

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

MODUL 7



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

Disusun oleh:

ABDA FIRAS RAHMAN

2311102049

IF-11-B

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

BAB I

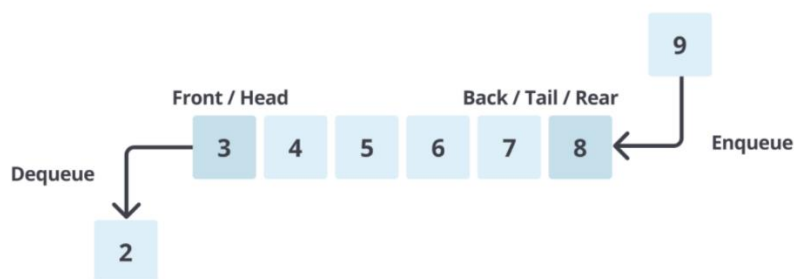
DASAR TEORI

QUEUE

1. Pengertian

Queue atau dalam bahasa Indonesia yang berarti antrean adalah struktur data yang menyusun elemen-elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrean akan menjadi yang pertama pula untuk dikeluarkan.

Caranya bekerja adalah seperti jejeran orang yang sedang menunggu antrean di supermarket di mana orang pertama yang datang adalah yang pertama dilayani (First In, First Out). Pada struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut Front atau Head. Sebaliknya, data pada urutan terakhir (data yang baru saja ditambahkan) disebut Back, Rear, atau Tail. Proses untuk menambahkan data pada antrean disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrean disebut dengan Dequeue.



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.

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.

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.

Guided

```
#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 { // Antrianya 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;
    }
}

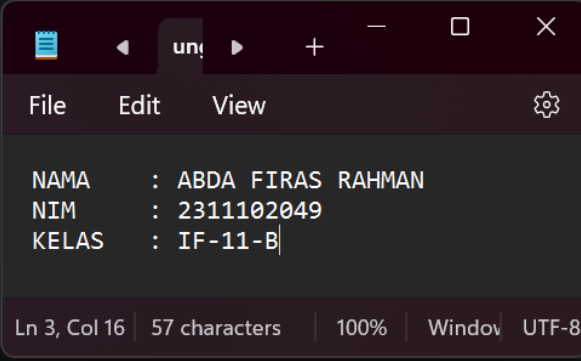
void viewQueue() { // Fungsi melihat antrian
    cout << "Data antrian teller:" << endl;
    for (int i = 0; i < maksimalQueue; i++) {
        if (queueTeller[i] != "") {
            cout << i + 1 << ". " << queueTeller[i] <<
endl;
        } else {
            cout << i + 1 << ". (kosong)" << endl;
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

```

```
    }  
}  
  
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;  
}
```

OUTPUT

```
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
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
```



DESKRIPSI PROGRAM

Program ini mengimplementasikan struktur data Queue (antrian) dengan kapasitas maksimum 5 untuk mengelola antrian nasabah pada teller bank. Terdapat fungsi-fungsi untuk menambah data ke antrian (`enqueueAntrian`), menghapus data dari antrian (`dequeueAntrian`), memeriksa apakah antrian penuh (`isFull`) atau kosong (`isEmpty`), menghitung jumlah data dalam antrian (`countQueue`), menghapus seluruh data dalam antrian (`clearQueue`), dan menampilkan isi antrian (`viewQueue`).

Dalam fungsi main, program mendemonstrasikan penggunaan fungsi-fungsi tersebut dengan menambahkan dua data ke antrian, menampilkan isi antrian, menghapus satu data, menampilkan isi antrian kembali, mengosongkan seluruh antrian, dan terakhir menampilkan isi antrian yang sudah kosong. Implementasi ini menggunakan array statis untuk menyimpan data antrian.

UNGUIDED

UNGUIDED 1

```
#include <iostream>
using namespace std;
struct Node
{
    string data;
    Node *next;
};
class Queue
{
private:
    Node *front;
    Node *rear;
public:
    Queue()
    {
        front = nullptr;
        rear = nullptr;
    }
    bool isEmpty()
    {
        return (front == nullptr);
    }
    void enqueue(string data)
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->next = nullptr;
        if (isEmpty())
        {
            front = newNode;
            rear = newNode;
        }
        else
        {
            rear->next = newNode;
            rear = newNode;
        }
    }
    void dequeue()
    {
        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 *current = front;
        while (current != nullptr)
        {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
}

};

int main()
{
    Queue queue;
    cout << "\nMenambahkan 3 orang ke antrian" << endl;
    queue.enqueue("piras");
    queue.enqueue("jaki");
    queue.enqueue("alpin");
    queue.viewQueue();
    cout << "\nMengurangi antrian, orang yang pertama masuk akan pertama
keluar " << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "\nMenambahkan 2 orang ke antrian" << endl;
    queue.enqueue("daoa");
    queue.enqueue("parel");
    queue.viewQueue();
    cout << "\nMenghapus semua antrian" << endl;
    while (!queue.isEmpty())
    {
        queue.dequeue();
    }
    queue.viewQueue();
    return 0;
}

```

```
}
```

OUTPUT

```
Menambahkan 3 orang ke antrian
piras jaki alpin

Mengurangi antrian, orang yang pertama masuk akan pertama keluar
jaki alpin

Menambahkan 2 orang ke antrian
jaki alpin daoa parel

Menghapus semua antrian
Antrian kosong
```

