

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VII  
PENGENALAN QUEUE**



**Disusun Oleh :**

NAMA : Pandia Arya Brata

NIM : 2311102076

**Dosen**

Wahyu Andi Saputra, S.Pd.,M.Eng.

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

## **MODUL VIII**

### **QUEUE**

#### **A. TUJUAN PRAKTIKUM**

1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

## B. 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. 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.

### C. GUIDED

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

const int maksimalQueue = 5;
int front = 0;
int back = 0;
string queueTeller[5];
bool isFull()
{
    if (back == maksimalQueue)
    {
        return true; // =1
    }
    else
    {
        return false;
    }
}

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

```

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

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian 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 << "Antrian kosong" << endl;
    }
    else
    {

```

```

        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

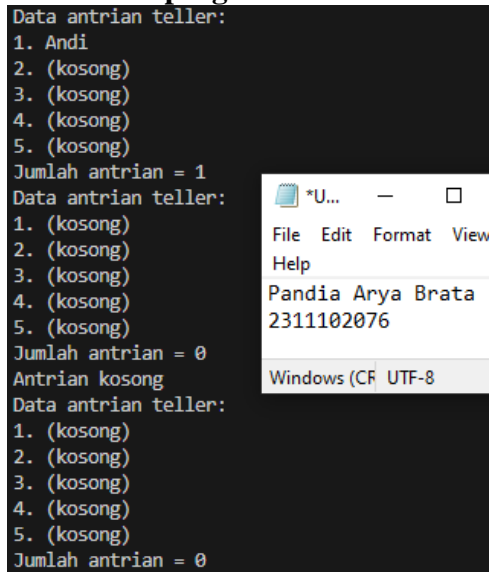
void viewQueue()
{
    cout << "Data antrian teller:" << endl;
    for (int i = 0; i < maksimalQueue; i++)
    {
        if (queueTeller[i] != "")
        {
            cout << i + 1 << ". " << queueTeller[i] <<

                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;
}

```

### Screenshot program :



```
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
```

### Deskripsi program :

Program ini mensimulasikan sebuah antrian untuk teller. Program ini dapat digunakan untuk memindahkan (menambah) nasabah ke dalam antrian, memisahkan (menghapus) nasabah dari antrian, menentukan apakah antrian tersebut lancar atau macet, memanipulasi antrian, dan melihat gambar antrian. Untuk menentukan panjang antrian maksimum, program ini terlebih dahulu mendefinisikan panjang antrian minimum. Selanjutnya, deklarasikan tiga variabel integer: front, back, dan queueTeller. Front digunakan untuk menyelaraskan tepi depan antrian, back digunakan untuk menyelaraskan tepi belakang antrian, dan queueTeller adalah sebuah string array yang menyelaraskan data pengguna di dalam antrian.

#### D. UNGUIDED

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

**Source Code :**

```
#include <iostream>

using namespace std;

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

class Queue
{
private:
    Node *front;
    Node *back;

public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }

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

    bool isFull()
    {
        return false;
    }

    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;
        delete temp;

        if (front == nullptr)
        {
            back = nullptr;
        }
    }
}

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

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

void viewQueue()
{
    cout << "Data antrian teller:" << endl;
    Node *current = front;

```

```

        int i = 1;
        while (current != nullptr)
        {
            cout << i << ". " << current->data << endl;
            current = current->next;
            i++;
        }

        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
    }
};

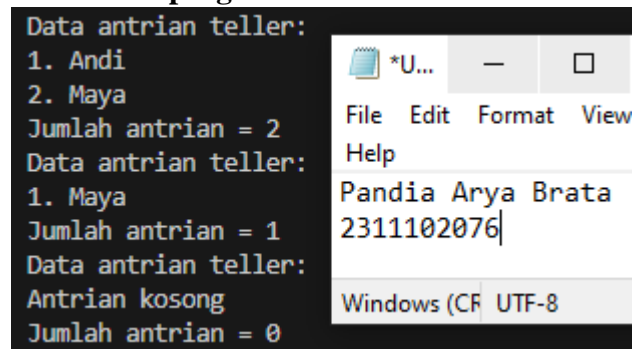
int main()
{
    Queue antrianTeller;

    antrianTeller.enqueueAntrian("Andi");
    antrianTeller.enqueueAntrian("Maya");
    antrianTeller.viewQueue();
    cout << "Jumlah antrian = " << antrianTeller.countQueue() <<
endl;
    antrianTeller.dequeueAntrian();
    antrianTeller.viewQueue();
    cout << "Jumlah antrian = " << antrianTeller.countQueue() <<
endl;
    antrianTeller.clearQueue();
    antrianTeller.viewQueue();
    cout << "Jumlah antrian = " << antrianTeller.countQueue() <<
endl;

    return 0;
}

```

#### Screenshot program :



```

Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Antrian kosong
Jumlah antrian = 0

```

**Deskripsi program :**

diatas adalah program dari guided 1 yang menggunakan konsep queue yang di ubah dari array menjadi linked list

