A. Tujuan Pembelajaran

Mahasiswa mampu menjelaskan pengertian queue dan dequeue

- Mahasiswa mampu menjelaskan dan menunjukkan cara pembuatan queue, operasi push dan pop pada array
- Mahasiswa mampu menjelaskan dan menunjukkan program dengan ADT (Abstract Data Type) queue dan dequeue dengan array

B. Dasar Teori

QUEUE

Queue atau antrian adalah suatu kumpulan data yang penambahan elemennya 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).

Kalau tumpukan dikenal dengan menggunakan prinsip LIFO (Last In First Out), maka pada antrian prinsip yang digunakan adalah FIFO (First In First Out).

Implementasi Antrian dengan Array

Untuk memahami penggunaan antrian dalam array, kita membutuhkan deklarasi antrian, misalnya:

```
#define MAXN 6
typedef enum {NOT_OK, OK} Tboolean;
typedef struct {
    Titem array[MAXN];
    int first;
    int last;
    int number_of_items;
} Tqueue;
```

Dengan deklarasi di atas, elemen antrian dinyatakan dalam tipe integer yang semuanya terdapat dalam struktur. Variabel first menunjukkan posisi elemen pertama dalam *array*, dan variable last menunjukkan posisi elemen terakhir dalam *array*.

Algoritma dari penggalan program di atas adalah:

1. Tentukan elemen yang akan dimasukkan ke dalam antrian (dalam hal ini adalah 6 elemen)
2. Deklarasikan struktur untuk menampung elemen pada antrian
3. Selesai

Untuk menambah elemen baru dan mengambil elemen dari antrian dalam antrian, diperlukan deklarasi berikut ini:

```
void initialize_queue (Tqueue *Pqueue) {
    Pqueue->first = 0;
    Pqueue->last= -1;
    Pqueue->number_of_items = 0;
}
Tboolean enqueue (Tqueue *Pqueue, Titem item) {
    if (Pqueue->number_of_items>=MAXN)
        return(NOT_OK);
    else {
        Pqueue->last++;
        if (Pqueue->last>MAXN-1)
            Pqueue->last=0;
        Pqueue->array[Pqueue->last]=item;
        Pqueue->number_of_items++;
        return (OK);
    }
}

Tboolean dequeue (Tqueue *Pqueue, Titem *Pitem){
    if (Pqueue ->number_of_items == 0)
        return (NOT_OK);
    else {
        *Pitem = Pqueue->array[Pqueue->first++];
        if (Pqueue->first>MAXN-1)
            Pqueue->first=0;
        Pqueue->number_of_items--;
        return(OK);
    }
}
```

Implementasi Antrian dengan Pointer

Untuk mengimplementasikan antrian dengan menggunakan pointer, perhatikan algoritma berikut ini:

1. Tentukan struktur untuk menampung node yang akan dimasukkan pada antrian. Deklarasi struktur pada penggalan program berikut ini:

```
struct queueNode {
    char data;
    struct queueNode *nextPtr;
};
typedef struct queueNode QUEUENODE;
typedef QUEUENODE *QUEUENODEPTR;
QUEUENODEPTR headPtr=NULL, tailPtr=NULL;
```

2. Deklarasikan penambahan elemen baru pada antrian, di mana letaknya adalah paling belakang. Deklarasi penambahan elemen baru tersebut dapat dilihat pada penggalan program berikut ini:

```
void enqueue (QUEUENODEPTR *headPtr, QUEUENODEPTR *tailPtr,
              char value) {
    QUEUENODEPTR newPtr=malloc (sizeof(QUEUENODE));
    if(newPtr!=NULL){
        newPtr->data=value;
        newPtr->nextPtr=NULL;
        if(isEmpty(*headPtr))
            *headPtr=newPtr;
        else
            (*tailPtr)->nextPtr=newPtr;
        *tailPtr=newPtr;
    }
```

3. Lakukan pengecekan terhadap antrian, apakah antrian dalam kosong atau tidak. Kalau kondisi antrian kosong, maka elemen bisa dihapus. Penggalan program berikut ini akan menunjukkan kondisi tersebut.

