JURNAL PRAKTIKUM
(LAB. ACTIVITY)

STRUKTUR DATA
SI025

Materi 11 :
QUEUE
(ANTREAN)

Dosen:

Agung Nugroho, M.Kom
Prof. Dr. Ema Utami, S.Si, M.Kom
Ikmah, M.Kom
Ninik Tri H, M.Kom
Lilis Dwi Farida, M.Kom
Windha Mega PD, M.Kom

S1 – SISTEM INFORMASI
UNIVERSITAS AMIKOM YOGYAKARTA
2018

Queue(Antrean)
Pendahuluan

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami konsep Queue
2. Memahami operasi dasar Queue
3. Menerapkan Queue menggunakan bahasa C++

B. Peralatan

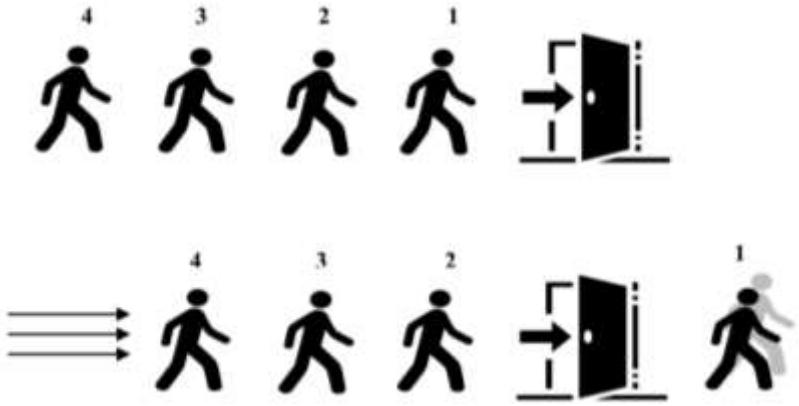
1. PC Desktop
2. Windows 7
3. MinGW

C. Teori

QUEUE (ANTREAN)

Queue nama lainnya adalah antrean. Merupakan sekumpulan data yang mengalami penambahan data(elemen) melalui satu sisi, yaitu depan (head) dan penghapusan data(elemen) melalui sisi belakang (tail). Sifat tersebut biasa disebut dengan FIFO (First In First Out), yaitu data yang pertama masuk akan keluar terlebih dahulu. Dan data yang terakhir masuk akan keluar paling akhir. Perhatikan ilustrasi pada Gambar 1.

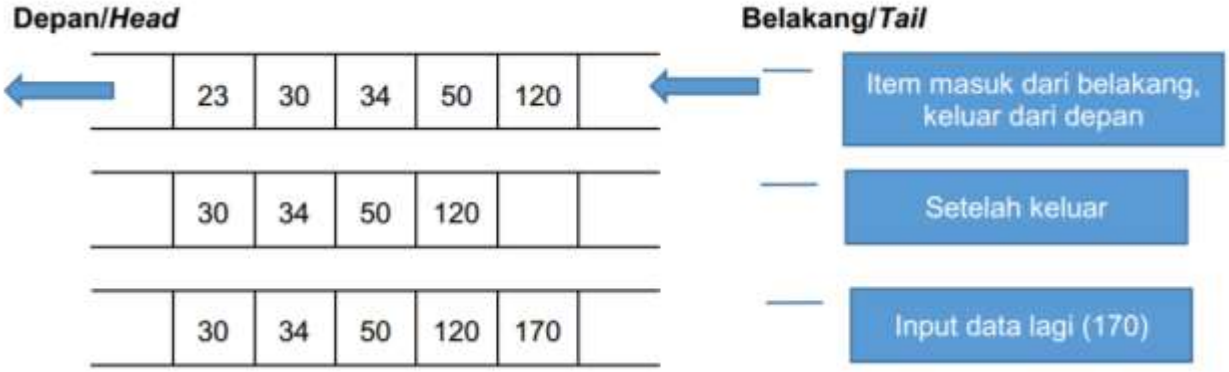
Elemen yang pertama kali masuk ke queue disebut dengan elemen depan (front/head of queue), sedangkan elemen yang terakhir kali masuk ke queue disebut dengan elemen belakang (rear/tail of queue).



