# LAPORAN PRAKTIKUM
# STRUKTUR DATA DAN ALGORITME

## MODUL 7

## QUEUE


**Disusun Oleh :**

Nama : Fatkhurrohman Purnomo

NIM : 21102125

**Dosen Pengampu**

Ipam Fuaddina Adam, S.T., M.Kom.



**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**PURWOKERTO**

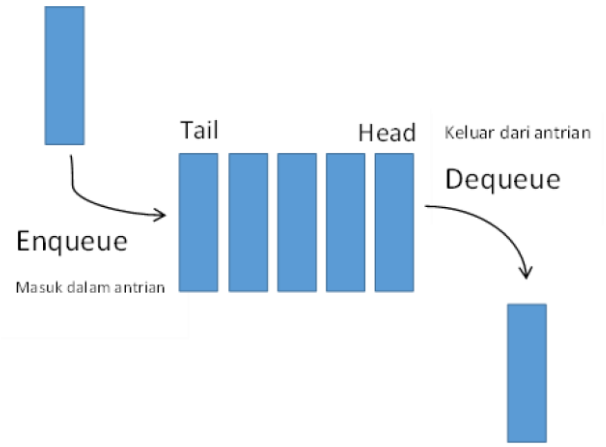
**2022**

# A. Dasar Teori

## Pengertian Queue

Queue (Antrian) merupakan suatu kumpulan data yang penambahan elemennya hanya dapat dilakukan pada suatu ujung (disebut dengan sisi belakang atau tail), dan penghapusan atau pengambilan elemen dilakukan melalui ujung yang lain (disebut dengan sisi depan atau head).

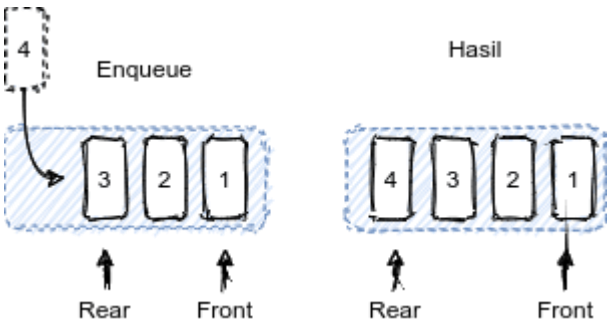
Queue bersifat FIFO (First In First Out) atau FCFS (First Come First Serve), artinya elemen pertama yang masuk antrian (enqueue) adalah yang pertama dilayani atau yang pertama dipindahkan/keluar (dequeue).



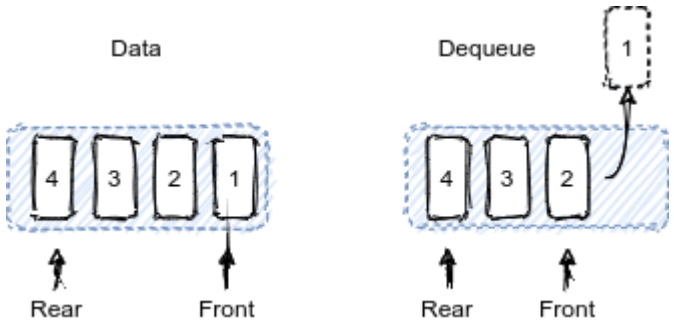

## Proses QUEUE

Seperti halnya pada antrian yang biasa kita lakukan sehari-hari, di manapun. Antrian dimulai dari depan ke belakang, jika didepan belum pergi meninggalkan antrian maka antrian terus bertambah dari belakang dan antrian paling belakang disini dinamakan rear/tail.

Jadi selama antrian terus bertambah (enqueue) maka antrian yang paling akhir adalah tail/rear.



Jika ada yang keluar dari antrian (dequeue) maka data tersebut adalah yang paling depan (head/front), dan data berikutnya setelah data yang keluar berubah menjadi yang paling depan (head/front).




Queue menggunakan metode FIFO, dimana yang masuk pertama kali akan keluar pertama kali juga.

