

**LAPORAN PRAKTIKUM**  
**Modul VIII**  
**“QUEUE”**



**Disusun Oleh:**  
**Zivana Afra Yulianto -2211104039**  
**SE-07-02**

**Dosen:**  
**Wahyu Andi Saputra**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY**  
**PURWOKERTO**  
**2024**

## 1. Tujuan

Tujuan dari praktikum ini adalah untuk memahami dan mengimplementasikan konsep Queue (Antrian) menggunakan struktur data Linked List. Dalam praktikum ini, peserta diharapkan dapat:

1. Memahami cara kerja Queue sebagai struktur data FIFO (First In First Out).
2. Mempelajari cara implementasi Queue dengan menggunakan Linked List yang memungkinkan penyisipan dan penghapusan elemen secara dinamis tanpa batasan ukuran seperti array.
3. Menerapkan konsep prioritas antrian dengan mengurutkan elemen berdasarkan NIM mahasiswa yang lebih kecil didahulukan untuk keluar dari antrian.
4. Meningkatkan kemampuan dalam pemrograman dengan bahasa C++ dan penggunaan struktur data dinamis dalam implementasi antrian.

## 2. Landasan Teori

Queue (Antrian) adalah salah satu struktur data yang digunakan untuk menyimpan data secara berurutan dengan prinsip FIFO (First In First Out), yaitu elemen pertama yang masuk adalah elemen pertama yang keluar. Queue digunakan pada berbagai aplikasi seperti proses antrian dalam sistem operasi, pemrograman jaringan, dan simulasi antrian.

Secara umum, ada dua cara untuk mengimplementasikan Queue:

- Array: Menggunakan array untuk menyimpan elemen-elemen dalam antrian, namun memiliki keterbatasan ukuran tetap.
- Linked List: Menggunakan struktur data Node yang saling terhubung dengan pointer. Linked list memungkinkan penambahan dan penghapusan elemen secara dinamis tanpa batasan ukuran.

Pada praktikum ini, Linked List digunakan untuk mengimplementasikan Queue. Setiap node dalam linked list akan menyimpan data berupa Nama dan NIM mahasiswa. Setiap kali mahasiswa ditambahkan ke antrian, data mahasiswa akan disisipkan berdasarkan urutan NIM (yang lebih kecil didahulukan).

Fungsi-fungsi utama dalam implementasi ini:

- Enqueue: Menambahkan elemen ke antrian dengan urutan prioritas berdasarkan NIM.
- Dequeue: Menghapus elemen dari depan antrian.
- ViewQueue: Menampilkan seluruh elemen antrian.
- ClearQueue: Menghapus seluruh elemen dalam antrian.

### 3. Guided

#### GUIDED 1 :

```
#include <iostream>
#define MAX 100

using namespace std;

class Queue
{
private:
    int front, rear;
    int arr[MAX];

public:
    Queue()
    {
        front = -1;
        rear = -1;
    }

    bool isFull()
    {
        return rear == MAX - 1;
    }

    bool isEmpty()
    {
        return front == -1 || front > rear;
    }

    void enqueue(int x)
    {
        if (isFull())
        {
            cout << "Queue Overflow\n";
            return;
        }
        if (front == -1)
            front = 0;
        arr[++rear] = x;
    }

    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Queue Underflow\n";
            return;
        }
        front++;
    }

    int peek()
    {
        if (!isEmpty())
        {
            return arr[front];
        }
        cout << "Queue is empty\n";
        return -1;
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Queue is empty\n";
            return;
        }
    }
}
```

```

        for (int i = front; i <= rear; i++)
        {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main()
{
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Queue elements: ";
    q.display();

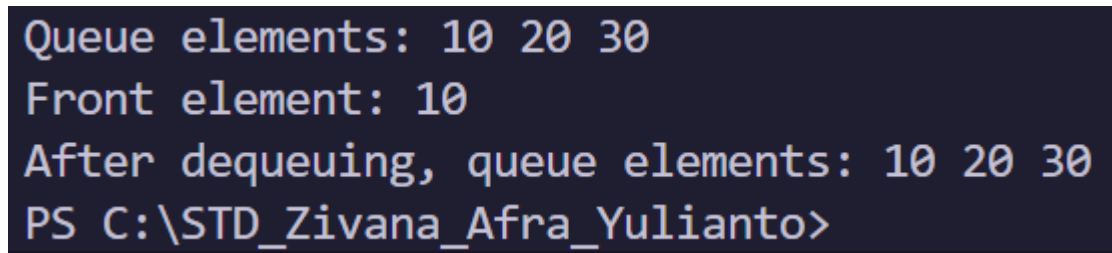
    cout << "Front element: " << q.peek() << "\n";

    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}

