

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL VII  
QUEUE**



**Disusun Oleh :**

NAMA : GALIH TRISNA  
NIM : 2311102050

**Dosen**

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

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

## A. Dasar Teori

Queue adalah struktur data yang mengikuti prinsip First In First Out (FIFO), artinya item pertama dalam antrian akan menjadi item pertama yang keluar dari antrian. Queue dapat diibaratkan sebagai antrian dimana item-item baru ditambahkan pada salah satu ujung antrian (belakang) dan item-item yang sudah ada dikeluarkan pada ujung yang lain (depan). Queue mirip dengan antrian di kehidupan nyata yang sering kita jumpai dalam kehidupan sehari-hari.



Karakteristik utama dari Queue adalah prinsip FIFO (First-In-First-Out). Elemen pertama yang dimasukkan ke dalam Queue akan menjadi elemen pertama yang diambil atau dihapus dari Queue. Elemen-elemen baru ditambahkan di ujung belakang Queue dan elemen-elemen yang sudah ada dikeluarkan dari ujung depan Queue. Dengan prinsip FIFO ini, Queue dapat membantu mengatur urutan data dan mempertahankan prioritas saat memproses elemen-elemen yang ada di dalamnya.

Contoh penggunaan Queue dalam kehidupan sehari-hari dapat ditemukan di berbagai situasi. Misalnya, saat kita mengantri di sebuah toko, bank, atau tempat layanan pelanggan. Orang yang pertama kali mengantri akan dilayani terlebih dahulu dan orang yang datang kemudian akan menunggu giliran mereka sesuai dengan urutan kedatangan. Selain itu, Queue juga digunakan dalam sistem pemrosesan data, seperti antrian pesan di aplikasi komunikasi atau penjadwalan tugas pada sistem operasi.

Dengan konsep dan karakteristiknya yang sederhana, Queue menjadi struktur data yang penting dalam pemrograman dan digunakan dalam berbagai aplikasi untuk mengelola urutan data dan mempertahankan prinsip FIFO.

## B. Guided

### Guided 1

```
#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;
    }
}
}

```

```

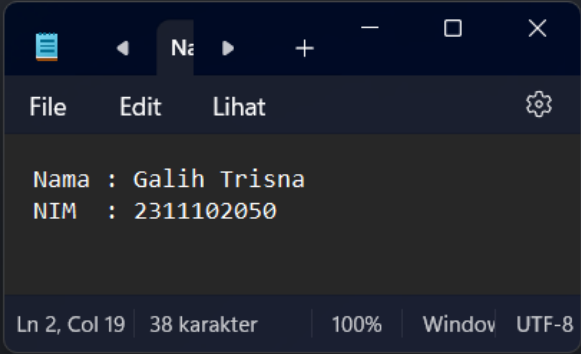
void viewQueue()
{ // Fungsi melihat antrian
    cout << "Data antrian teller:" << endl;
    for (int i = 0; i < maksimalQueue; i++)
    {
        if (queueTeller[i] != "")
        {
            cout << i + 1 << ". " << queueTeller[i] << endl;
        }
        else
        {
            cout << i +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

```

## Screenshots Output

```
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\220524> cd ..
g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
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 C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\220524>
```



## Deskripsi:

Program di atas adalah implementasi dari sebuah queue menggunakan array. Queue di dalam program ini memiliki kapasitas maksimum yang ditentukan oleh variabel `maksimalQueue`, yaitu 5. Dua variabel, `front` dan `back`, digunakan sebagai penanda untuk menunjukkan posisi depan dan belakang antrian. `front` menandakan posisi elemen pertama di antrian, sedangkan `back` menandakan posisi berikutnya yang kosong dalam antrian. Fungsi `isFull` digunakan untuk memeriksa apakah antrian sudah penuh. Fungsi ini mengembalikan nilai `true` jika nilai `back` sama dengan `maksimalQueue`, menunjukkan bahwa tidak ada tempat lagi dalam antrian untuk elemen baru. Sebaliknya, fungsi `isEmpty` digunakan untuk memeriksa apakah antrian kosong, dengan mengembalikan nilai `true` jika nilai `back` sama dengan 0. Fungsi `enqueueAntrian` digunakan untuk menambahkan elemen baru ke antrian. Jika antrian penuh, fungsi ini akan mencetak pesan "Antrian penuh". Jika antrian kosong, elemen baru akan ditempatkan di posisi pertama dan nilai `front` dan `back` akan ditingkatkan. Jika antrian tidak kosong, elemen baru akan ditempatkan di posisi belakang, dan hanya nilai `back` yang akan ditingkatkan. Fungsi `dequeueAntrian` digunakan untuk mengeluarkan elemen dari antrian. Jika antrian kosong, fungsi ini akan mencetak pesan "Antrian kosong". Jika tidak, semua elemen akan digeser satu posisi ke depan, dan nilai `back` akan