2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

**Source code :**

```
#include <iostream>

using namespace std;

struct Student
{
    string name;
    string nim;
};

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

class StudentQueue
{
private:
    Node *front;
    Node *back;

public:
    StudentQueue()
    {
        front = nullptr;
        back = nullptr;
    }

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

    void enqueue(Student student)
    {
        Node *newNode = new Node;
        newNode->data = student;
        newNode->next = nullptr;
```

```

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

Student dequeue()
{
    if (isEmpty())
    {
        cout << "Antrian mahasiswa kosong" << endl;
        return Student{"", ""};
    }
    else
    {
        Node *temp = front;
        Student student = front->data;
        front = front->next;
        delete temp;

        if (front == nullptr)
        {
            back = nullptr;
        }
        return student;
    }
}

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

void clear()
{
    while (!isEmpty())

```

```

        {
            dequeue();
        }
    }

void view()
{
    cout << "Data antrian mahasiswa:" << endl;
    Node *current = front;
    int i = 1;
    while (current != nullptr)
    {
        cout << "Nama : " << current->data.name << endl;
        cout << "NIM : " << current->data.nim << endl;
        cout << endl;
        current = current->next;
        i++;
    }

    if (isEmpty())
    {
        cout << "Antrian mahasiswa kosong" << endl;
    }
}

};

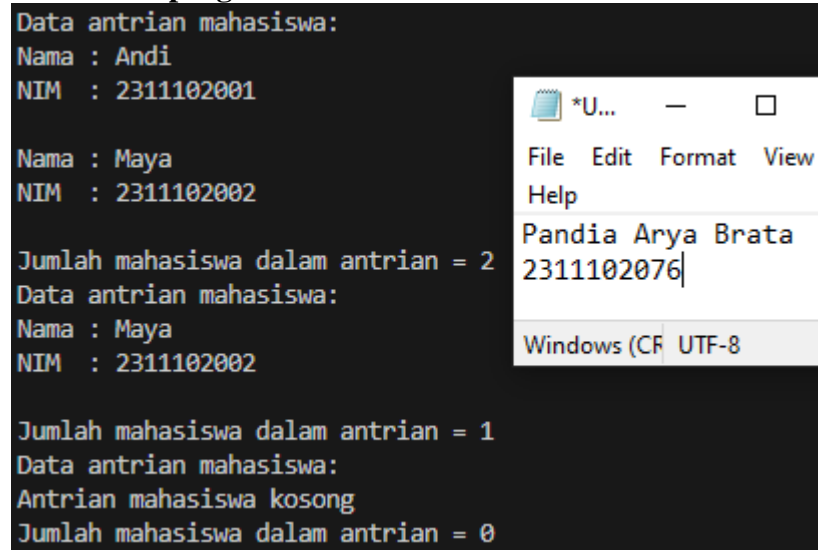
int main()
{
    StudentQueue studentQueue;

    studentQueue.enqueue(Student{"Andi", "2311102001"});
    studentQueue.enqueue(Student{"Maya", "2311102002"});
    studentQueue.view();
    cout << "Jumlah mahasiswa dalam antrian = " <<
studentQueue.count() << endl;
    studentQueue.dequeue();
    studentQueue.view();
    cout << "Jumlah mahasiswa dalam antrian = " <<
studentQueue.count() << endl;
    studentQueue.clear();
    studentQueue.view();
    cout << "Jumlah mahasiswa dalam antrian = " <<
studentQueue.count() << endl;

    return 0;
}

```

### Screenshot program:



The screenshot shows a terminal window with a black background and white text. The text displays the execution of a program that manages a queue of students. It starts by adding two students, 'Andi' and 'Maya', to the queue. Then, it prints the number of students in the queue (2) and the data of the students. Next, it prints the number of students in the queue (1) and the data of the student. Finally, it prints 'Antrian mahasiswa kosong' (Queue empty) and the number of students in the queue (0). A context menu is open over the terminal, showing options like File, Edit, Format, View, Help, and a search bar. The search bar contains the text 'Pandia Arya Brata 2311102076'.

```
Data antrian mahasiswa:  
Nama : Andi  
NIM : 2311102001  
  
Nama : Maya  
NIM : 2311102002  
  
Jumlah mahasiswa dalam antrian = 2  
Data antrian mahasiswa:  
Nama : Maya  
NIM : 2311102002  
  
Jumlah mahasiswa dalam antrian = 1  
Data antrian mahasiswa:  
Antrian mahasiswa kosong  
Jumlah mahasiswa dalam antrian = 0
```

### Deskripsi program :

program ini mendemonstrasikan implementasi dasar dari sebuah antrian siswa dengan menggunakan struktur data linked list. Hal ini memungkinkan untuk menambah, menghapus, memeriksa ukuran, dan melihat informasi tentang siswa yang sedang mengantri.

## **E. DAFTAR PUSTAKA**

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.