```

Screenshoot :



```

Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS C:\STD_Zivana_Afra_Yulianto>

```

Deskripsi :

1. Operasi Dasar:

- enqueue: Menambahkan elemen ke antrian.
- dequeue: Menghapus elemen dari antrian.
- peek: Mengambil elemen di depan antrian.
- display: Menampilkan semua elemen.

2. Pengecekan:

- isFull: Mengecek apakah antrian penuh.
- isEmpty: Mengecek apakah antrian kosong.

## GUIDED 2 :

```
#include <iostream>

using namespace std;

// Node untuk setiap elemen Queue
class Node
{
public:
    int data;    // Data elemen
    Node *next; // Pointer ke node berikutnya

    // Konstruktor untuk Node
    Node(int value)
    {
        data = value;
        next = nullptr;
    }
};

// Kelas Queue menggunakan linked list
class Queue
{
private:
    Node *front; // Pointer ke elemen depan Queue
    Node *rear;  // Pointer ke elemen belakang Queue

public:
    // Konstruktor Queue
    Queue()
    {
        front = rear = nullptr;
    }

    // Mengecek apakah Queue kosong
    bool isEmpty()
    {
        return front == nullptr;
    }

    // Menambahkan elemen ke Queue
    void enqueue(int x)
    {
        Node *newNode = new Node(x);
        if (isEmpty())
        {
            front = rear = newNode; // Jika Queue kosong
            return;
        }
        rear->next = newNode; // Tambahkan node baru ke belakang
        rear = newNode;      // Perbarui rear
    }

    // Menghapus elemen dari depan Queue
    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Queue Underflow\n";
            return;
        }
        Node *temp = front; // Simpan node depan untuk dihapus
        front = front->next; // Pindahkan front ke node berikutnya
        delete temp;        // Hapus node lama
        if (front == nullptr) // Jika Queue kosong, rear juga harus null
            rear = nullptr;
    }
}
```

```

// Mengembalikan elemen depan Queue tanpa menghapusnya
int peek()
{
    if (!isEmpty())
    {
        return front->data;
    }
    cout << "Queue is empty\n";
    return -1; // Nilai sentinel
}

// Menampilkan semua elemen di Queue
void display()
{
    if (isEmpty())
    {
        cout << "Queue is empty\n";
        return;
    }
    Node *current = front; // Mulai dari depan
    while (current)
    { // Iterasi sampai akhir
        cout << current->data << " ";
        current = current->next;
    }
    cout << "\n";
}

};

// Fungsi utama untuk menguji Queue
int main()
{
    Queue q;

    // Menambahkan elemen ke Queue
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    // Menampilkan elemen di Queue
    cout << "Queue elements: ";
    q.display();

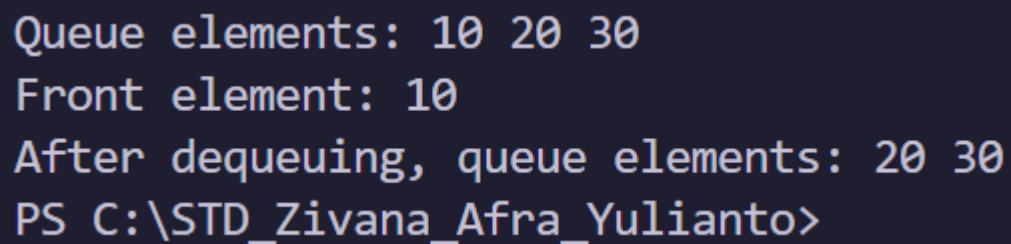
    // Menampilkan elemen depan
    cout << "Front element: " << q.peek() << "\n";

    // Menghapus elemen dari depan Queue
    q.dequeue();
    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}

```

**Screenshoot :**



```

Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS C:\STD_Zivana_Afra_Yulianto>

```

### Deskripsi :

- Node: Menyimpan data dan pointer ke elemen berikutnya.
- Queue Operations:
  - enqueue: Menambahkan elemen ke akhir Queue.
  - dequeue: Menghapus elemen dari depan Queue.
  - peek: Mengambil elemen depan tanpa menghapusnya.
  - display: Menampilkan semua elemen.
  - isEmpty: Mengecek apakah Queue kosong.