Ref:

Modul 6 : STACK

Contoh Program Queue (Program Antrian) C++ Dan Penjelasannya ~ Coding IsmyNR - Cara Dan Contoh Pemrograman

Implementasi Queue di C++ | Rahmat Subekti

## B. Guided

### 1. Program Queue dengan Struct Array

```
#include <iostream>
#define MAX 5 //menetapkan nilai konstanta MAX = 5
(ukuran antrian)
using namespace std;

///PROGRAM QUEUE DENGAN STRUCTURE ARRAY
//Deklarasi Queue
struct queue{
    int data[MAX];
    int head;
    int tail;
}antrian;

//Buat Queue
void create(){ //membuat antrian kosong
```

```
    antrian.head = antrian.tail = -1;
}

//isEmpty
int isEmpty(){ //mengecek apakah antrian kosong
    if (antrian.tail == -1)
        return 1;
    else
        return 0;
}

//isFull
int isFull(){ //mengecek apakah antrian penuh
    if (antrian.tail == MAX -1)
        return 1;
    else
        return 0;
}

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

//Dequeue
void dequeue(){ //mengambil antrian
    int dq = antrian.data[antrian.head];
    if (isEmpty() == 0){ //jika antrian tidak kosong
        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;
    }
}

//Clear
void clear(){ //mengosongkan seluruh antrian
    antrian.head = antrian.tail = -1;
    cout << " Data clear" << endl;
}

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

int main(){
    int pil;
    int data;

    create();
    do{
        system("cls");
        cout << " 1. Enqueue" << endl;
        cout << " 2. Dequeue" << endl;
        cout << " 3. Tampil" << endl;
        cout << " 4. Clear" << endl;
        cout << " 5. Exit" << endl;
        cout << " Pilihan: ";
        cin >> pil;
```

```
        switch (pil){ // switch case untuk mengakses
fungsi yang diinginkan
        case 1:
            cout << " Data = ";
            cin >> data;
            enqueue(data);
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
        case 4:
            clear();
            break;
        }
        cout << endl;
        system("pause");
    } while (pil != 5);
    return 0;
}
```

**Deskripsi:**

Yang pertama adalah membuat Deklarasi dan inisialisasi dari stack itu sendiri. Kemudian membuat fungsi untuk mengecek apakah stack sedang kosong atau penuh. Dilanjut membuat fungi enqueue yang digunakan untuk menambah data. Fungsi dequeue, untuk mengambil data. Ada clear untuk menghapus semua data, dan display untuk menampilkan semua data. Dilanjut membuat menu pilihan bagi user/pengguna.

**Output:**

```
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan: 3
1
2
3
```

Input yang dimasukkan adalah 1, 2, 3 secara urut. Saat memilih menu 3 maka akan ditampilkan semuanya.

## 2. Program Queue Dengan Linked List

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

///PROGRAM QUEUE DENGAN LINKED LIST
//Delarasi
struct antrian{
    int data;
    antrian *next;
};

antrian *head, *tail, *baru, *bantu, *hapus;

// fungsi untuk menghitung jumlah node (elemen) pada
queue
int hitungAntrian(){
    if(head == NULL){
        return 0;
    } else{
        int counter = 0;
        bantu = head;
        while(bantu != NULL){ // looping untuk menghitung
jumlah elemen pada queue
            counter++;
            bantu = bantu->next;
        }
        return counter;
    }
}

// fungsi untuk mengecek bahwa antrian kosong atau tidak
bool isEmpty(){
    if(hitungAntrian() == 0){
        return true;
    } else{
        return false;
    }
}

// penambahan data antrian
void enqueue(int data){
```

```
    if(isEmpty()){
        head = new antrian();
        head->data = data;
        head->next = NULL;
        tail = head;
        cout << " berhasil menambahkan elemen baru" <<
endl;
    } else{
        baru = new antrian();
        baru->data = data;
        baru->next = NULL;
        tail->next = baru;
        tail = baru;
        cout << " berhasil menambahkan elemen baru" <<
endl;
    }
}

// pengeluaran data antrian
void dequeue(){
    if(isEmpty()){
        cout << " Antrian kosong" << endl;
    } else{
        hapus = head;
        head = head->next;
        hapus->next = NULL;
        delete hapus;
        cout << " berhasil mengeluarkan elemen pertama"
<< endl;
    }
}

