

**LAPORAN PRAKTIKUM
STRUKTUR DATA PEMOGRAMAN**

**MODUL VII
“QUEUE”**



Disusun Oleh:

NAMA : Trie Nabilla Farhah

NIM : 2311102071

Dosen Pengampu:

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

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

2024

A. DASAR TEORI

Kebalikan dari stack, queue (antrian) adalah suatu jenis struktur data yang dapat diproses dengan sifat FIFO (First In First Out), dimana elemen yang pertama kali masuk ke antrian akan keluar pertama kalinya. Ada dua jenis operasi yang bias dilakukan di antrian enqueue (memasukkan elemen baru ke dalam elemen) dan dequeue (adalah mengeluarkan satu elemen dari suatu antrian). Antrian dapat dibuat dengan menggunakan: Linier Array dan Circular Array.

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.



Pada gambar di atas, karena elemen 1 ditambahkan ke antrian lebih dulu daripada 2, maka 1 adalah elemen yang pertama dihapus dari antrian. Hal ini mengikuti aturan operasi FIFO.

Berbeda dengan struktur data stack yang menyimpan data secara bertumpuk dimana hanya terdapat satu ujung yang terbuka untuk melakukan operasi data, struktur data queue justru disusun secara horizontal dan terbuka di kedua ujungnya. Ujung pertama (head) digunakan untuk menghapus data sedangkan ujung lainnya (tail) digunakan untuk menyisipkan data.

Persamaan antara stack dan queue adalah keduanya dapat diimplementasikan menggunakan struktur data linked list atau array.

Queue adalah struktur data abstrak (ADT) yang memungkinkan operasi berikut:

- Enqueue: Menambahkan elemen ke akhir antrian
- Dequeue: Menghapus elemen dari depan antrian
- IsEmpty: Memeriksa apakah antrian kosong
- IsFull: Memeriksa apakah antrian sudah penuh
- Peek: Mendapatkan nilai bagian depan antrian tanpa menghapusnya
- Initialize: Membuat antrian baru tanpa elemen data (kosong)

B. GUIDED

GUIDED 1

Source Code

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

const int maksimalQueue = 5; //Batas maksimal antrian
int front = 0;                //Indeks antrian awal
int back = 0;                 //Indeks antrian akhir
string queueTeller[maksimalQueue]; //Array untuk menyimpan
elemen antrian

// Fungsi untuk memeriksa apakah antrian penuh
bool isFull(){
    return back == maksimalQueue;
}

//Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty(){
    return back == 0;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string data){
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    }else {
        queueTeller[back]= data;
        back++;
    }
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian(){
    if (isEmpty()){
        cout << "Antrian kosong" << endl;
    }else {
        for (int i=0; i< back - 1; i++){
            queueTeller[i] = queueTeller[i+1];
        }
    }
}
```

```

        }
        queueTeller [back-1]= ""; // Membersihkan data
        terakhir
        back--;
    }
}

// Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue(){
    return back;
}

// Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue(){
    for (int i = 0; i < back; i++){
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

// Fungsi untuk menampilkan semua elemen dalam 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");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    dequeueAntrian();
    viewQueue();
}

```