dikurangi. Fungsi `countQueue` mengembalikan jumlah elemen dalam antrian dengan mengembalikan nilai `back`. Fungsi `clearQueue` digunakan untuk menghapus semua elemen dari antrian dengan mengosongkan array dan mengatur kembali nilai `front` dan `back` ke 0. Fungsi `viewQueue` digunakan untuk menampilkan isi antrian. Fungsi ini mencetak semua elemen dalam antrian beserta posisi mereka. Jika suatu posisi kosong, akan ditampilkan "(kosong)". Di dalam fungsi `main`, beberapa operasi pada antrian dilakukan untuk uji coba pada fungsi-fungsi tersebut. Pertama, elemen "Andi" dan "Maya" ditambahkan ke dalam antrian, kemudian antrian ditampilkan dan jumlah elemen dalam antrian dihitung dan dicetak. Setelah itu, satu elemen dikeluarkan dari antrian, antrian ditampilkan kembali, dan jumlah elemen dicetak lagi. Terakhir, semua elemen dalam antrian dihapus, antrian ditampilkan, dan jumlah elemen dicetak.

### C. Unguided/Tugas

#### Unguided 1

```
#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 << "List masih kosong!" << 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;
}

```

### Screenshots Output

```

PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\220524> g++ unguided1.cpp -o unguided1 ; if ($?) { .\unguided1 }
1. Andi
2. Maya

Jumlah antrian = 2
1. Maya

Jumlah antrian = 1
Semua antrian dihapus
Tidak ada antrian
Jumlah antrian = 0
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\220524>

```

Inset window content:

```

Nama : Galih Trisna
NIM  : 2311102050

```

### Deskripsi

Program di atas adalah implementasi dari queue menggunakan linked list. Dalam program ini, terdapat beberapa bagian, yaitu:

1. Struct Node: Struktur data Node digunakan untuk merepresentasikan setiap elemen dalam queue. Setiap node memiliki dua atribut yaitu data yang menyimpan nilai string dan next yang merupakan pointer ke node berikutnya dalam queue.
2. Variabel head dan tail digunakan untuk menunjuk ke node pertama (head) dan node terakhir (tail) dalam queue. Selain itu, terdapat variabel length yang melacak jumlah elemen dalam queue, dan variabel maximalQueue yang menentukan kapasitas maksimum queue.
3. Fungsi init yaitu untuk menginisialisasi queue dengan mengatur head dan tail menjadi NULL, menandakan bahwa queue kosong.

4. Fungsi `isFull` dan `isEmpty`: Fungsi `isFull` mengembalikan `true` jika queue penuh, yaitu ketika `length` sama dengan `maximalQueue`. Fungsi `isEmpty` mengembalikan `true` jika queue kosong, yaitu ketika `head` adalah `NULL`.

5. Fungsi `enqueueAntrian`: Fungsi ini menambahkan elemen baru ke queue. Jika queue penuh, akan mencetak pesan "Antrian Penuh". Jika queue kosong, elemen baru menjadi `head` dan `tail`. Jika queue tidak kosong, elemen baru ditambahkan di akhir queue, dan `tail` diperbarui untuk menunjuk ke elemen baru. Panjang queue (`length`) juga diperbarui.

6. Fungsi `dequeueAntrian`: Fungsi ini menghapus elemen dari queue. Jika queue tidak kosong, elemen pertama (yang ditunjuk oleh `head`) dihapus. Jika hanya ada satu elemen, `head` dan `tail` diatur kembali menjadi `NULL`. Jika queue kosong, mencetak pesan "Tidak ada Antrian".