// fungsi untuk mencetak nilai pada antrian
void display(){
    cout << " Data Antrian:" << endl;
    if(isEmpty()){
        cout << " Antrian kosong" << endl;
    } else{
        cout << " Jumlah data: " << hitungAntrian() <<
endl;
        bantu = head;
        while(bantu != NULL){ // mencetak data antrian
            cout << bantu->data << endl;
            bantu = bantu->next;
        }
```

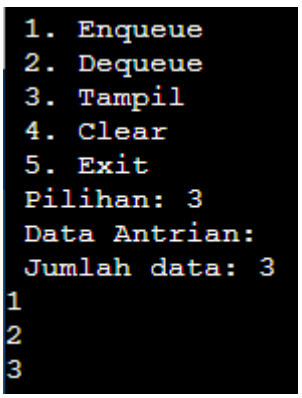
```
    }
    cout << endl;
}

// fungsi untuk menghapus seluruh data pada antrian
void clear(){
    if(isEmpty()){ // jika antrian kosong
        cout << " Antrian Kosong" << endl;
    } else {
        head->next = NULL;
        head = NULL;
        cout << " menghapus seluruh data pada Queue" <<
endl;
        bantu = head;

        while(bantu != NULL){ // menghapus data antrian
            hapus = bantu;
            bantu = bantu->next;
            // menghapus node
            hapus->next = NULL;
            delete hapus;
        }
        head = NULL;
    }
}

int main(){
    int pil;
    int data;

    do{
        system("cls");
        cout << " 1. Enqueue" << endl;
        cout << " 2. Dequeue" << endl;
        cout << " 3. Tampil" << endl;
        cout << " 4. Clear" << endl;
        cout << " 5. Exit" << endl;
        cout << " Pilihan: ";
        cin >> pil;

        switch (pil){ // switch case untuk mengakses
fungsi yang diinginkan
        case 1:
            cout << " Data = ";
            cin >> data;
```

```
            enqueue(data);
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
        case 4:
            clear();
            break;
        }
        cout << endl;
        system("pause");
    } while (pil != 5);
    return 0;
}
```

**Dekripsi:**

Yang pertama adalah membuat Deklarasi dan inisialisasi dari stack itu sendiri, di lanjut dengan pointernya. Membuat fungsi hitungAntrian untuk menghitung total data atau antrian. fungsi isEmpty untuk mengecek apakah data kosong. Fungsi enqueue sama seperti sebelumnya, digunakan untuk menambah data dan dequeue untuk mengeluarkan elemen. Ada fungsi display untuk menampilkan data, dan clear untuk menghapus. Dilanjut membuat menu pilihan bagi user/pengguna.

**Output:**



Input yang dimasukkan adalah 1, 2, 3 secara urut. Saat memilih menu 3 maka akan ditampilkan semuanya.

## C. Tugas (Unguided)

1. **Buatlah program untuk menampung nilai mahasiswa menggunakan konsep queue. Tambahkan fitur yang dapat menghitung banyak data (count), rata-rata nilai (average) dan jumlah nilai (sum).**

```
#include <iostream>

#define MAX 5 //menetapkan nilai konstanta MAX = 5
(ukuran antrian)
using namespace std;

///PROGRAM QUEUE DENGAN STRUCTURE ARRAY
//Deklarasi Queue
struct queue{
    int data[MAX];
    int head;
    int tail;
}antrian;

//Buat Queue
void create(){ //membuat antrian kosong
    antrian.head = antrian.tail = -1;
}

//isEmpty
int isEmpty(){ //mengecek apakah antrian kosong
    if (antrian.tail == -1)
        return 1;
    else
        return 0;
}

//isFull
int isFull(){ //mengecek apakah antrian penuh
    if (antrian.tail == MAX -1)
        return 1;
    else
        return 0;
}
```

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

//Dequeue
void dequeue(){ //mengambil antrian
    int dq = antrian.data[antrian.head];
    if (isEmpty() == 0){ //jika antrian tidak kosong
        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;
    }
}

//Clear
void clear(){ //mengosongkan seluruh antrian
    antrian.head = antrian.tail = -1;
    cout << " Data clear" << endl;
}

//Tampil
void display(){ //menampilkan data antrian
```

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

// menghitung jumlah data (count)
int count(){
    int count = 0;
    if (isEmpty() == 0){
        for (int i = antrian.head; i <= antrian.tail;
i++){
            count++;
        }
        cout << " Jumlah data = " << count << endl;
    }
    return count;
}

