

**LAPORAN PRAKTIKUM STRUKTUR DATA
DAN ALGORITMA MODUL 7
“Queue”**



**DISUSUN OLEH:
Farrell Edric Kelvianto
2311102079
S1 IF-11-B**

DOSEN:

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

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

A. Dasar Teori

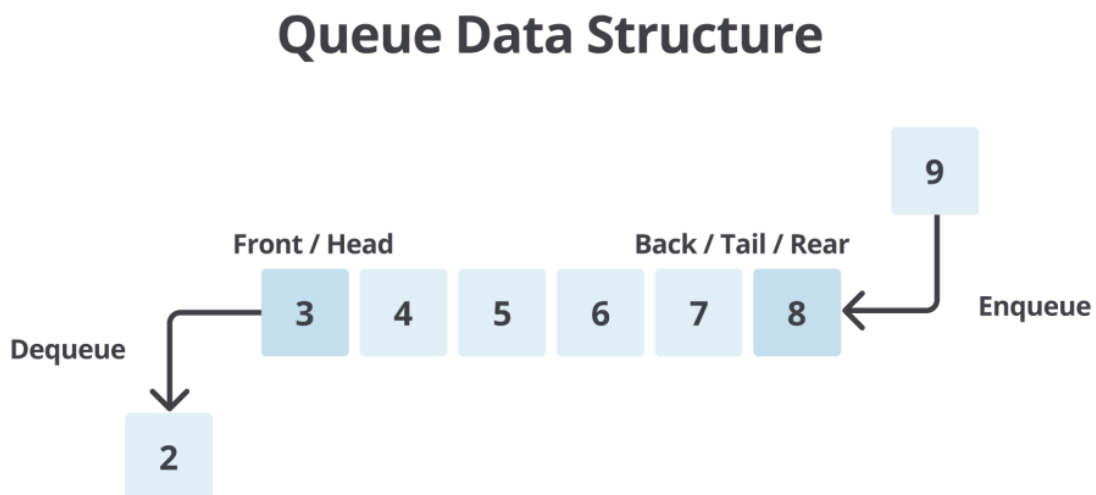
a. Pengertian Queue

Queue adalah struktur untuk menyimpan data dengan cara FIFO yaitu First-in First-Out atau LILO Last-In Last-Out. Data pertama akan dimasukan kedalam queue akan menjadi data yang pertama untuk dikeluarkan dari queue. queue ini mirip dengan konsep antrian jadi analoginya seperti kita mengantri membeli Mie Gacoan atau makanan lain, dan pasti didalam antrian yang paling pertama itu pasti dilayani terlebih dahulu. Untuk melakukan implementasi queue ini dapat dilakukan dengan menggunakan array atau linked list. Untuk perbedaan antara stack dan queue ini terdapat pada penghapusan elemen dan aturan penambahan. Pada stack ini operasi penambahan dan penghapusan elemen ini dilakukan di satu ujung. Elemen yang terakhir diinputkan berada paling dengan ujung atau dipaling atas maka akan dihapus paling awal, karena itulah sifatnya dikenal dengan LIFO. Pada queue ini ditempatkan yang berbeda yaitu melalui salah satu ujung jadi hanya satu insert saja maupun delete.

b. Operasi Basic Queue

Dalam operasi queue ini berbeda dengan teknik hash table seperti kemarin disini ada beberapa operasi khusus untuk stack sesuai

- a. Enqueue () : berfungsi untuk menambahkan data kedalam antrian.
 - b. Dequeue () : berfungsi untuk mengeluarkan data dari antrian.
 - c. Peek () : berfungsi untuk mengambil data didalam antrian tetapi tidak menghapusnya.
 - d. isEmpty () : berfungsi untuk mengecek diantrian kosong atau tidak.
 - e. isFull () : Berfungsi untuk mengecek antrian penuh atau tidak.
 - f. Size () : ini berfungsi untuk menghitung jumlah elemen didalam queue.
- Sebagai berikut adalah gambaran dari suatu queue yang berisikan sebuah antrian:



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
    return back == maksimalQueue;
}
bool isEmpty()
{ // Antriannya kosong atau tidak
    return back == 0;
}

void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "Antrian Penuh " << endl;
    }
    else
    {
        queueTeller[back] = data;
        back++;
    }
}

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] = " ";
    }
}
```

```

        back--;
    }
}
int countQueue()
{ // Fungsi menghitung banyak antrian
    return back;
}
void clearQueue()
{ // Fungsi menghapus semua antrian

    for (int i = 0; i < back; i++)
    {
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