7. Fungsi `clearQueue`: Fungsi ini menghapus semua elemen dalam queue dengan menghapus setiap node satu per satu hingga queue kosong. Setelah semua elemen dihapus, `head` dan `tail` diatur kembali menjadi `NULL`, dan mencetak pesan "Semua antrian dihapus".

8. Fungsi `viewQueue`: Fungsi ini menampilkan semua elemen dalam queue. Jika queue tidak kosong, elemen-elemen ditampilkan satu per satu. Jika queue kosong, mencetak pesan "Tidak ada antrian".

9. Fungsi `countQueue`: Fungsi ini menghitung dan mengembalikan jumlah elemen dalam queue.

10. Fungsi `main`: Fungsi utama ini menunjukkan penggunaan queue dengan beberapa operasi: menambahkan elemen (`enqueuequeue`), menghapus elemen (`dequeuequeue`), menampilkan elemen (`viewQueue`), menghitung jumlah elemen (`countQueue`), dan menghapus semua elemen (`clearQueue`). Contoh penggunaan dalam `main` yaitu dengan elemen "Andi" dan "Maya" ditambahkan ke queue, kemudian queue ditampilkan dan jumlah elemen dihitung. Setelah itu, elemen pertama dihapus, queue ditampilkan kembali dan jumlah elemen dihitung lagi. Terakhir, semua elemen dihapus dari queue, queue ditampilkan, dan jumlah elemen dihitung lagi.

## Unguided 2

```
#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;
}

```

## Screenshots Output

- Tambah Antrian

The first screenshot shows a terminal window with a menu: 1. Tambah Antrian, 2. Keluar Antrian, 3. Jumlah Antrian, 4. Lihat Antrian, 5. Hapus Antrian, 0. Keluar. The user selects option 1. The prompt asks for a name and NIM. The user enters 'Galih Trisna' and '2311102050'. The terminal displays 'Berhasil Masuk Antrian'.

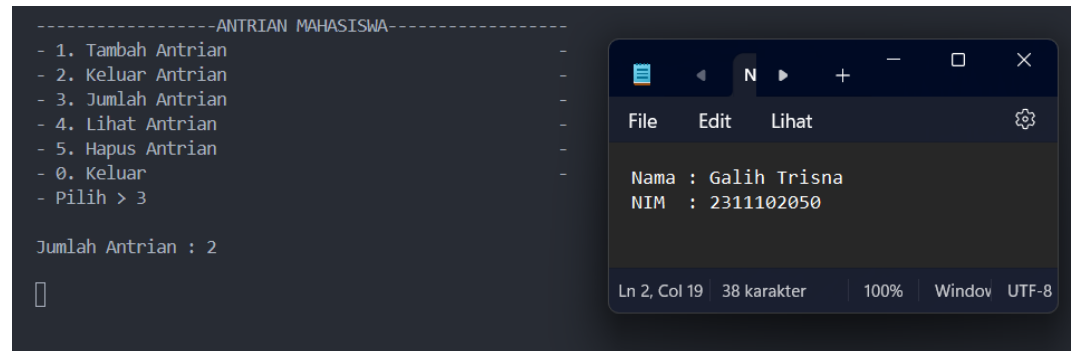
The second screenshot shows the same menu. The user selects option 1. The prompt asks for a name and NIM. The user enters 'Yanto' and '000000000'. The terminal displays 'Berhasil Masuk Antrian'.

The third screenshot shows the same menu. The user selects option 1. The prompt asks for a name and NIM. The user enters 'Budi' and '9999999'. The terminal displays 'Berhasil Masuk Antrian'.

- Keluar Antrian

The screenshot shows a terminal window with the same menu as before. The user selects option 2. The terminal displays 'Berhasil keluar'.

