

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

NAMA : Tegar Serli Arunzika

NIM : 2311102041

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. 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.

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 belakang 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

B. Guided

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
    {
        // antrianya ada isi
        queueTeller[back] = data; // 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;

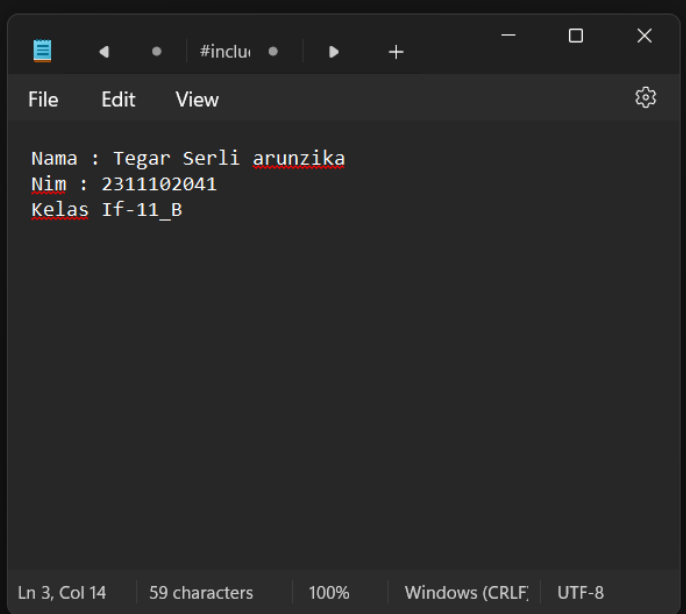
    clearQueue();
    viewQueue();

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

```

Screenshots Output

```
PS C:\Users\M S I> & 'c:\Users\M S I\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebu
ngine-In-2hlrgs5p.2ml' '--stdout=Microsoft-MIEngine-Out-r01aickr.czt' '--stderr=Microsoft-MIEngine-Error-usnk4ldi.0u3' '--
pq' '--dbgExe=C:\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'
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)
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
PS C:\Users\M S I>
```



```
Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_B
```

Deskripsi:

Kode di atas merupakan implementasi dasar dari antrian menggunakan array di C++. Meskipun berfungsi, terdapat beberapa area untuk peningkatan, seperti penggunaan front yang sebenarnya tidak terlalu digunakan dalam kode saat ini dan cara penghapusan elemen yang bisa dioptimalkan.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

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

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

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

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

    bool isFull()
    {
        return false;
    }

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

```

void enqueueAntrian(string data)
{
    Node *newNode = new Node();
    newNode->data = data;
    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--;
        if (isEmpty())
        {
            back = nullptr;
        }
    }
}

int countQueue()
{
    return size;
}

void clearQueue()
{

```



```

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

void viewQueue()
{
    cout << "data antrian teller : " << endl;
    Node *temp = front;
    int index = 1;
    while (temp != nullptr)
    {
        cout << index << ". " << temp->data << endl;
        temp = temp->next;
        index++;
    }
    for (; index <= 5; index++)
    {
        cout << index << ". (kosong)" << endl;
    }
}

};

int main()
{
    Queue queue;
    queue.enqueueAntrian("Andi");
    queue.enqueueAntrian("Maya");
    queue.enqueueAntrian("Budi");
    queue.viewQueue();
    cout << "jumlah antrian = " << queue.countQueue() << endl;

    queue.dequeueAntrian();
    queue.viewQueue();