```

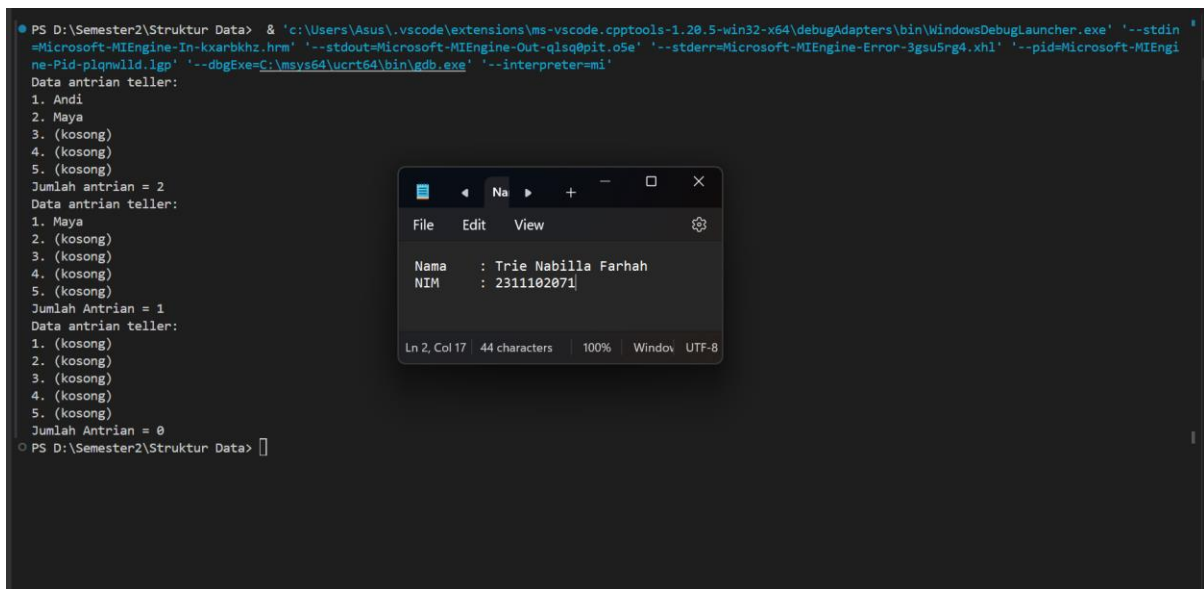
        cout << "Jumlah Antrian = " << countQueue() << endl;

        clearQueue();
        viewQueue();
        cout << "Jumlah Antrian = " << countQueue() << endl;

        return 0;
    }

```

Output



```

PS D:\Semester2\Struktur Data> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin '
=Microsoft-MIEngine-In-kxarbkxz.hrm' '--stdout=Microsoft-MIEngine-Out-qlsq0pit.o5e' '--stderr=Microsoft-MIEngine-Error-3gsu5rg4.xhl' '--pid=Microsoft-MIEngi
ne-Pid-plqmwild.lgp' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
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
PS D:\Semester2\Struktur Data>

```

Deskripsi Program

Program ini menggunakan array untuk menjalankan struktur data antrian (queue) dalam bahasa C++. Berikut adalah penjelasan tentang cara kerja program :

- **maksimalQueue:** Maksimum jumlah elemen yang dapat disimpan dalam antrian, yaitu 5.
- **Front** adalah indeks elemen antrian pertama, yang tidak digunakan dalam implementasi ini.
- **Back** adalah indeks elemen antrian terakhir, yang diinisialisasi ke 0.
- **QueueTeller** adalah array untuk menyimpan elemen antrian dengan ukuran **maksimalQueue**.
- **Fungsi isFull():** Fungsi ini memeriksa apakah antrian penuh dengan membandingkan nilai **back** dengan **maksimalQueue**. Jika nilai **back** sama dengan **maksimalQueue**, maka antrian dianggap penuh.
- **Fungsi enqueueAntrian(string data):** Jika antrian tidak penuh, ia menambahkan elemen baru ke dalamnya. Jika **back** sama dengan **maksimalQueue**, maka antrian dianggap penuh.
- **Fungsi isEmpty():** Fungsi ini memeriksa apakah antrian kosong dengan membandingkan nilai **back** dengan **maksimalQueue**. Jika nilai **back** sama dengan 0, maka antrian dianggap kosong.

- Fungsi `dequeueAntrian()`: Fungsi ini menghapus elemen pertama dari antrian jika antrian tidak kosong. Jika antrian kosong, ditampilkan pesan "Antrian kosong"; jika tidak, elemen pertama dihapus dengan menggeser semua elemen ke kiri dan mengosongkan elemen terakhir.
- Fungsi `countQueue`: Fungsi ini mengembalikan jumlah elemen dalam antrian dengan mengembalikan nilai `back`.
- Fungsi `clearQueue`: Fungsi ini mengosongkan semua elemen dalam antrian dengan mengatur ulang semua elemen menjadi string kosong dan mengatur kembali dan depan ke 0.
- Fungsi `viewQueue()`: Fungsi ini menampilkan semua elemen dalam antrian; elemen dengan indeks akan ditampilkan, dan elemen kosong akan ditampilkan sebagai "(kosong)".
- Fungsi `main()`: Fungsi ini menjalankan beberapa operasi untuk menunjukkan fungsionalitas antrian.

C. UNGUIDED

Unguided 1

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

Source Code

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

const int maximalQueue = 5;
int length = 0;

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

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nilai)
{
    if (isFull())
```

```

    {
        cout << "Antrian Penuh" << endl;
    }
else
{
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
    length++;
}
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