- Jumlah Antrian

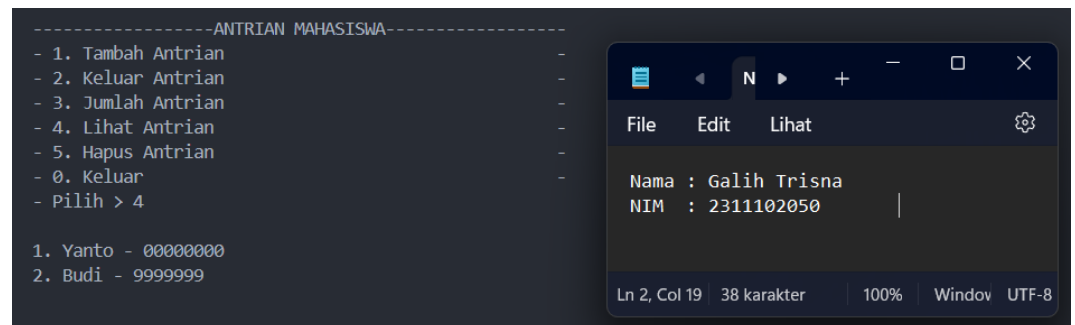


```

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

Jumlah Antrian : 2
  
```

- Lihat Antrian

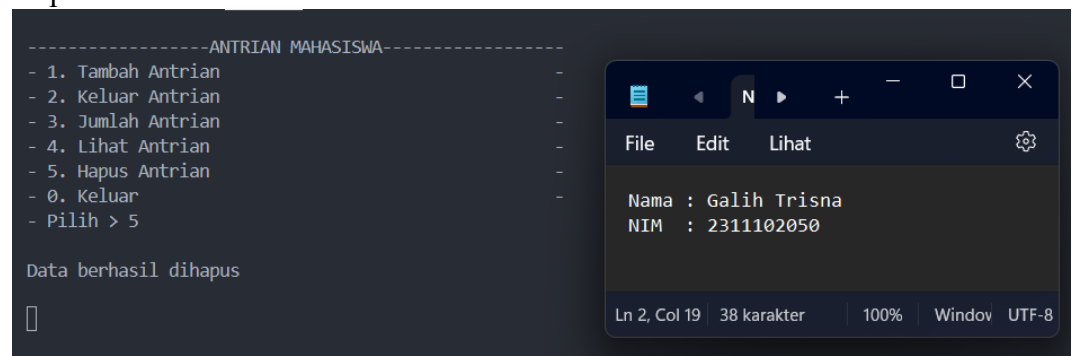


```

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

1. Yanto - 00000000
2. Budi - 9999999
  
```

- Hapus Antrian



```

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

Data berhasil dihapus
  
```

## Deskripsi

Program diatas merupakan program yang hampir sama dengan program sebelumnya. Namun terdapat perbedaan utama antara program unguided 1 dan unguided 2 yaitu pada program ini (unguided 2) node memiliki atribut nama dan nim yang menyimpan data mahasiswa, serta pointer next yang menunjuk ke node berikutnya. Pada program ini juga terdapat menu untuk mengelola daftar antrian mahasiswa. Untuk bagian lainnya dari program ini sama seperti pada program unguided 1 yaitu memiliki fungsi menambahkan elemen (enqueueAntrian), menghapus elemen (dequeueAntrian), menampilkan elemen (viewQueue), menghitung jumlah elemen (countQueue), dan menghapus semua elemen (clearQueue).

#### D. Kesimpulan

Queue (antrian) adalah suatu kumpulan data yang penambahan elemen hanya bisa dilakukan pada suatu ujung (disebut dengan sisi belakang atau rear), dan penghapusan atau pengambilan elemen dilakukan lewat ujung yang lain (disebut dengan sisi depan atau front). Pada queue prinsip yang digunakan adalah FIFO (first in first out).

#### E. Referensi

Asisten Praktikum. (2024). Modul VII : Queue

Programiz. (n.d.). C++ Queue. Diakses pada 23 Mei 2024, dari <https://www.programiz.com/cpp-programming/queue>

Pratama, Riczky. (2023). Queue: Pengenalan, Implementasi, Operasi Dasar, dan Aplikasi. Diakses pada 23 Mei 2024, dari <https://medium.com/@furatamarizuki/queue-pengenalan-implementasi-operasi-dasar-dan-aplikasi-c5eed7e871a3>