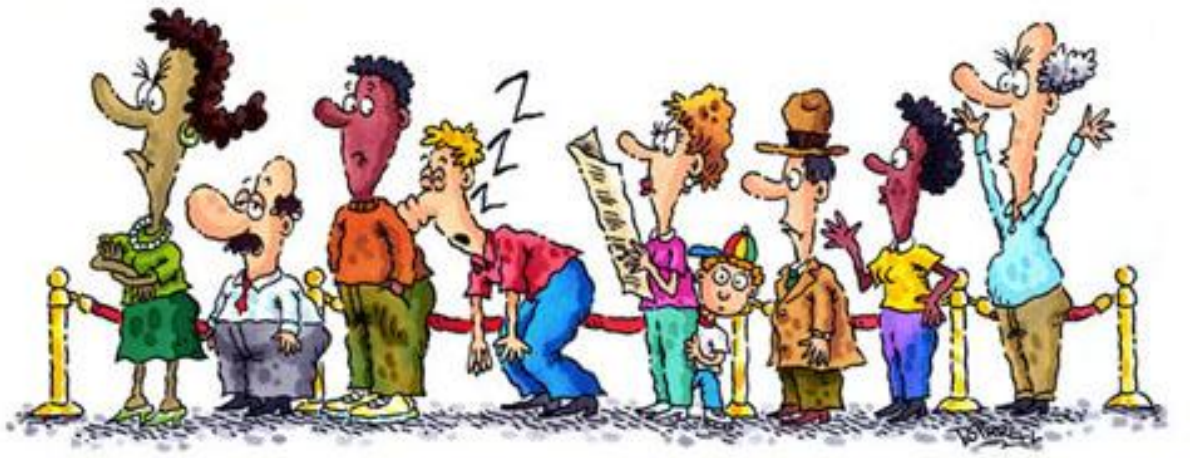
```
int isEmpty(QUEUENODEPTR headPtr) {
    return headPtr==NULL;
}
```

C. Latihan

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define max 10

using namespace std;

int antrian[1000000]; //array untuk menampung data antrian
int head = 0 , tail = 0 , cur = 0;
//head -> data antrian paling depan
//tail -> data antiran paling belakang
//cur  -> jumlah data pada antrian

bool full();
bool empty();
int push(int data);
int pop();
int print();

int main(){
    int pilihan,data;
    do{
        system("cls");
        printf("\n----------MENU----------\n");
        printf("1. PUSH Data Ke Antrian \n");
        printf("2. POP Data Dari Antrian \n");
        printf("3. Print Data Di Antrian \n");
        printf("4. Exit \n\n");
        printf("Tentukan Pilihanmu : ");
        scanf("%d",&pilihan);
        switch(pilihan){
            case 1 :
                if(!full()){
                    printf("---PUSH DATA---\n");
                    printf("Masukkan Data : ");
                    scanf("%d",&data);
                    push(data);
                }
                else{
                    printf("Antrian Penuh !!!\n");
                    getch();
                }
                break;
            case 2:
                if(!empty()){ //sama kayak if(empty() == false/0) kalo -> if(empty()) sama kayak if(empty() == true/1)
                    printf("---POP DATA---\n");
                    printf("Data yang dihapus : %d\n", pop()); //fungsi pop() return data yang di hapus, kemudian di print
                    getch();
                }
                else{
                    printf("ANTRIAN KOSONG !!!\n");
                    getch();
                }
                break;
            case 3:
                printf("DATA ANTRIAN : \n");
                print();
                break;
            case 4:
                printf("Keluar. . .\n");
                break;
            default :
                printf("PILIHAN TIDAK ADA !!!\n");
                getch();
                break;
        }
    }while(pilihan != 4);
    getch();
}
```

```
bool full(){
    return (cur >= max) ? true : false; //cek apakah cur >= max ? , kalo benar return true kalo salah return false
}

bool empty(){
    return (cur == 0) ? true : false;
}

int push (int data){
    antrian[tail] = data;
    tail++;
    cur++;
}

int pop(){
    int data = antrian[head];
    head++;
    cur--;
    return data;
    getch();
}

int print(){
    for (int i= head ; i< tail ; i++){
        printf("%d ",antrian[i] );
    }
    getch();
}
```

D. Tugas Praktikum



Sebuah bank membutuhkan program untuk melakukan antrian data , buatlah program tersebut dengan metode queue.

Syarat:

- Menggunakan array atau linked list
- Ada 2 menu berbeda untuk teler dan nasabah

E. Tugas

Buatlah sebuah program yang dapat menghitung waktu tunggu pasien pada saat mengantri untuk berobat. Gunakan algoritma queue
Minimal program dapat melakukan hal berikut :