```



```

}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "Semua antrian dihapus" << endl;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->data << " " <<
endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)

```

```

        {
            jumlah++;
            hitung = hitung->next;
        }
        return jumlah;
    }

int main()
{
    init();
    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

The screenshot shows a Windows command prompt window with the following output:

```

PS D:\Semester2\Struktur Data> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin
=Microsoft-MIEngine-In-tdcymwsv.htb' '--stdout=Microsoft-MIEngine-Out-iujkp3le.nrn' '--stderr=Microsoft-MIEngine-Error-liu0elib.zer' '--pid=Microsoft-MIEngi
ne-Pid-qkz1asv4.pxm' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
1. Andi
2. Maya

Jumlah antrian = 2
1. Maya

Jumlah antrian = 1
Semua antrian dihapus
Tidak ada antrian
Jumlah antrian = 0
PS D:\Semester2\Struktur Data>

```

Overlaid on the command prompt is a Notepad window titled "Na" containing the following text:

```

Nama : Trie Nabilla Farhah
NIM : 2311102071

```

The Notepad window also shows a status bar at the bottom: "Ln 2, Col 17 | 44 characters | 100% | Window | UTF-8".

Deskripsi Program

Program tersebut merupakan implementasi dari struktur data antrian (queue) menggunakan linked list. Berikut penjelasan cara kerjanya:

- struct Node menunjukkan setiap elemen dalam antrian. Setiap node memiliki string data dan pointer yang menunjuk ke node berikutnya.
- Head adalah pointer yang menunjuk ke simpul pertama dalam antrian, dan tail adalah pointer yang menunjuk ke simpul terakhir.
- MaximalQueue adalah konstanta yang menentukan batas panjang queue maksimum.
- Length adalah variabel yang menyimpan jumlah elemen dalam queue.
- Fungsi init() : Dengan menginisialisasi head dan tail menjadi NULL, Anda akan melihat antrian kosong.
- Fungsi isFull() : Jika panjang antrian sudah mencapai batas tertinggi, kembalikan nilai asli.
- Fungsi isEmpty() : mengembalikan nilai benar jika antrian kosong (head NULL)
- Fungsi enqueueAntrian(nilai string) : Jika antrian sudah penuh, cetak pesan "Antrian Penuh". Jika antrian masih tersisa, tambahkan node baru ke ujungnya dan update ujungnya.
- Fungsi dequeueAntrian() : Hapus elemen pertama dari antrian. Jika antrian tidak kosong, hapus node pertama dan update head. Jika antrian tidak kosong, cetak pesan "Tidak ada antrian".
- Fungsi clearQueue() : Menghapus semua elemen dari rantai dengan melewati setiap node dan menghapus elemen satu per satu.
- Fungsi viewQueue() : Setelah semua node dihapus, atur head dan tail menjadi NULL. Akan Menampilkan seluruh item dalam satu baris. Tampilkan data setelah melewati setiap node dalam antrian.
- Fungsi countQueue() : Mengembalikan jumlah elemen yang ada dalam baris. Menghitung jumlah setelah melewati setiap node dalam antrian.

Setelah menambahkan "Andi" dan "Maya" ke dalam antrian, program akan menampilkan jumlah dan isi antrian. Kemudian, program akan menghapus "Andi" dari antrian dan menampilkan kembali jumlah dan isi antrian. Setelah itu, program akan menghapus semua elemen dari antrian dan menampilkan bahwa antrian kosong.

Unguided 2

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

Source Code

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

const int maximalQueue = 5;
int length = 0;

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

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nama, string nim)
{
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
```

```

        baru->nama = nama;
        baru->nim = nim;
        baru->next = NULL;
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
        cout << endl << "Berhasil Masuk Antrian";
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
}