### GUIDED 3 :

```
#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
        { // Antriannya 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;
}

```



## Screenshoot :

```
Data antrian teller:
1. Andi
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
Antrian kosong
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS C:\STD_Zivana_Afra_Yulianto>
```

## Deskripsi :

Code ini adalah implementasi Queue berbasis array di C++ untuk mengelola antrian teller dengan kapasitas tetap (5 elemen).

Fitur:

- Operasi Utama:  
enqueueAntrian, dequeueAntrian, viewQueue, countQueue, clearQueue.
- Pengecekan:  
isFull (penuh) dan isEmpty (kosong).

## 4. Unguided

### UNGUIDED 1

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

using namespace std;

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

Node *front = nullptr;
Node *back = nullptr;

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

void enqueueAntrian(string data)
{
    Node *newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;

    if (isEmpty())
    {
        front = back = newNode;
    }
    else
    {
        back->next = newNode;
        back = newNode;
    }
}

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        front = front->next;
        if (front == nullptr)
            back = nullptr; // Jika kosong
        delete temp;
    }
}

void viewQueue()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        cout << "Data antrian:" << endl;
        while (temp != nullptr)
        {
            cout << temp->data << endl;
            temp = temp->next;
        }
    }
}

void clearQueue()
{
    while (!isEmpty())
    {
        dequeueAntrian();
    }
}
```

```
int main()
{
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");

    viewQueue();

    dequeueAntrian();
    viewQueue();

    clearQueue();
    viewQueue();

    return 0;
}
```

### Screenshoot output :



```
Data antrian:
Andi
Maya
Data antrian:
Maya
Antrian kosong
PS C:\STD_Zivana_Afra_Yulianto>
```

### Deskripsi :

Program ini mengimplementasikan Queue menggunakan Linked List. Fitur utama:

1. Enqueue: Menambahkan elemen ke belakang antrian.
2. Dequeue: Menghapus elemen dari depan antrian.
3. ViewQueue: Menampilkan isi antrian.
4. ClearQueue: Menghapus semua elemen antrian.
5. isEmpty: Mengecek apakah antrian kosong.

## UNGUIDED 2

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

using namespace std;

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

Node *front = nullptr;
Node *back = nullptr;

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

void enqueueAntrian(string nama, string nim)
{
    Node *newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

    if (isEmpty())
    {
        front = back = newNode;
    }
    else
    {
        back->next = newNode;
        back = newNode;
    }
}

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        front = front->next;
        if (front == nullptr)
            back = nullptr;
        delete temp;
    }
}

void viewQueue()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        cout << "Data antrian:" << endl;
        while (temp != nullptr)
        {
            cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
            temp = temp->next;
        }
    }
}

void clearQueue()
{
    while (!isEmpty())
    {
        dequeueAntrian();
    }
}
```

```

int main()
{
    enqueueAntrian("Andi", "210123");
    enqueueAntrian("Maya", "210122");

    viewQueue();

    dequeueAntrian();
    viewQueue();

    clearQueue();
    viewQueue();

    return 0;
}

```

Screenshoot output :

```

Data antrian:
Nama: Andi, NIM: 210123
Nama: Maya, NIM: 210122
Data antrian:
Nama: Maya, NIM: 210122
Antrian kosong
PS C:\STD_Zivana_Afra_Yulianto>

```

### Deskripsi :

Fitur utama:

1. Enqueue: Menambahkan mahasiswa ke antrian.
2. Dequeue: Menghapus mahasiswa dari antrian (dari depan).
3. ViewQueue: Menampilkan semua mahasiswa dalam antrian.
4. ClearQueue: Menghapus seluruh antrian.
5. isEmpty: Mengecek apakah antrian kosong.