Gambar 1. Ilustrasi Queue

Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu tempat atau satu ujung. Sementara pada queue operasi penambahan dapat dilakukan di tempat yang berbeda.

Penambahan elemen dilakukan di salah satu ujung, biasa disebut dengan elemen depan, dan ujung lainnya, yang biasa disebut dengan ujung belakang digunakan untuk penghapusan data. Sifat yang demikian biasa dikenal dengan FIFO. Ilustrasi operasi queue seperti pada Gambar 2.



Gambar 2. Operasi Queue

Operasi standard pada queue antara lain:

1. Membuat queue atau inisialisasi;
2. Mengecek apakah queue penuh/full;
3. Mengecek apakah queue kosong/empty;
4. Memasukkan elemen ke dalam queue dan InQueue (Insert Queue);
5. Menghapus elemen dari queue atau DeQueue (Delete Queue).

Operasi pada queue atau antrean:

1. Fungsi init()

Digunakan untuk menciptakan queue yang baru atau kosong, yaitu dengan memberi nilai awal (head) dan nilai akhir (tail) dengan -1.

```
void init(void)
{
    antre.awal= -1;
    antre.akhir= -1;
}
```

2. Fungsi full()

Berguna untuk mengecek apakah antrean sudah penuh atau belum. Dilakukan dengan cara mengecek nilai nilai akhir (tail) apakah sudah sama dengan nilai maksimal dari queue. Jika nilai tail sama dengan nilai maksimal, maka fungsi akan mengembalikan nilai true dan jika tidak, maka fungsi akan mengembalikan nilai false

```
bool full(void)
{
    if(antre.akhir== MAX-1){
       return true;
    } else {
       return false;
    }
}
```

3. Fungsi empty()

Fungsi ini berguna untuk mengecek apakah queue masih kosong atau sudah terisi data. Proses pengecekan dilakukan dengan memeriksa nilai akhir (tail) apakah bernilai nol atau tidak. Jika tail = -1, maka nilai yang dikembalikan adalah true, dan sebaliknya. Head (kepala antrean) tidak perlu diperiksa lagi karena tidak akan berubah. Hal ini disebabkan karena pergerakan pada queue terjadi di elemen tail.

```
bool empty(void)
{
    if(antre.akhir== -1){
       return true;
    } else {
       return false;
    }
}
```

4. Fungsi inQueue()

Digunakan untuk memasukkan elemen ke dalam queue. Penambahan queue terjadi di elemen paling akhir. Jika queue belum penuh, maka nilai akhir (tail) akan ditambah 1(increment)

```
void inQueue(){
    tampilData();
    int elemen;

    if(!full()){
        cout<<"Data yang akan dimasukkan : ";
        cin>>elemen;
        cout<<"Data berhasil ditambahkan\n";
        antrean.data[antrean.akhir]=elemen;
        antrean.akhir++;
    } else {
        cout<<"Queue penuh\n";
    }
    getchar();
}
```

5. Fungsi deQueue()

Berguna untuk mengambil elemen dari queue, dengan cara memindahkan semua elemen satu langkah ke posisi di depannya. Sehingga elemen yang paling depan akan tertimpa. Pergeseran dilakukan dengan menggunakan perulangan

```
void deQueue(){
    int i;

    if(!empty()){
        for(i=antre.awal; i<antre.akhir; i++){
            antre.data[i]=antre.data[i+1];
        }
        antre.akhir--;
    } else {
        cout<<"Antrean kosong";
    }
}
```

6. Fungsi clear()

Berguna untuk menghapus elemen-elemen queue dengan cara membuat Tail dan Head = -1. Penghapusan elemen queue sebenarnya tidak menghapus array-nya, tetapi hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen-elemen antrean tidak lagi terbaca.

```
void clear(){
    antre.awal = -1;
    antre.akhir = -1;
}
```