```

```

    }
    head = tail = NULL;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->nama << " - " <<
bantu->nim << endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Antrian masih kosong" << endl;
    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

int main()
{
    init();
    system("cls");
    do
    {
        int choice;
        string nama, nim;
        cout << "-----ANTRIAN MAHASISWA-----"
        << endl;

```

```

        cout << "- 1. Tambah Antrian                -
" << endl;
        cout << "- 2. Keluar Antrian                -
" << endl;
        cout << "- 3. Jumlah Antrian                -
" << endl;
        cout << "- 4. Lihat Antrian                -
" << endl;
        cout << "- 5. Hapus Antrian                -
" << endl;
        cout << "- 0. Keluar                        -
" << endl;
        cout << "- Pilih > ";
        cin >> choice;
        cout << endl;
        switch (choice)
        {
        case 1:
            cout << "Masukkan Nama > ";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan NIM > ";
            cin >> nim;
            enqueueAntrian(nama, nim);
            system("pause > nul");
            system("cls");
            break;

        case 2:
            dequeueAntrian();
            cout << "Berhasil keluar" << endl;
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 3:
            cout << "Jumlah Antrian : " << countQueue() <<
endl;
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 4:

```

```
        viewQueue();
        cout << endl;

        system("pause > nul");
        system("cls");
        break;

    case 5:
        clearQueue();
        cout << "Data berhasil dihapus" << endl;
        cout << endl;

        system("pause > nul");
        system("cls");
        break;

    case 0:
        cout << "Terima kasih telah menggunakan program  
ini" << endl;
        exit(0);

    default:
        break;
}

} while (true);

return 0;

}
```


Output

```
PS D:\Semester2\Struktur Data> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ijzds5t3.oli' '--stdout=Microsoft-MIEngine-Out-4p051pkx.k4d' '--stderr=Microsoft-MIEngine-Error-uSzutyfe.vqg' '--pid=Microsoft-MIEngine-Pid-ijxyqtuh.eds' '--dbgExe=C:\msys64\bin\gdb.exe' '--interpreter=mi'

-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 1

Masukkan Nama : trie nabilla
Masukkan NIM : 2311102071

Berhasil Masuk Antrian
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 1

Masukkan Nama : billa
Masukkan NIM : 2311102098

Berhasil Masuk Antrian
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS

-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 3

Jumlah Antrian : 2
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 4

1. trie nabilla - 2311102071
2. billa - 2311102098
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
```

```
5. Hapus Antrian
0. Keluar
Pilih : 2

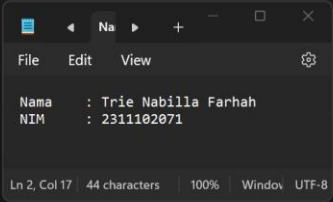
Berhasil keluar

-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 5

Data berhasil dihapus

-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 0

Terima kasih telah menggunakan program ini
PS D:\Semester2\Struktur Data>
```



Deskripsi Program

Program ini menjalankan antrian siswa dengan daftar terkait. Dengan program ini, pengguna dapat melakukan banyak hal, seperti menambahkan siswa ke antrian, mengeluarkan siswa dari antrian, melihat jumlah siswa dalam antrian, melihat daftar siswa dalam antrian, dan menghapus semua data siswa.

