktikum, Struktur Data "QUEUE", WhatsApp, 2024.

[2] TylerMSFT. (n.d.) <queue Class>. diakses dari  
<https://learn.microsoft.com/en-us/cpp/standard-library/queue-class?view=msvc-170>

[3] TylerMSFT. (n.d.) <iomanip>. diakses dari  
<https://learn.microsoft.com/id-id/cpp/standard-library/iomanip?view=msvc-170>