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

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

    queue.clearQueue();
    queue.viewQueue();
}

```

```

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

}

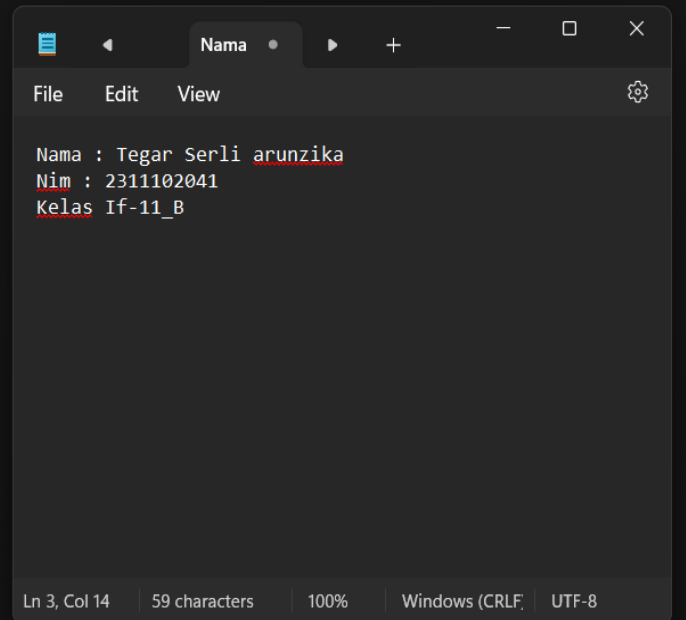
```

Screenshots Output

```

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)
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
PS C:\Users\M S I>

```



Deskripsi:

Kode diatas menunjukkan implementasi dasar dari antrian menggunakan linked list dalam C++, dengan operasi dasar seperti enqueue, dequeue, dan melihat isi antrian.

Unguided 2

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

using namespace std;

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

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

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

    bool isEmpty()
    {
        return count == 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;
        }
        count++;
    }

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

int countQueue()
{
    return count;
}

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

```

```

void viewQueue()
{
    cout << "Data antrian mahasiswa: " << endl;
    Node *temp = front;
    int index = 1;
    while (temp != nullptr)
    {
        cout << index << ". Nama: " << temp->nama << ", NIM: " <<
temp->nim << endl;
        temp = temp->next;
        index++;
    }
    if (isEmpty())
    {
        cout << "(Antrian kosong)" << endl;
    }
}
};

void showMenu()
{
    cout << "\nMenu Antrian Mahasiswa:\n";
    cout << "1. Tambah Antrian\n";
    cout << "2. Hapus Antrian\n";
    cout << "3. Lihat Antrian\n";
    cout << "4. Jumlah Antrian\n";
    cout << "5. Hapus Semua Antrian\n";
    cout << "6. Keluar\n";
    cout << "Pilih opsi: ";
}

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

    do
    {
        showMenu();
        cin >> choice;
        cin.ignore();

        switch (choice)
        {

```

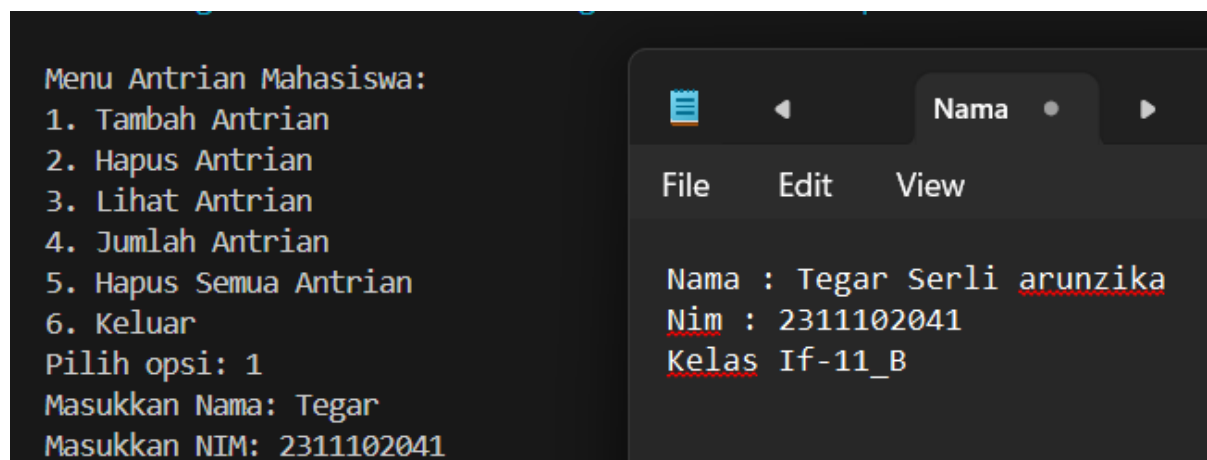
```

        case 1:
            cout << "Masukkan Nama: ";
            getline(cin, nama);
            cout << "Masukkan NIM: ";
            getline(cin, nim);
            queue.enqueueAntrian(nama, nim);
            break;
        case 2:
            queue.dequeueAntrian();
            break;
        case 3:
            queue.viewQueue();
            break;
        case 4:
            cout << "Jumlah antrian: " << queue.countQueue() << endl;
            break;
        case 5:
            queue.clearQueue();
            cout << "Semua antrian telah dihapus." << endl;
            break;
        case 6:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Opsi tidak valid. Silakan coba lagi." << endl;
    }
} while (choice != 6);

return 0;
}
s

