

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 7
QUEUE**



Disusun Oleh:

Muhamad ihsan

2311102077

Dosen

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

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

Modul 7

A. Tujuan

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.

Queue memiliki peran yang penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrian tugas atau operasi secara efisien. Dalam sistem komputasi, ia digunakan untuk menangani tugas-tugas seperti penjadwalan proses, antrian pesan, dan manajemen sumber daya.

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.

Jenis-jenis Queue

Jenis struktur data ini dapat diklasifikasikan berdasarkan cara implementasinya, maupun berdasarkan penggunaannya. Di antara jenis-jenis tersebut adalah sebagai berikut.

1. Berdasarkan Implementasinya
 - Linear/Simple Queue: Elemen-elemen data disusun dalam barisan linear dan penambahan serta penghapusan elemen hanya terjadi pada dua ujung barisan tersebut.
 - Circular Queue: Mirip dengan jenis linear, tetapi ujung-ujung barisan terhubung satu sama lain, menciptakan struktur antrean yang berputar.
2. Berdasarkan Penggunaan
 - Priority Queue: Setiap elemen memiliki prioritas tertentu. Elemen dengan prioritas tertinggi akan diambil terlebih dahulu.
 - Double-ended Queue (Deque): Elemen dapat ditambahkan atau dihapus dari kedua ujung antrean.

Keuntungan

- Data berjumlah besar dapat dikelola dengan mudah dan efisien.
- Proses insert dan delete data dapat dilakukan dengan mudah karena mengikuti aturan first in first out (FIFO).
- Efisien dalam menangani tugas berdasarkan urutan kedatangan.

Keterbatasan

- Tidak efisien untuk pencarian elemen tertentu dalam antrean.
- Memerlukan alokasi memori yang cukup untuk menyimpan antrean.

GUIDED 1

```
#include <iostream>
using namespace std;
// queue array
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;
    }
}
// fungsi pengecekan
bool isEmpty()
{ // antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
// fungsi menambahkan antrian
void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "antrian penuh" << endl;
    }
    else
    { // nested if, nested for
        if (isEmpty())
        { // kondisi ketika queue kosong
            queueTeller[0] = data;
            front++; // front = front +1;
            back++;
        }
        else
        {
            queueTeller[back] = data; // antriannya ada isi // queueTeller[1]=data
            back++; // back=back+1; 2
        }
    }
}
// fungsi mengurangi antrian
void dequeueAntrian()
```

```

{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}
// fungsi menghitung banyak antrian
int countQueue()
{
    return back;
}
// fungsi menghapus semua antrian
void clearQueue()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}
// fungsi melihat antrian
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");
enqueueAntrian("Budi");
viewQueue();
cout << "jumlah antrian = " << countQueue() << endl;

dequeueAntrian();
viewQueue();

enqueueAntrian("Ucup");
enqueueAntrian("Umar");
viewQueue();

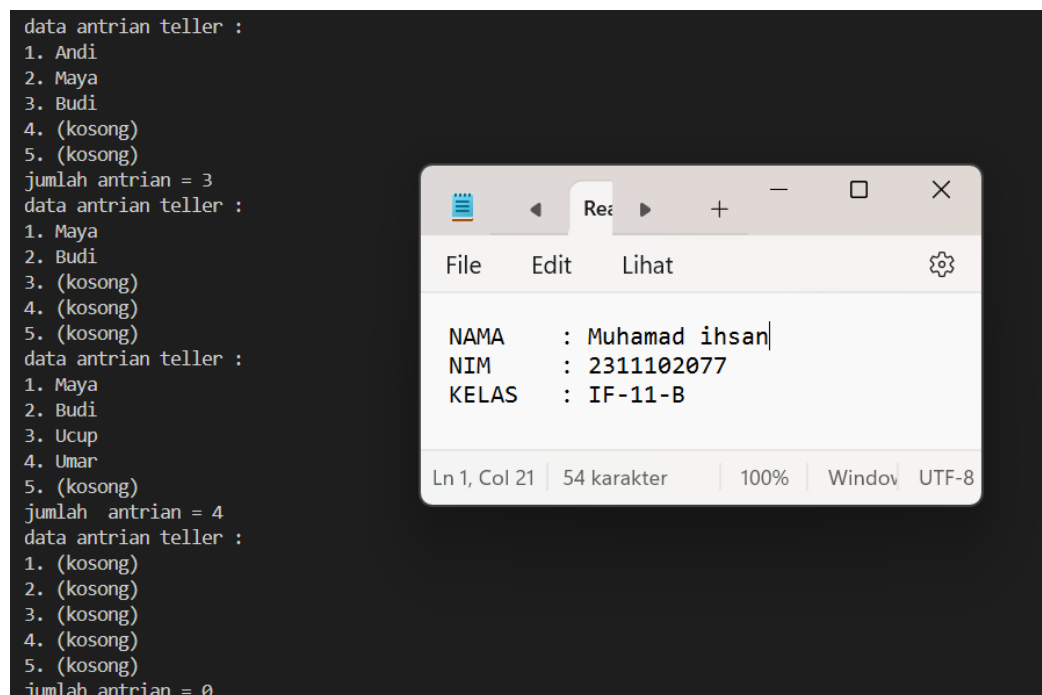
cout << "jumlah antrian = " << countQueue() << endl;

clearQueue();
viewQueue();

cout << "jumlah antrian = " << countQueue() << endl;
return 0;
}