File	Edit	View
NAMA	:	ABDA FIRAS RAHMAN
NIM	:	2311102049
KELAS	:	IF-11-B

Ln 3, Col 16 | 57 characters | 100%

DESKRIPSI PROGRAM

Program ini mengimplementasikan struktur data antrian (queue) menggunakan linked list. Antrian direpresentasikan sebagai kelas Queue yang memiliki node depan (front) dan node belakang (rear). Setiap node memiliki data berupa string dan pointer ke node berikutnya. Terdapat operasi enqueue untuk menambahkan data baru ke antrian, dequeue untuk menghapus data dari depan antrian, isEmpty untuk memeriksa apakah antrian kosong, dan viewQueue untuk menampilkan semua data dalam antrian. Program utama mendemonstrasikan penggunaan kelas Queue dengan menambahkan, menghapus, dan menampilkan data dalam antrian. Operasi enqueue menambahkan node baru di akhir antrian, sedangkan dequeue menghapus node dari depan antrian. Aturan antrian mengikuti prinsip "First In First Out" (FIFO), di mana data yang pertama masuk akan keluar terlebih dahulu.

UNGUIDED 2

```
#include <iostream>
using namespace std;
struct Node
{
    string data;
    Node *next;
};
class Queue
{
private:
    Node *front;
    Node *rear;
public:
```

```

Queue()
{
    front = nullptr;
    rear = nullptr;
}
bool isEmpty()
{
    return (front == nullptr);
}
void enqueue(string data)
{
    Node *newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    if (isEmpty())
    {
        front = newNode;
        rear = newNode;
    }
    else
    {
        rear->next = newNode;
        rear = newNode;
    }
}
void dequeue()
{
    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 *current = front;
        while (current != nullptr)

```

```

        {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
}

};

int main()
{
    Queue queue;
    cout << "\nMenambahkan 3 orang ke antrian" << endl;
    queue.enqueue("piras");
    queue.enqueue("jaki");
    queue.enqueue("alpin");
    queue.viewQueue();
    cout << "\nMengurangi antrian, orang yang pertama masuk akan pertama
keluar " << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "\nMenambahkan 2 orang ke antrian" << endl;
    queue.enqueue("daoa");
    queue.enqueue("parel");
    queue.viewQueue();
    cout << "\nMenghapus semua antrian" << endl;
    while (!queue.isEmpty())
    {
        queue.dequeue();
    }
    queue.viewQueue();
    return 0;
}

```

OUTPUT

The screenshot shows the output of a C++ program and a Notepad++ window. The program output is as follows:

```

=====
Manajemen Antrian
=====
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
=====
Masukkan Pilihan: 4
Tampilkan Data :

abda NIM : 49

> Kembali ke menu utama?[Y/n]

```

The Notepad++ window shows the following text:

```

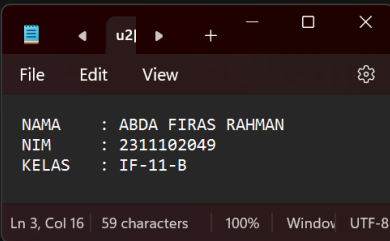
NAMA      : ABDA FIRAS RAHMAN
NIM       : 2311102049
KELAS     : IF-11-B

```

The Notepad++ status bar indicates the cursor is at Line 3, Column 16, with 59 characters, 100% zoom, and UTF-8 encoding.

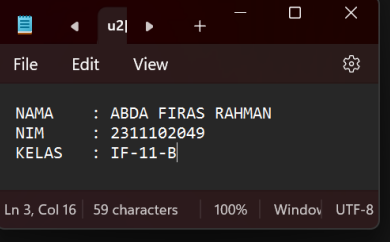
```
=====
Manajemen Antrian
=====
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
=====
Masukkan Pilihan: 2
[!] 1 Data berhasil dihapus

> Kembali ke menu utama?[Y/n]
```



```
=====
Manajemen Antrian
=====
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
=====
Masukkan Pilihan: 3
[!] Data berhasil di reset

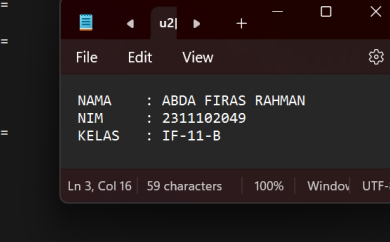
> Kembali ke menu utama?[Y/n]
```



```
=====
Manajemen Antrian
=====
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
=====
Masukkan Pilihan: 4
Tampilkan Data :

Antrian kosong

> Kembali ke menu utama?[Y/n]
```



DESKRIPSI PROGRAM

Program ini mengimplementasikan struktur data antrian (queue) menggunakan linked list untuk mengelola data mahasiswa yang terdiri dari nama dan NIM. Antrian direpresentasikan sebagai kelas Queue yang memiliki operasi untuk menambah (enqueue), menghapus (dequeue), memeriksa kekosongan (isEmpty), dan menampilkan (viewQueue) data mahasiswa dalam antrian. Program utama menyediakan menu interaktif dengan pilihan untuk menambah data mahasiswa baru, menghapus data mahasiswa dari depan antrian, mereset atau mengosongkan seluruh antrian, dan menampilkan semua data mahasiswa dalam antrian. Ketika memilih opsi 1, pengguna diminta untuk memasukkan nama dan NIM mahasiswa yang akan ditambahkan ke antrian. Opsi 2 akan menghapus data mahasiswa dari depan antrian sesuai dengan aturan "First In First Out" (FIFO). Opsi 3 akan mengosongkan seluruh antrian dengan menghapus semua data mahasiswa yang ada di dalamnya. Opsi 4 akan menampilkan semua data mahasiswa yang tersimpan dalam antrian saat ini. Program akan terus berjalan dalam loop sampai pengguna memilih untuk keluar dari menu utama.

DAFTAR PUSTAKA

- 1) Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya

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