D.I Praktikum

Latiham 1.1

Program lengkap dari contoh penerapan queue adalah sebagai berikut :

```
#include<iostream>
#include<cstdlib>
#define MAX 5

using namespace std;

struct queue{
    int data[MAX];
    int awal,akhir;
}antrean;

void init(){
    antrean.awal= -1;
    antrean.akhir= -1;
}

bool full(){
    if(antrean.akhir == MAX-1){
        return true;
    } else {
        return false;
    }
}

bool empty(){
    if(antrean.akhir== -1){
        return true;
    } else {
        return false;
    }
}

void tampilData(){
    int i;
    if(!empty()){
        for(i=antrean.awal; i<antrean.akhir; i++){
            cout<<antrean.data[i]<<" | ";
        }
    }
    cout<<"\n";
}
void inQueue(){
    tampilData();
    int elemen;

    if(!full()){
        cout<<"Data yang akan dimasukkan : ";
        cin>>elemen;
        cout<<"Data berhasil ditambahkan\n";
        antrean.data[antrean.akhir]=elemen;
        antrean.akhir++;
```

```
    } else {
        cout<<"Queue penuh\n";
    }
    getchar();
}

void deQueue(){
    int i;
    tampilData();

    if(!empty()){
    cout<<"\nMengambil data \" "<<antrean.data[antrean.awal]<<" \"..."<<endl;
        for(i=antrean.awal; i<antrean.akhir; i++){
            antrean.data[i]=antrean.data[i+1];
        }
        antrean.akhir--;
    } else {
        cout<<"Antrean kosong";
    }
    getchar();
}

void clear(){
    antrean.awal = -1;
    antrean.akhir = -1;
}


int main(){
    int pilihan, elemen;
    init();
    cout<<"Demo Queue dengan Linear Array"<<endl;
    do{
        tampilData();
        cout<<"\nMenu Utama\n";
        cout<<"================\n";
        cout<<"[1] Init \n[2] InQueue \n[3] DeQueue \n[4] Clear \n[0] Keluar\n";
        cout<<"================\n";
        cout<<"\nMasukkan pilihan: ";cin>>pilihan;
        cout<<"================\n";
        switch(pilihan){
            case 1: init(); break;
            case 2: inQueue(); break;
            case 3: deQueue(); break;
            case 4: clear(); break;
        }
    }
    while(pilihan!=0);

    return 0;
}
```

Latiham 1.2

```
typedef struct queue{
    int head;
    int tail;
    char data[10][20]; //menampung 10 data dengan jumlah string max 20 huruf
} antrean;

queue antre;
void init(){
    antre.head = antre.tail = -1;
}

int isFull(){
    if(antre.tail == max-1)
        return 1;
    else
        return 0;
}

int isEmpty(){
    if(antre.tail == -1){
        antre.head = -1;
        return 1;
    } else {
        return 0;
    }
}

void inQueue(char d[10]){
    antre.head = 0;
    antre.tail++;
    strcpy(antre.data[antre.tail],d);
}

void deQueue(){
    cout<<"data diambil"<<" "<<antre.data[antre.head]<<endl;
    for(int i=antre.head; i<antre.tail; i++){
        strcpy(antre.data[i], antre.data[i+1]);
        antre.tail--;
    }
}

void clear(){
    antre.head=antre.tail = -1;
    cout<<"Semua data terhapus\n";
}

void print(){
    for(int i=0;i<=antre.tail; i++){
        cout<<"Data ke-"<<i+1<<". "<<antre.data[i]<<endl;
    }
}
```

```
int main(){
    int pilih;
    init();
    char value[20];

    do{
        cout<<"\n1. input\n";
        cout<<"2. delete\n";
        cout<<"3. print\n";
        cout<<"4. clear\n";
        cout<<"5. exit\n";
        cout<<"Pilihan : ";cin>>pilih;

        switch(pilih){
            case 1: if(isFull()!=1){
                        cout<<"Data : ";cin>>value;
                        inQueue(value);
                    } else {
                        cout<<"\nQueue sudah penuh!\n";
                    }
                    break;
            case 2: if(isEmpty() != 1){
                        deQueue();
                    } else {
                        cout<<"\nMasih kosong\n";
                    }
                    break;
            case 3: if(isEmpty() != 1){
                        print();
                    } else {
                        cout<<"\nMasih kosong\n";
                    }
                    break;
            case 4: clear();
                        cout<<"\nSudah kosong\n";
                    break;
        }
    }
    while(pilih != 5);
    getchar();
}
```

E. Tugas

1. Tambahkan function untuk mencari suatu elemen di dalam queue yang sudah diinputkan.
2. Carilah nilai total, rata-rata, terbesar dan terkecil dari elemen-elemen queue dalam function tersendiri