```

Output:



The screenshot shows a terminal window on the left and a Notepad++ window on the right. The terminal displays the output of a program that manages a queue of names. It shows three states: initial state with 'Andi', 'Maya', and 'Budi'; after dequeuing 'Andi' with 'Maya' and 'Budi'; and after enqueueing 'Ucup' and 'Umar' with all four names. The Notepad++ window shows a text file with student information: NAMA: Muhamad ihsan, NIM: 2311102077, and KELAS: IF-11-B.

```

data antrian teller :
1. Andi
2. Maya
3. Budi
4. (kosong)
5. (kosong)
jumlah antrian = 3
data antrian teller :
1. Maya
2. Budi
3. (kosong)
4. (kosong)
5. (kosong)
jumlah antrian = 2
data antrian teller :
1. Maya
2. Budi
3. Ucup
4. Umar
5. (kosong)
jumlah antrian = 4
data antrian teller :
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
jumlah antrian = 0

```

File Edit Lihat

NAMA : Muhamad ihsan
NIM : 2311102077
KELAS : IF-11-B

Ln 1, Col 21 | 54 karakter | 100% | Window UTF-8

Deskripsi:

Pada program di atas membuat struktur data antrian menggunakan linked list, Setiap elemen dalam antrian direpresentasikan oleh Node yang memiliki dua atribut : nama dan nim. Setiap Node juga memiliki pointer next yang menunjuk ke elemen berikutnya dalam antrian. Implementasi fungsi antrian meliputi enqueue (menambah antrian), dequeue (menghapus antrian), lihat antrian, jumlah antrian, dan hapus semua antrian. Program utama menampilkan menu pilihan untuk pengguna, termasuk opsi untuk menambah, menghapus, dan melihat antrian.

C. UNGUIDED

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>
using namespace std;

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

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

public:
    Queue() {
        front = nullptr;
        back = nullptr;
        size = 0;
    }

    bool isFull() {
        return false; // Linked list based queue is never full unless out of memory
    }

    bool isEmpty() {
        return size == 0;
    }

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

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        }
    }
};
```

```

    } else {
        back->next = newNode;
        back = newNode;
    }
    size++;
}

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

int countQueue() {
    return size;
}

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

void viewQueue() {
    cout << "Data antrian mahasiswa:" << endl;
    Node* current = front;
    int position = 1;
    while (current != nullptr) {
        cout << position << ". " << current->nama << " (" << current->nim << ")"
<< endl;
        current = current->next;
        position++;
    }
}

};

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

    do {
        cout << "Menu Antrian Mahasiswa:" << endl;
        cout << "1. Tambah Antrian" << endl;

```