- Struktur Node : Data siswa, termasuk nama dan nim, serta pointer yang menunjuk ke node berikutnya, disimpan dalam struct node.
- Head adalah pointer yang menunjuk ke simpul pertama dalam antrian, dan tail adalah pointer yang menunjuk ke simpul terakhir.
- MaximalQueue adalah konstanta yang menentukan batas panjang queue maksimum.
- Length queue adalah variabel yang menyimpan jumlah elemen dalam queue.
- Fungsi init() : Dengan menginisialisasi head dan tail menjadi NULL, Anda akan melihat antrian kosong.
- Fungsi isFull() : Jika panjang antrian sudah mencapai batas tertinggi, kembalikan nilai asli.
- Fungsi isEmpty() : mengembalikan nilai benar jika antrian kosong (head NULL).
- Fungsi enqueueAntrian adalah sebagai berikut: string nama, string nim Masukkan siswa ke dalam antrian. Jika antrian sudah penuh, cetak pesan "Antrian Penuh". Jika antrian masih tersisa, tambahkan node baru ke ujungnya dan update ujungnya.
- Fungsi dequeueAntiran() : Hapus siswa pertama dari antrian. Jika tidak ada antrian, hapus node pertama dan update head. Jika tidak ada antrian, cetak pesan "Tidak ada antrian".
- Fungsi clearQueue() : Menghapus semua data siswa dari daftar. Hapus setiap node dalam antrian.
- Fungsi viewQueue() : Setelah semua node dihapus, atur head dan tail menjadi NULL. Menampilkan antrian seluruh siswa. Menampilkan data siswa (nama dan nim) melalui setiap node dalam antrian.
- Fungsi countQueue() : Menghitung jumlah siswa dalam antrian dengan mengintip ke setiap node dan menghitung jumlah.
- Konfigurasi dan Menu : menginisialisasi antrian dengan memanggil fungsi init(). menunjukkan kepada pengguna menu pilihan operasi.

- Pengulangan: Program mengeksekusi fungsi yang sesuai sesuai dengan pilihan pengguna. Setelah operasi selesai, program menampilkan pesan atau hasil yang sesuai, dan kembali ke menu.
- Program berjalan dalam loop tak terbatas sampai pengguna memilih untuk keluar (pilihan 0).

D. KESIMPULAN

Queue (antrian) adalah salah satu list linier dari struktur data yang beroperasi dengan cara First In First Out (FIFO) yaitu elemen pertama yang masuk merupakan elemen yang pertama keluar. Data-data di dalam antrian dapat bertipe integer, real, record dalam bentuk sederhana atau terstruktur.

Queue dilakukan dengan cara penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut rear/tail, sedang ujung penghapusa disebut front/head. Sebuah queue dalam program setidaknya harus mengandung tiga variabel, yakni: head untuk penanda bagian depan antrian, tail untuk penanda bagian belakang antrian, dan array data untuk menyimpan data-data yang dimasukkan ke dalam queue tersebut.

Pada queue ada operasi – operasi dasar, yaitu: prosedur create untuk membuat queue baru yang kosong, fungsi IsEmpty untuk mengecek queue tersebut kosong atau tidak, fungsi IsFull untuk mengecek apakah queue penuh, prosedur EnQueue untuk memasukkan data ke dalam queue, prosedur DeQueue untuk mengeluarkan elemen pada posisi head dari queue, fungsi clear untuk menghapus elemen dari queue, dan prosedur tampil untuk menampilkan elemen yang ada di queue.

E. REFERENSI

Asisten praktikum, “*Modul 7 Queue*”, learning Management System, 2024

Johnson, S.(2020).”*Penerapan Stack dan Queue Pada Array dan Linked List Dalam Java*”,jurnal struktur data. 1(2), 6.

Trivusi.(2021, 1 Juli). *Struktur Data Queue : Pengertian, Jenis, dan Kegunaanya* . Diakses pada 22 Mei 2024, dari <https://www.trivusi.web.id/2022/07/struktur-data-queue.html>

Rizki, M. (2023, 29 November). *Struktur Data Queue : Pengertian, Fungsi, dan Jenisnya*. Diakses pada 22 Mei 2024, dari <https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/>