LAPORAN PRAKTIKUM

MODUL 7
QUEUE

Disusun Oleh:
Riyon Aryono **: 2211102241**

Dosen

Muhamad Azrino Gustalika

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO

2023

BAB I
TUJUAN PRAKTIKUM

A. Tujuan Praktikum

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Queue
2. Mahasiswa mampu menerapkan Queue kedalam pemograman

BAB II
DASAR TEORI

1. **Pengertian Queue**

   Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep antrian pada kehidupan sehari-hari.

2. **Implementasi**

   Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. front adalah pointer ke elemen pertama dalam queue dan rear adalah pointer ke elemen terakhir dalam queue.

3. **Fungsi Queue**

   Pada Queue, karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut Enqueue dan Dequeue pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

4. **Operasi Pada Queue**

   enqueue(): menambahkan data ke dalam queue. dequeue(): mengeluarkan data dari queue. peek(): mengambil data dari queue tanpa menghapusnya. isEmpty(): mengecek apakah queue kosong atau tidak. isFull(): mengecek apakah queue penuh atau tidak. size(): menghitung jumlah elemen dalam queue.

BAB III
LATIHAN & TUGAS

1. Guided

- Demo Stack

Source Code

```
#include <iostream>
using namespace std;
// queue array
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
   if (back == maksimalQueue)
   {
      return true; //=1
   }
   else
   {
      return false;
   }
}
// fungsi pengecekan
bool isEmpty()
{ // antriannya kosong atau tidak
   if (back == 0)
   {
      return true;
   }
   else
   {
      return false;
   }
}
// fungsi menambahkan antrian
void enqueueAntrian(string data)
{
   if (isFull())
   {
      cout << "antrian penuh" << endl;
   }
   else
   { // nested if, nested for
```

```
        if (isEmpty())
        { // kondisi ketika queue kosong
            queueTeller[0] = data;
            front++; // front = front +1;
            back++;
        }
        else
        {                           // antrianya ada isi
            queueTeller[back] = data; // queueTeller[1]=data
            back++;                 // back=back+1; 2
        }
    }
}
// fungsi mengurangi antrian
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}
// fungsi menghitung banyak antrian

int countQueue()
{
    return back;
}
// fungsi menghapus semua antrian
void clearQueue()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
```

```
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}
// fungsi melihat antrian
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;
}
```

Output

```
data antrian teller :
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
jumlah antrian = 2
```

Penjelesan

Kode di atas adalah implementasi konsep antrian (queue) menggunakan array pada C++. Kode tersebut memiliki beberapa fungsi untuk melakukan operasi pada antrian, seperti menambahkan elemen (enqueue), menghapus elemen (dequeue), menghitung jumlah elemen dalam antrian (countQueue), menghapus semua elemen dalam antrian (clearQueue), dan melihat elemen-elemen dalam antrian (viewQueue).

2. Unguided

- Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list, lalu buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

  Source Code

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

struct Node {
   string nama;
   string nim;
   Node* next;
};

Node* front = NULL;
Node* back = NULL;

bool isEmpty() {
   return front == NULL;
}

void enqueueAntrian(string nama, string nim) {
   Node* newNode = new Node;
   newNode->nama = nama;
   newNode->nim = nim;
   newNode->next = NULL;

   if (isEmpty()) {
      front = newNode;
      back = newNode;
   } else {
```

```
        back->next = newNode;
        back = newNode;
    }
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
    }
}

int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    return count;
}

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

void viewQueue() {
    if (isEmpty()) {
        cout << "antrian kosong" << endl;
    } else {
        Node* temp = front;
        int index = 1;
        while (temp != NULL) {
            cout << index << ". " << temp->nama << " NIM: " << temp->nim
   << endl;
            temp = temp->next;
            index++;
        }
    }
}
```

```
int main() {
    int menu;
    string nama, nim;
    do{
        cout << "1.Masukkan Data"<<endl;
        cout << "2.Hapus Satu Data"<<endl;
        cout << "3.Reset Data"<<endl;
        cout << "4.Tampil Data"<<endl;
        cout << "Pilih Menu [1-4]: ";
        cin >> menu;
        switch (menu)
        {
        case 1:
            cout << "Masukkan Nama Mahasiswa: ";
            cin  >> nama;
            cout << "Masukkan NIM Mahasiswa: ";
            cin  >> nim;
            enqueueAntrian(nama,nim);
            break;
        case 2:
            dequeueAntrian();
            break;
        case 3:
            clearQueue();
            break;
        case 4:
            viewQueue();
            break;
        case 0:
            cout << "Selamat Tinggal" <<endl;
            break;
        default:
            cout << "Pilihan tidak ada" <<endl;
            break;
        }
        cout <<endl;
    }while (menu != 0);


}
```

Output

Tambah Data

```
1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 1
Masukkan Nama Mahasiswa: Riyon
Masukkan NIM Mahasiswa: 22102341
```

Tampil Data

```
1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 4
1. Riyon NIM: 22102341
2. Fitri NIM: 123123
```

Hapus Satu Data

```
1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 2

1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 4
1. Fitri NIM: 123123
```

Reset Data

```
1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 3

1.Masukkan Data
2.Hapus Satu Data
3.Reset Data
4.Tampil Data
Pilih Menu [1-4]: 4
antrian kosong
```

Penjelesan

Pada implementasi di atas, konsep antrian menggunakan linked list dengan menggunakan struct Node yang memiliki atribut nama dan nim. Setiap elemen dalam antrian direpresentasikan oleh Node dan memiliki pointer next yang menunjuk ke elemen berikutnya.

BAB IV
KESIMPULAN

Queue adalah struktur data yang mengikuti konsep FIFO (First-In-First-Out), di mana elemen yang pertama masuk ke dalam antrian akan menjadi elemen pertama yang keluar dari antrian. Operasi yang umum dilakukan pada antrian adalah enqueue (menambahkan elemen ke antrian), dequeue (menghapus elemen dari antrian), isEmpty (memeriksa apakah antrian kosong), isFull (memeriksa apakah antrian penuh), countQueue (menghitung jumlah elemen dalam antrian), dan viewQueue (melihat elemen-elemen dalam antrian).