```

Screenshots Outpu

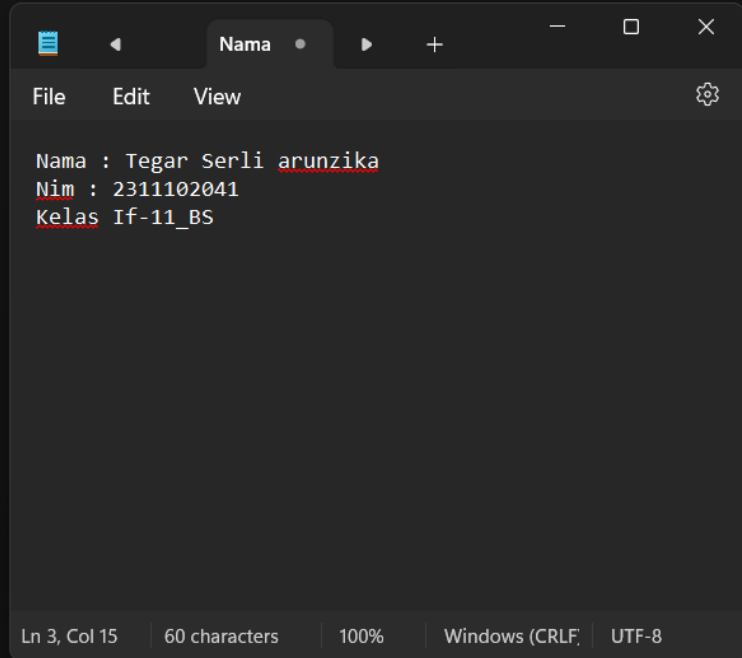


Pilih opsi: 1
Masukkan Nama: yudis
Masukkan NIM: 564648845

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 1
Masukkan Nama: yunita
Masukkan NIM: 65946548

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 2

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 3
Data antrian mahasiswa:
1. Nama: yudis, NIM: 564648845
2. Nama: yunita, NIM: 65946548



```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 4
Jumlah antrian: 2

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 8
Opsi tidak valid. Silakan coba lagi.

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih opsi: 6
Keluar dari program.
PS C:\Users\M S I>
```

Nama

File Edit View

Nama : Tegar Serli arunzika
Nim : 2311102041
Kelas If-11_BS

Ln 3, Col 15 | 60 characters | 100% | Windows (CRLF)

Deskripsi:

Kode di atas adalah implementasi dari antrian (queue) menggunakan struktur data linked list untuk menyimpan data mahasiswa dengan atribut Nama dan NIM.

D. Kesimpulan

Queue merupakan kumpulan data dengan penambahan data hanya melalui satu sisi, yaitu belakang (tail) dan penghapusan data hanya melalui sisi depan (head). Berbeda dengan stack yang bersifat LIFO maka queue bersifat FIFO(First In First Out), yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yang terakhir masuk akan keluar terakhir. Berikut ini adalah gambaran struktur data queueReferensi

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