// menghitung rata-rata nilai (average)
void average(){
    int sum = 0;
    int count = 0;
    if (isEmpty() == 0){
        for (int i = antrian.head; i <= antrian.tail;
i++){
            sum += antrian.data[i];
            count++;
        }
    }
    cout << " Rata-rata nilai = " << sum/count << endl;
}

// menghitung jumlah nilai (sum)
void sum(){
    int sum = 0;
    if (isEmpty() == 0){
        for (int i = antrian.head; i <= antrian.tail;
i++){
            sum += antrian.data[i];
```

```
        }
    }
    cout << " Jumlah nilai = " << sum << endl;
}

int main(){
    int pil;
    int data;

    create();
    do{
        system("cls");
        cout << " 1. Enqueue" << endl;
        cout << " 2. Dequeue" << endl;
        cout << " 3. Tampil" << endl;
        cout << " 4. Clear" << endl;
        cout << " 5. Jumlah data" << endl;
        cout << " 6. Rata-rata" << endl;
        cout << " 7. Jumlah nilai" << endl;
        cout << " 8. Exit" << endl;
        cout << " Pilihan: ";
        cin >> pil;

        switch (pil){ // switch case untuk mengakses
fungsi yang diinginkan
        case 1:
            cout << " Data = ";
            cin >> data;
            enqueue(data);
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
        case 4:
            clear();
            break;
        case 5:
            count();
            break;
        case 6:
            average();
            break;
```

```
        case 7:
            sum();
            break;
        }
        cout << endl;
        system("pause");
    } while (pil != 8);
    return 0;
}
```

**Deskripsi:**

Yang pertama adalah melakukan inisialisasi stack, lalu mengecek stack kosong atau stack penuh. Membuat fungsi pop and push untuk menambah dan menghapus data di stack. Lalu melakukan reverse atau membalik input yang dimasukan, kemudian langsung ditampilkan.

**Output:**

Dengan menggunakan inputan 1, 2, 3 maka hasil jumlah data, rata-rata, dan jumlah nilai, yaitu:

a. Jumlah data

```
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Jumlah data
6. Rata-rata
7. Jumlah nilai
8. Exit
Pilihan: 5
Jumlah data = 3
```

b. Rata-rata

```
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Jumlah data
6. Rata-rata
7. Jumlah nilai
8. Exit
Pilihan: 6
Rata-rata nilai = 2
```

c. Jumlah nilai

