

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VII  
QUEUE**



**Disusun Oleh :**

NAMA : Ahmad Titana Nanda Pramudya  
NIM : 2311102042

**Dosen**

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

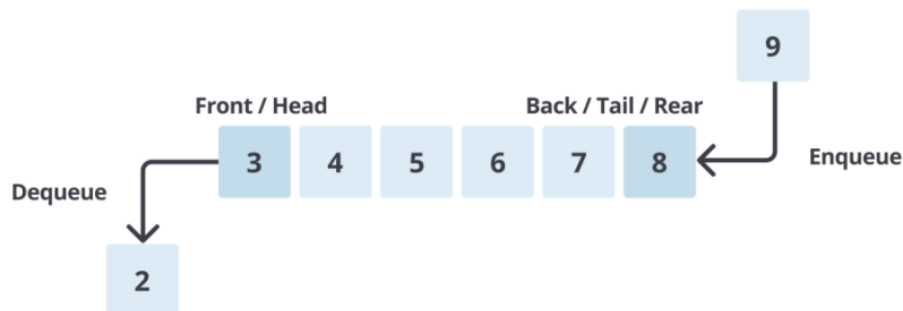
**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## 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.



**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

Queue (Antrian) adalah suatu bentuk khusus dari List Linier dengan operasi penyisipan (insertion) hanya diperbolehkan pada salah satu sisi, yang disebut sisibelakang (REAR), dan operasi penghapusan (deletion) hanya diperbolehkan pada sisiyang lainnya. 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 InFirst Out), yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yangterakhir masuk akan keluar terakhir. Hal yang perlu diingat :

1. Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisiyang lain.
2. Queue bersifat FIFO (First In First Out).

Elemen yang pertama kali masuk ke dalam queue disebut elemen depan (front/head of queue), sedangkan elemen yang terakhir kali masuk ke queue disebut elemen belakang(rear/tail of queue). Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan danpenghapusan elemen dilakukan di satu ujung. Elemen yang terakhir kali dimasukkanakan berada paling dekat dengan ujung atau dianggap paling atas sehingga padaoperasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO. Pada queue,operasi tersebut dilakukan di tempat yang berbeda.Penambahan elemen selalu dilakukan melalui salah satu ujung, menempati posisi dibelakang elemen - elemen yang sudah masuk sebelumnya atau menjadi elemen palingbelakang. Sedangkan penghapusan elemen dilakukan di ujung yang berbeda, yaitu pada posisi elemen yang masuk paling awal atau elemen terdepan. Sifat yang demikiandikenal dengan FIFO.

Operasi – operasi pada queue :

1. Deklarasi

Pendeklarasian queue menggunakan tipe data struct..

```
struct Queue {  
    int data[MAX];  
    int head;  
    int tail;  
} antrian;
```

## 2. Creat

Operasi untuk membuat dan menginisialisasi sebuah queue kosong dengan cara membuat Head dan Tail = -1.

```
void create(){
    antrian.head =
    antrian.tail = -1;
}
```

## 3. isEmpty

Operasi untuk memeriksa apakah suatu queue masih kosong. Queue kosong ditandai dengan nilai Tail kurang dari nol (-1).

```
int isEmpty()
{
    if (antrian.tail == -1)
        return 1; //true
    else
        return 0; //false
}
```

## 4. isFull

Operasi untuk memeriksa apakah queue yang ada sudah penuh. Queue akan mengindikasikan penuh jika ujung antrian (tail) mendekati nilai maksimum yang dapat ditampung antrian (MAX-1).

```
int isFull()
{
    if (antrian.tail == MAX-
    1)
        return 1; //true
    else
```

```
return 0; //false  
}
```

## 5. Enqueue

Operasi untuk menambahkan satu elemen ke dalam queue dan tidak dapat dilakukan jika antrian dalam keadaan penuh.

```
void enqueue (int data)  
{  
    if (isEmpty() == 1){  
        antrian.head = antrian.tail = 0;  
        antrian.data[antrian.tail] = data;  
        cout << " " << antrian.data[antrian.tail] << " masuk!" << endl;  
    }  
    else if (isFull() == 0){  
        antrian.tail++;  
        antrian.data[antrian.tail] = data;  
  
        cout << " " << antrian.data[antrian.tail] << " masuk!" << endl;  
    }  
    else{  
        cout << " antrian sudah penuh!" << endl;  
    }  
}
```

## 6. Dequeue

Operasi untuk mengambil atau menghapus data terdepan (head) dari queue.

```
int dequeue()
{
    int dq = antrian.data[antrian.head];
    if (isEmpty() == 0){
        for (int i = antrian.head; i <= antrian.tail; i++){
            antrian.data[i] = antrian.data[i+1];
        }
        antrian.tail--;
        cout << " antrian depan terhapus." << endl;
        cout << " data terhapus = " << dq << endl;
    }
    else{
        cout << " antrian masih kosong!" << endl;
    }
}
```

## 7. Clear

Operasi untuk menghapus atau mengosongkan seluruh data queue dengan cara membuat Tail dan Head = -1 seperti sedia kala.

```
void clear()
{
    antrian.head = antrian.tail = -1;
}
```

## 8. Display

Operasi untuk menampilkan seluruh data queue.

```
void display()
{
    if (isEmpty() == 0){
        for (int i = antrian.head; i <= antrian.tail; i++){
            cout << " " << antrian.data[i] << endl;
        }
    }
    else{
        cout << " antrian masih kosong!" << endl;
    }
}
```