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;
}

```

Screenshot Output

```

PS C:\Users\VICTUS\Documents\Modul VII queue> cd "C:\Users\VICTUS\Documents\Modul VII queue\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
Data antrian teller
1Andi
2Maya
3
4
5
Jumlah antrian = 2
Data antrian teller
1Maya
2. (Kosong)
3
4
5
Jumlah antrian = 1
Data antrian teller
1
2. (Kosong)
3
4
5
Jumlah antrian = 0
PS C:\Users\VICTUS\Documents\Modul VII queue>

```

Deskripsi Program

Pada guided 1 ini merupakan implementasi sederhana dan operasi sederhana untuk menjalankan queue. Di guided 1 ini menggunakan array, diatas terdapat maksimalQueue ini berfungsi untuk membatasi antrian dari Queue ini, front dan back berfungsi untuk menandai antrian awal yaitu 0. Sebagai berikut terdapat beberapa operasi – operasi untuk menjalankan atau memakai program Guided 1 ini:

- isFull() ini berfungsi untuk pengecekan antrian bahwa penuh atau tidak.
- isEmpty() ini berfungsi untuk pengecekan antrian kosong atau tidak, jika kosong maka reutrn.
- enqueueAntrian () ini berfungsi untuk menambah antrian pada queue ini dengan cara menambahkan pada belakang.
- DequeueAntrian () ini berfungsi untuk menghilangkan antrian depan, lalu lanjut pada antrian berikutnya.
- countQueue () ini berfungsi untuk menghitung banyaknya antrian.
- clearQueue () ini berfungsi untuk menghapus semua antrian.
- viewQueue () ini berfungsi untuk menampilkan seluruh data antrian.
- Main () disini berfungsi untuk mengoperasikan perintah dari operasi tadi yang telah dibuat.

B. Tugas

Unguided 1

Soal No 1. Merubah array pada guided 1 menjadi linked list.

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

using namespace std;

const int maksimalQueue = 5; // maksimal

class linkedMhs {
public:
    string mahasiswa;
    long long int nim;
    linkedMhs* next;
    linkedMhs* prev;
};

class node {
private:
    linkedMhs* front;
    linkedMhs* back;
    int size;
public:
    node() {
        front = nullptr;
        back = nullptr;
        size = 0;
    }

    bool isFull() {
        return size == maksimalQueue;
    }

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

    void enqueueAntrian(string namaMhs, long long int nim) {
        if (isFull()) {
            cout << "Antrian penuh" << endl;
            return;
        }
    }
}
```

```

    linkedMhs* NewList = new linkedMhs;
    NewList->mahasiswa = namaMhs;
    NewList->nim = nim;
    NewList->next = nullptr;
    NewList->prev = back;

    if (isEmpty()) {
        front = NewList;
    } else {
        back->next = NewList;
    }
    back = NewList;
    size++;
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian Kosong " << endl;
        return;
    }

    linkedMhs* temp = front;

    if (front == back) {
        front = back = nullptr;
    } else {
        front = front->next;
        front->prev = nullptr;
    }

    delete temp;
    size--;
}

int countQueue() {
    return size;
}

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

```

```

void displayQueue() {
    if (isEmpty()) {
        cout << "Antrian Kosong" << endl;
        return;
    }

    linkedMhs* current = front;
    cout << left << setw(25) << "Nama Mahasiswa" << setw(15) << "NIM" << endl;
    cout << "-----" << endl;
    while (current != nullptr) {
        cout << left << setw(25) << current->mahasiswa << setw(15) << current-
>nim << endl;
        current = current->next;
    }
    cout << "-----" << endl;
}

};

int main() {
    node antrian;
    int pilih;

    while (true) {
        cout << "Menu Tampilan " << endl;
        cout << "1. Menambahkan Antrian " << endl;
        cout << "2. Mengeluarkan Antrian " << endl;
        cout << "3. Menghapus Semua Antrian " << endl;
        cout << "4. Display " << endl;
        cout << "0. Return " << endl;
        cout << "Pilih menu: ";
        cin >> pilih;

        switch (pilih) {
            case 0:
                return 0;
            case 1: {
                string mhs;
                long long int nim;
                cout << "Nama Mahasiswa: ";
                cin.ignore();
                getline(cin, mhs);
                cout << "Nim Mahasiswa: ";
            }
        }
    }
}