```
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Jumlah data
6. Rata-rata
7. Jumlah nilai
8. Exit
Pilihan: 7
Jumlah nilai = 6
```

**2. Buatlah program untuk membantu operator administrasi event Meet & Greet Idol Group dalam mencetak nomor antrian penggemar dengan menggunakan konsep queue!**

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

///PROGRAM QUEUE DENGAN LINKED LIST
//Delarasi
struct antrian{
    string data;
    antrian *next;
};

antrian *head, *tail, *baru, *bantu, *hapus;

// fungsi untuk menghitung jumlah node (elemen) pada
queue
int hitungAntrian(){
    if(head == NULL){
        return 0;
    } else{
        int counter = 0;
        bantu = head;
        while(bantu != NULL){ // looping untuk menghitung
jumlah elemen pada queue
            counter++;
            bantu = bantu->next;
        }
        return counter;
    }
}
```

```
// fungsi untuk mengecek bahwa antrian kosong atau tidak
bool isEmpty(){
    if(hitungAntrian() == 0){
        return true;
    } else{
        return false;
    }
}

// jika antrian sudah penuh, maka tidak dapat menambahkan
data
void enqueue(string data){
    if(isEmpty()){
        head = new antrian();
        head->data = data;
        head->next = NULL;
        tail = head;
    } else{
        baru = new antrian();
        baru->data = data;
        baru->next = NULL;
        tail->next = baru;
        tail = baru;
    }
}

// antrian makimal 20 orang
void maksimal(string nama){
    if (hitungAntrian() == 20){
        cout << endl << " Antrian penuh" << endl;
    } else{
        enqueue(nama);
        cout << " Nama : " << nama << endl;
        cout << " No. Antrian : " << hitungAntrian() <<
"/20" << endl;
        cout << " Estimasi waktu personal meet & great
adalah 5 menit." << endl;
        cout << " Silahkan tunggu " << hitungAntrian() *
5 << " menit lagi untuk tiba giliran anda." << endl;
        cout << endl << " Terima kasih." << endl;

    }
}
```

```
// rilis antrian
void rilisAntrian(){
    if(isEmpty()){
        cout << endl << " Antrian kosong" << endl;
    } else{
        cout << endl;
        cout <<
"=========================================" << endl;
        cout << "|               Antrian
Rilis              |" << endl;
        cout <<
"=========================================" << endl;
        cout << " Nama              : " << head->data <<
endl;
        cout << " No. Antrian       : " <<
hitungAntrian() << endl;
        cout << " Telah keluar" << endl;
        cout <<
"=========================================" << endl;
        hapus = head;
        head = head->next;
        hapus->next = NULL;
        delete hapus;
    }
}

// reset antrian
void resetAntrian(){
    if(isEmpty()){
        cout << " Antrian kosong" << endl;
    } else{
        bantu = head;
        while(bantu != NULL){
            hapus = bantu;
            bantu = bantu->next;
            delete hapus;
        }
        head = NULL;
        tail = NULL;
        cout << " Antrian berhasil direset" << endl;
    }
}

int main(){
    int pil;
```

```
    string nama;

    do{
        system("cls");
        cout << endl;
        cout <<
"=========================================" << endl;
        cout << "|               Cetak
Antrian              |" << endl;
        cout <<
"=========================================" << endl;
        cout << " 1. Cetak No. antrian" << endl;
        cout << " 2. Rilis antrian" << endl;
        cout << " 3. Reset antrian" << endl;
        cout << " 4. Keluar" << endl;
        cout << " Pilihan: ";
        cin >> pil;

        switch (pil){ // switch case untuk mengakses
fungsi yang diinginkan
        case 1:
            cout << " Masukan nama : ";
            cin >> nama;
            cout << endl;
            maksimal(nama);
            break;
        case 2:
            rilisAntrian();
            break;
        case 3:
            resetAntrian();
            break;
        }
        cout << endl;
        system("pause");
    } while (pil != 4);
    return 0;
}
```

