MODUL 7

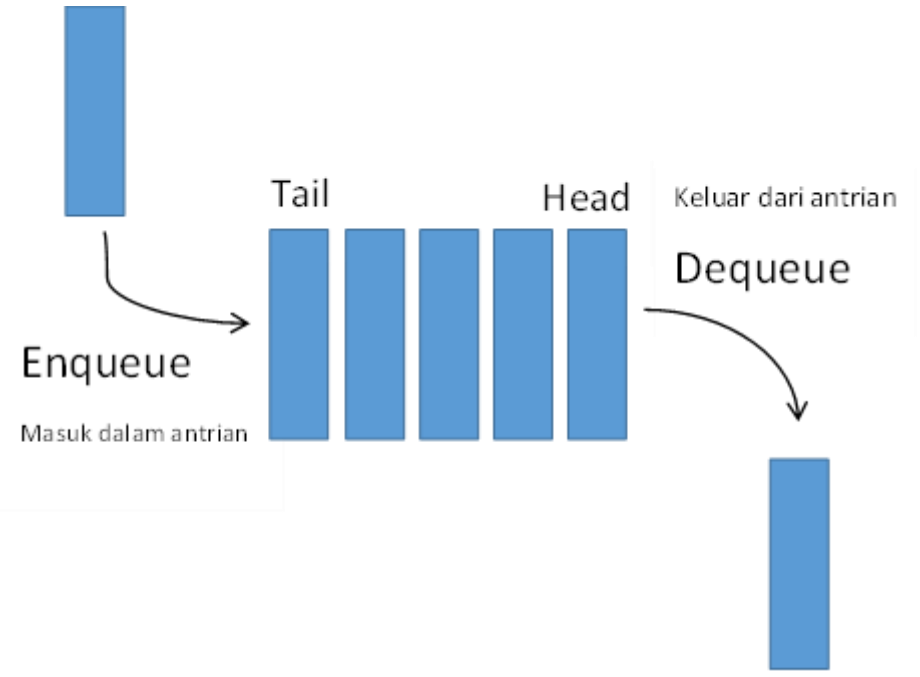
QUEUE

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*).



Gambar 1 Ilustrasi sebuah Queue

Operasi Dasar Pada Queue

- **Deklarasi**

  Pendeklarasian queue menggunakan tipe data struct.

```
struct Queue {
    int data[MAX];
    int head;
    int tail;
}antrian;
```

- **Create**

  Operasi untuk membuat dan menginisialisasi sebuah queue kosong dengan cara membuat Head dan Tail = -1.

```
void create(){
    antrian.head = antrian.tail = -1;
}
```

- **isEmpty**

  Operasi untuk memeriksa apakah suatu queue masih kosong. Queue kosong ditandai dengan nilai Tail kurang dari nol (-1).

```
int isEmpty()
{
      if (antrian.tail == -1)
            return 1; //true
      else
            return 0; //false
}
```

- **isFull**

  Operasi untuk memeriksa apakah queue yang ada sudah penuh. Queue akan mengindikasikan penuh jika ujung antrian (tail) mendekati nilai maksimum yang dapat ditampung antrian (MAX-1).

```
int isFull()
```

```
{
      if (antrian.tail == MAX-1)
            return 1; //true
      else
            return 0; //false
}
```

- **Enqueue**

  Operasi untuk menambahkan satu elemen ke dalam queue dan tidak dapat dilakukan jika antrian dalam keadaan penuh.

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

- **Dequeue**

  Operasi untuk mengambil atau menghapus data terdepan (head) dari queue.

```
int dequeue()
{
    int dq = antrian.data[antrian.head];
```

```
    if (isEmpty() == 0){
        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**

  Operasi untuk menghapus atau mengosongkan seluruh data queue dengan cara membuat Tail dan Head = -1 seperti sedia kala.

```
void clear()
{
      antrian.head = antrian.tail = -1;
}
```

- **Display**

  Operasi untuk menampilkan seluruh data queue.

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

```
    }
}
```

GUIDED

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){
        antrian.head = antrian.tail = 0;
        antrian.data[antrian.tail] = data;
        cout << " " << antrian.data[antrian.tail] << " masuk!" << endl;
    }
    else if (isFull() == 0){
        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){
        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){
            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;
}
```

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){
            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){
            cout << bantu->data << endl;
            bantu = bantu->next;
```

```
        }
    }
    cout << endl;
}

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

        bantu = head;
        while(bantu != NULL){
            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){
            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;
}
```

TUGAS

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*)**. **[Bobot 45]**

2. Buatlah program untuk membantu operator administrasi event Meet & Greet Idol Group dalam mencetak nomor antrian penggemar dengan menggunakan konsep queue! **[Bobot 55]**

   Contoh output program:

```
PROGRAM CETAK NO ANTRIAN

1. CETAK NO. ANTRIAN
2. RILIS ANTRIAN
3. RESET ANTRIAN
```

```
-CETAK NO. ANTRIAN-

Masukkan Nama: ...

Nama        : ...
No. Antrian : .../(kuota antrian)
Estimasi waktu personal meet & greet adalah ... menit.
Silahkan tunggu ... menit lagi untuk tiba giliran Anda.

Terima kasih.

press any key to continue...
```

```
-RILIS ANTRIAN-

Nama        : ...
No. Antrian : ...
telah keluar.

press any key to continue...
```

```
-RESET ANTRIAN-

Data antrian berhasil dihapus.

press any key to continue...
```

Contoh Output:

```
-CETAK NO. ANTRIAN-

Masukkan Nama: Alex

Nama        : Alex
No. Antrian : 5/20
Estimasi waktu personal meet & greet adalah 4 menit.
Silahkan tunggu 16 menit lagi untuk tiba giliran Anda.

Terima kasih.

press any key to continue...
```

Ketentuan:

Nilai estimasi berdasarkan angka digit terakhir NIM Anda.

Contoh:

NIM: 21102104

Estimasi waktu personal meet & greet adalah 4 menit.

Jika digit terakhir adalah 0, maka gunakan angka sebelumnya selain nol.

Contoh:

NIM: 21102500

Estimasi waktu personal meet & greet adalah 5 menit.

Gunakan rumus untuk menghitung waktu tunggu.

~ SELAMAT MENGERJAKAN 😊 ~