### GUIDED 3 :

```

#include <iostream>
#include <string>

using namespace std;

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

Node *front = nullptr;

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

```

```

void enqueueAntrian(string nama, string nim)
{
    Node *newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

    if (isEmpty() || nim < front->nim)
    {
        newNode->next = front;
        front = newNode;
    }
    else
    {
        Node *temp = front;
        while (temp->next != nullptr && temp->next->nim < nim)
        {
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        front = front->next;
        delete temp;
    }
}

void viewQueue()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        cout << "Data antrian:" << endl;
        while (temp != nullptr)
        {
            cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
            temp = temp->next;
        }
    }
}

void clearQueue()
{
    while (!isEmpty())
    {
        dequeueAntrian();
    }
}

int main()
{
    string nama, nim;
    int n;

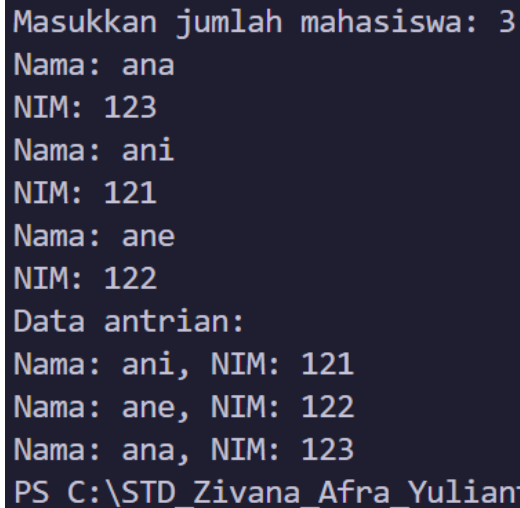
    cout << "Masukkan jumlah mahasiswa: ";
    cin >> n;
}

```

```
for (int i = 0; i < n; i++)
{
    cout << "Nama: ";
    cin >> nama;
    cout << "NIM: ";
    cin >> nim;
    enqueueAntrian(nama, nim);
}

viewQueue();
return 0;
}
```

### Screenshoot Output :



```
Masukkan jumlah mahasiswa: 3
Nama: ana
NIM: 123
Nama: ani
NIM: 121
Nama: ane
NIM: 122
Data antrian:
Nama: ani, NIM: 121
Nama: ane, NIM: 122
Nama: ana, NIM: 123
PS C:\STD_Zivana_Afra_Yulianto>
```

### Deskripsi :

Fitur utama:

1. Enqueue: Menambahkan mahasiswa ke antrian dengan prioritas berdasarkan NIM.
2. Dequeue: Menghapus mahasiswa dari antrian (dari depan).
3. ViewQueue: Menampilkan semua mahasiswa dalam antrian.
4. ClearQueue: Menghapus seluruh antrian.
5. isEmpty: Mengecek apakah antrian kosong.

## 5. Kesimpulan

Dalam praktikum ini, peserta berhasil mengimplementasikan Queue menggunakan Linked List dengan kemampuan untuk mengurutkan elemen berdasarkan NIM mahasiswa, yang menjadikan mahasiswa dengan NIM lebih kecil didahulukan untuk keluar dari antrian.

Beberapa hal yang dapat disimpulkan dari praktikum ini adalah:

1. Queue dengan menggunakan Linked List sangat fleksibel karena tidak dibatasi ukuran tetap seperti array, dan operasi seperti penyisipan dan penghapusan elemen dapat dilakukan dengan efisien.
2. Penerapan prioritas antrian dengan mengurutkan berdasarkan NIM memberikan kemampuan untuk memanipulasi urutan elemen sesuai kebutuhan.
3. Linked List sebagai struktur data dinamis memungkinkan penanganan data dalam jumlah besar secara efisien tanpa memerlukan alokasi memori yang tetap, yang merupakan keuntungan besar dibandingkan dengan penggunaan array.

Secara keseluruhan, praktikum ini memberikan pemahaman yang lebih mendalam tentang konsep Queue, implementasi Linked List, dan pengaplikasiannya dalam menyelesaikan permasalahan nyata terkait antrian dan prioritas data.