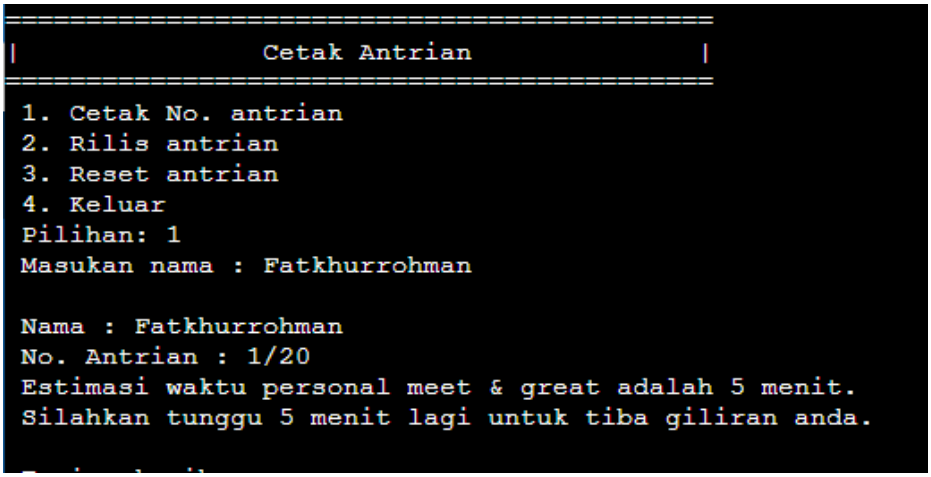
**Deskripsi:**

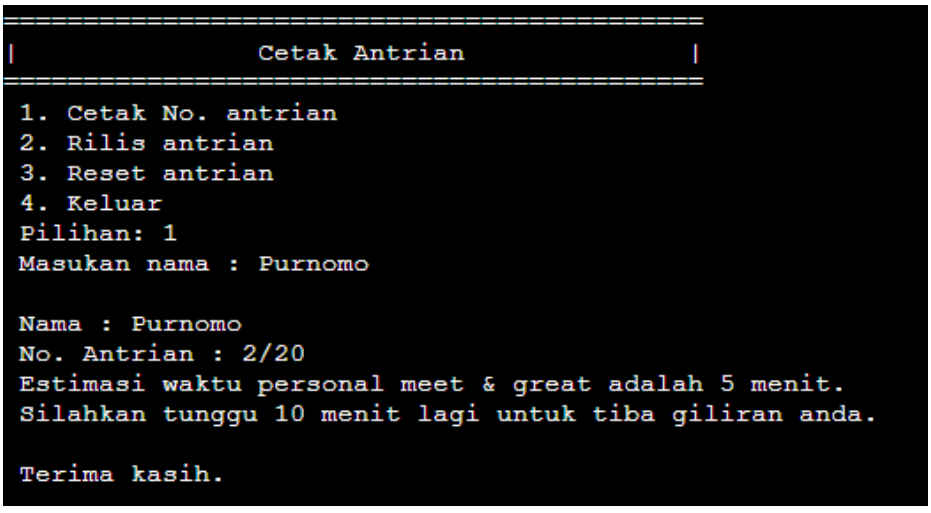
Program queue dengan linked list untuk membuat data antrian. Yang pertama deklrasi dan inisialisasi stuct dilanjut pointe. Fungsi menghitungAntrian untuk menghitung jumlah antrian. Fungsi isEmpty untuk mengecek apa  array kosong atau tidak, digunakan pada saat mau

membuat antrian. Fungi enqueue digunakan untuk menambahkan antrian ke array. Fungsi maksimal untuk membatasi agar maksimal antrian yang bisa adalah 20. Fungsi rilisAntrian digunakan untuk rilis antrian. Fungsi resetAntrian digunakan untuk resert atau menghapus semua antrian yang di input. Yang terakhir menu untuk memanggil fungsi-fungsi yang sudah dibuat.

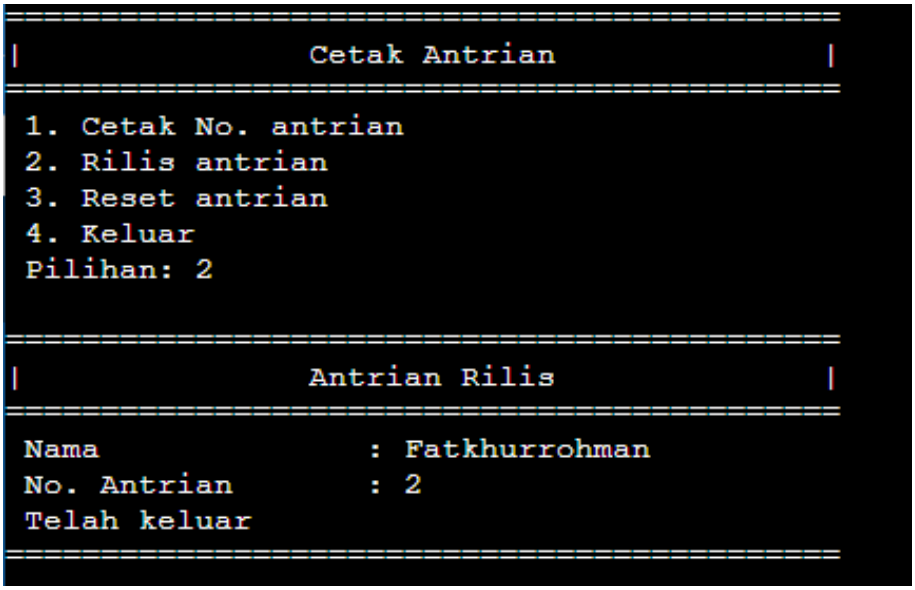
**Output:**

a. Cetak antrian




b. Rilis antrian




c. Reset antrian


## D. Kesimpulan

1. Bisa membuat queue
2. Belajar lebih dalam stack dan struct untuk membuat queue
3. Lebih mahir dalam menggunakan bahasa C++
4. Bisa melakukan problem solving bagi program yang error
5. Lebih paham dalam membuat program
6. Melatih daya pikir, imajinasi, dan langkah-langkah dalam membuat program
7. Bisa membuat program antrian