```

    cout << "2. Hapus Antrian" << endl;
    cout << "3. Lihat Antrian" << endl;
    cout << "4. Jumlah Antrian" << endl;
    cout << "5. Hapus Semua Antrian" << endl;
    cout << "0. Keluar" << endl;
    cout << "Pilih menu: ";
    cin >> pilihan;

    switch (pilihan) {
    case 1:
        cout << "Masukkan Nama: ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM: ";
        getline(cin, nim);
        antrian.enqueueAntrian(nama, nim);
        break;
    case 2:
        antrian.dequeueAntrian();
        break;
    case 3:
        antrian.viewQueue();
        break;
    case 4:
        cout << "Jumlah antrian = " << antrian.countQueue() << endl;
        break;
    case 5:
        antrian.clearQueue();
        break;
    case 0:
        cout << "Keluar dari program." << endl;
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
    } while (pilihan != 0);

    return 0;
}

```

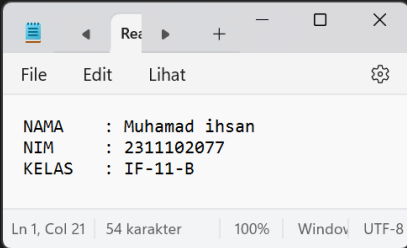
SCREENSHOT OUTPUT

1. Menu

Menu Antrian Mahasiswa:	File	Edit	Lihat
1. Tambah Antrian			
2. Hapus Antrian			
3. Lihat Antrian	NAMA	:	Muhamad ihsan
4. Jumlah Antrian	NIM	:	2311102077
5. Hapus Semua Antrian	KELAS	:	IF-11-B
0. Keluar			

2. Tambah

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 1
Masukkan Nama: grealish
Masukkan NIM: 2311156
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 1
Masukkan Nama: jackbrown
Masukkan NIM: 2311157
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 1
Masukkan Nama: bagot
Masukkan NIM: 2311158
```



3. Hapus

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 2
```

	File	Edit	Lihat
NAMA	:	Muhamad ihsan	
NIM	:	2311102077	
KELAS	:	IF-11-B	

4. Lihat

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 3
Data antrian mahasiswa:
1. jackbrown (2311157)
2. bagot (2311158)
```

	File	Edit	Lihat
NAMA	:	Muhamad ihsan	
NIM	:	2311102077	
KELAS	:	IF-11-B	

5. Jumlah

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 4
Jumlah antrian = 2
```

	File	Edit	Lihat
NAMA	:	Muhamad ihsan	
NIM	:	2311102077	
KELAS	:	IF-11-B	

6. Hapus semua

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu: 5
```

	File	Edit	Lihat
NAMA	:	Muhamad ihsan	
NIM	:	2311102077	
KELAS	:	IF-11-B	

DESKRIPSI:

Program di atas mengubah penerapan konsep queue pada bagian guided dari array menjadi linked list dan membuat konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

D. KESIMPULAN

“Queue (antrian) adalah struktur data yang mengikuti aturan FIFO (First-In First-Out). Elemen pertama yang dimasukkan ke dalam queue akan menjadi elemen pertama yang dikeluarkan. Implementasi queue dapat menggunakan array atau linked list, dengan dua pointer: front (menunjuk ke elemen pertama) dan rear (menunjuk ke elemen terakhir). Perbedaan antara stack dan queue terletak pada aturan penambahan dan penghapusan elemen. Operasi pada queue meliputi enqueue() (menambahkan data), dequeue() (mengeluarkan data), peek() (mengambil data tanpa menghapusnya), isEmpty() (memeriksa apakah queue kosong), isFull() (memeriksa apakah queue penuh), dan size() (menghitung jumlah elemen). Jenis-jenis queue meliputi linear/simple queue, circular queue, priority queue, dan double-ended queue (dequeue).”

E. REFERENSI

[1] Asprak “Modul 7 Queue”. Learning Meaning System

[2] Dicoding

<https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/>

[3] Arif. 2014. Queue, <http://kabularif17.blogspot.co.id/2014/06/queue.html>.