```



```

        cin >> nim;
        antrian.enqueueAntrian(mhs, nim);
        break;
    }
    case 2:
        antrian.dequeueAntrian();
        cout << "Antrian Kurang " << antrian.countQueue() << " lagi" <<
endl;

        break;

        break;
    case 3:
        antrian.clearQueue();
        cout << "Semua antrian telah dihapus" << endl;
        break;
    case 4:
        antrian.displayQueue();
        break;
    default:
        cout << "Pilihan Kembali" << endl;
        break;
    }
}

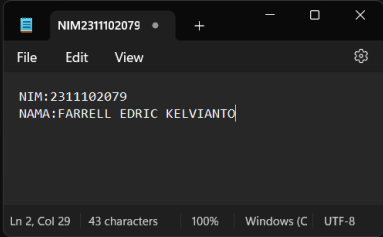
return 0;
}

```

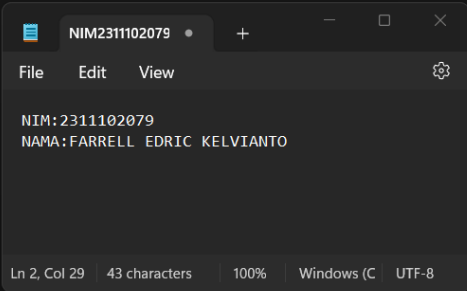
Screenshot Output

Soal No. 2 membuat atribut mahasiswa seperti contoh (Nama dan Nim) mahasiswa dan beserta operasi

```
Nim Mahasiswa: 1233
PS C:\Users\VICTUS\Documents\Modul VII queue> cd "c:\Users\VICTUS\Documents\Modul VII queue\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 1
Nama Mahasiswa: Farrell Edric Kelvianto
Nim Mahasiswa: 2311102079
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 1
Nama Mahasiswa: Gojo Satoru
Nim Mahasiswa: 2311102000
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 1
Nama Mahasiswa: Jason Williams
Nim Mahasiswa: 50
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 2
```



```
Pilih menu: 2
Antrian Kurang 2 lagi
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 4
Nama Mahasiswa      NIM
-----
Gojo Satoru         2311102000
Jason Williams       50
-----
Menu Tampilan
1. Menambahkan Antrian
2. Mengeluarkan Antrian
3. Menghapus Semua Antrian
4. Display
0. Return
Pilih menu: 
```



Deskripsi Program

Di unguided ini berbeda dengan guided tadi yang menggunakan array, disini yang membedakan adalah hanya dibagian list dan element saja secara fungsi sama namun cara kerjanya yang membuat berbeda. Disini ada beberapa deklarasi yang berbeda dari guided yaitu dari cara menampilkan atau fungsi untuk menampilkan dari isi Queue itu sendiri. Cara kerja program ini adalah pertama dibagian maksimum untuk antrian “maksimalQueue” ini untuk membatasi dari antrian itu sendiri ketika penuh dia akan menampilkan output penuh. Sebagai berikut untuk operasi – operasi yang bisa digunakan dalam objek disini untuk dioperasikan di Main ():

- Node () ini berfungsi untuk memanggil variabel yang didalam private.
- isFull () ini berfungsi untuk menyatakan bahwa antrian penuh.
- isEmpty () ini berfungsi untuk menyatakan bahwa jika antrian kosong maka di return.

- enqueueAntrian() ini berfungsi untuk menambahkan antrian dibelakang.
- dequeueAntrian() ini berfungsi untuk menghilangkan antrian didepan
- countQueue() ini berfungsi untuk menghitung jumlah antrian.
- clearQueue() ini berfungsi untuk menghapus semua antrian.
- displayQueue() ini berfungsi untuk menampilkan output semua antrian.
- Main() berfungsi untuk mengoperasikan queue menggunakan operasi yang telah dibuat.

Kesimpulan

Kesimpulannya adalah Queue ini merupakan sebuah konsep antrian yang direalisasikan menggunakan program, didalam queue ini cara kerjanya user menginputkan terlebih dahulu maka dia yang akan keluar atau biasanya disebut FIFO (First In First Out) analoginya ini seperti ambil contoh ketika mengantri di Mie Gacoan atau Restoran Lainnya disini sudah pasti siapa duluan yang mengantri maka dialah yang akan di proses atau di layani terlebih dahulu. Sama dalam konsep program ini siapa yang menginputkan terlebih dahulu maka dialah yang akan diproses terlebih dahulu.

Refrensi

Asisten Praktikum. "Modul 7 Queue". Learning Management System. 2024