## B. Guided

Sourcode :

```
#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
        {
            // antriannya 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 :

The screenshot shows a terminal window with the output of a C++ program. The program uses a queue implemented with an array. The output shows the queue being populated with 'Andi', 'Maya', and 'Budi', then 'Ucup' and 'Umar' are added. The queue is then cleared. A Notepad++ window is overlaid on the terminal, showing a text file with the following content:

```

Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B

```

The terminal output is as follows:

```

g++ C:\msys64\usr\bin\gdb.exe -i 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 D:\contoh>

```

Deskripsi:

Program di atas merupakan implementasi queue dalam bentuk array.

### C. Unguided

Sourcode :

```
#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);

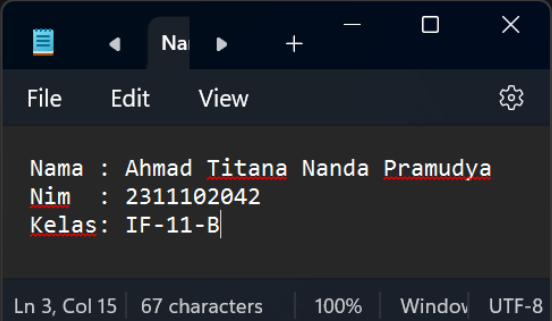
return 0;
}

```

## Screenshot Output:

### 1. Memu

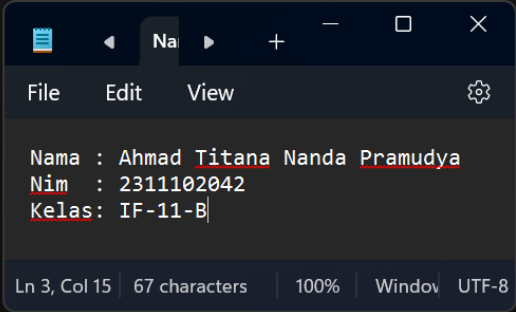
```
gExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
0. Keluar
Pilih menu:
```



```
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

### 2. Tambahan Antrian

```
Masukkan Nama: Ahmad
Masukkan NIM: 2311102042
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: Asep
Masukkan NIM: 2311102045
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: Ihsan
Masukkan NIM: 2311102077
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: Arvan
Masukkan NIM: 2311102074
Menu Antrian Mahasiswa:
```



```
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

### 3. Hapus Antrian

```
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 View
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

### 4. Lihat Antrian

```
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. Asep (2311102045)
2. Ihsan (2311102077)
3. Arvan (2311102074)
```

```
File Edit View
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

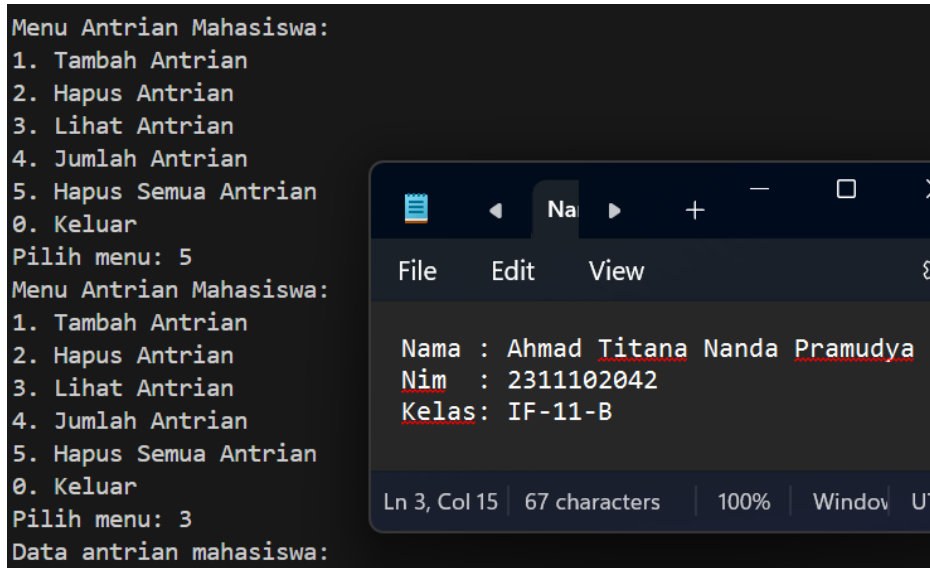
### 5. Jumlah Antrian

```
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 = 3
Menu Antrian Mahasiswa:
```

```
File Edit View
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

Ln 3, Col 15 | 67 characters | 100% | Window UTF-

## 6. Hapus Semua Antrian



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

Na

File Edit View

Nama : Ahmad Titana Nanda Pramudya  
Nim : 2311102042  
Kelas: IF-11-B

Ln 3, Col 15 | 67 characters | 100% | Window UT

Deskripsi:

Program ini adalah implementasi 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). Dengan demikian, queue bersifat FIFO (First InFirst Out), yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yang terakhir masuk akan keluar terakhir. Maka perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan.

## E. Referensi

[1] Academia.edu

[https://www.academia.edu/43671713/Struktur\\_Data\\_Queue\\_C](https://www.academia.edu/43671713/Struktur_Data_Queue_C)

[2] Dicoding

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