LAPORAN PRAKTIKUM
Modul 08
“QUEUE”

Disusun Oleh:
Aji Prasetyo Nugroho - 2211104049
S1SE-07-2

Assisten Praktikum :
Aldi Putra
Andini Nur Hidayah

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

A. Tujuan

1. Mahasiswa mampu menjelaskan definisi dan konsep dari queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

B. Landasan Teori

Struktur data *Queue* adalah salah satu jenis struktur data yang digunakan untuk menyimpan elemen-elemen secara terurut dengan prinsip *First In, First Out* (FIFO). Prinsip FIFO ini mengatur bahwa elemen pertama yang dimasukkan ke dalam antrian akan menjadi elemen pertama yang dikeluarkan. Queue dapat diibaratkan sebagai antrian di kehidupan sehari-hari, seperti antrian di kasir, di mana orang yang datang pertama kali akan dilayani terlebih dahulu.

1. Prinsip FIFO (First In, First Out)

   Queue mengikuti prinsip FIFO, yang artinya elemen yang pertama kali dimasukkan ke dalam queue akan menjadi elemen pertama yang dikeluarkan. Dengan demikian, elemen-elemen dalam queue diatur sesuai urutan kedatangan mereka, dan hanya dapat diakses secara terurut mulai dari elemen yang paling depan (front). Sebagai contoh, dalam antrian kasir, orang pertama yang datang akan dilayani terlebih dahulu, dan orang yang datang berikutnya akan dilayani setelahnya, mengikuti urutan kedatangan.

2. Perbedaan dengan Stack

   Queue berbeda dengan struktur data *Stack*, yang mengikuti prinsip *Last In, First Out* (LIFO). Dalam *Stack*, elemen yang terakhir dimasukkan akan menjadi yang pertama dikeluarkan. Proses ini bisa diibaratkan dengan tumpukan piring, di mana piring yang terakhir dimasukkan ke dalam tumpukan akan diambil terlebih dahulu.

   Berbeda dengan *Stack*, dalam Queue, elemen ditambahkan di ujung belakang (rear/tail) dan dihapus dari ujung depan (front/head). Hal ini memungkinkan Queue untuk mengelola elemen-elemen dengan cara yang lebih teratur sesuai dengan prinsip FIFO.

3. Operasi pada Queue

Queue memiliki berbagai operasi dasar yang digunakan untuk mengelola elemen-elemen dalam antrian. Beberapa operasi tersebut adalah:

- Enqueue(): Operasi untuk menambahkan elemen baru ke dalam queue. Elemen baru ditambahkan di belakang queue, yaitu setelah elemen terakhir yang ada.
- Dequeue(): Operasi untuk mengeluarkan elemen dari queue. Elemen yang paling depan (head) akan dihapus, dan posisi head akan bergeser ke elemen berikutnya.
- Peek(): Operasi untuk melihat elemen pertama dalam queue tanpa menghapusnya. Fungsi ini hanya memberikan informasi mengenai elemen yang ada di depan queue.
- isEmpty(): Operasi untuk mengecek apakah queue kosong atau tidak. Jika queue kosong, maka tidak ada elemen yang dapat diambil.
- isFull(): Operasi untuk mengecek apakah queue sudah penuh atau tidak, tergantung pada kapasitas queue yang tersedia.
- Size(): Operasi untuk menghitung jumlah elemen yang ada dalam queue pada saat tertentu.

4. Implementasi Queue

Queue dapat diimplementasikan menggunakan berbagai struktur data dasar seperti *array* atau *linked list*. Dalam implementasi menggunakan *array*, queue dapat dibangun dengan mendefinisikan dua pointer yaitu *front* dan *rear*. Pointer *front* menunjukkan elemen pertama dalam queue, sementara pointer *rear* menunjukkan elemen terakhir.

Pada implementasi linked list, elemen queue dihubungkan satu sama lain menggunakan pointer, dan operasi *enqueue* serta *dequeue* dilakukan dengan memanipulasi node-node pada list tersebut.

5. Aplikasi Queue dalam Kehidupan Sehari-Hari

Queue memiliki berbagai aplikasi dalam kehidupan sehari-hari, terutama dalam pengelolaan antrian yang melibatkan urutan kedatangan. Beberapa contoh penerapan queue adalah:

- Antrian di Kasir: Pembeli yang datang pertama kali akan dilayani terlebih dahulu.
- Proses Penjadwalan di Sistem Operasi: Antrian proses yang siap untuk dieksekusi oleh CPU.
- Proses Pengiriman Data pada Jaringan Komputer: Paket data yang dikirimkan melalui jaringan diurutkan berdasarkan waktu pengirimannya.

C. Guided

1. Guided 1

   Source Code :

```cpp
#include <iostream>
#define MAX 100

using namespace std;

class Queue {
private:
    int front, rear;
    int arr[MAX];
public:

    Queue() {
        front = -1;
        rear = -1;
    }


    bool isFull() {
        return rear == MAX - 1;
    }


    bool isEmpty() {
        return front == -1 || front > rear;
    }


    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue Overflow\n";
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
    }


    void dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow\n";
            return;
        }
        front++;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[front];
        }
        cout << "Queue is empty\n";
        return -1;
    }


    void display() {
        if (isEmpty()) {
            cout << "Queue is empty\n";
            return;
        }
        for (int i = front; i <= rear; i++) {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};


int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Queue elements: ";
    q.display();

    cout << "Front element: " << q.peek() << "\n";

    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}
```

Output :

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS D:\Praktikum STD_2211104049>
```

2. Guided 2

Source Code :

```cpp
#include <iostream>

using namespace std;

// Node untuk setiap elemen Queue
class Node {
public:
    int data;      // Data elemen
    Node* next;    // Pointer ke node berikutnya

    // Konstruktor untuk Node
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

// Kelas Queue menggunakan linked list
class Queue {
private:
    Node* front; // Pointer ke elemen depan Queue
    Node* rear;  // Pointer ke elemen belakang Queue

public:
    // Konstruktor Queue
    Queue() {
        front = rear = nullptr;
    }

    // Mengecek apakah Queue kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Menambahkan elemen ke Queue
    void enqueue(int x) {
        Node* newNode = new Node(x);
        if (isEmpty()) {
            front = rear = newNode; // Jika Queue kosong
            return;
        }
        rear->next = newNode; // Tambahkan node baru ke belakang
        rear = newNode;       // Perbarui rear
    }

    // Menghapus elemen dari depan Queue
    void dequeue() {
        if (isEmpty()) {
           cout << "Queue Underflow\n";
            return;
        }
        Node* temp = front;      // Simpan node depan untuk dihapus
        front = front->next;     // Pindahkan front ke node berikutnya
        delete temp;             // Hapus node lama
        if (front == nullptr)    // Jika Queue kosong, rear juga harus null
            rear = nullptr;
    }

    // Mengembalikan elemen depan Queue tanpa menghapusnya
    int peek() {
        if (!isEmpty()) {
            return front->data;
        }
       cout << "Queue is empty\n";
        return -1; // Nilai sentinel
    }

    // Menampilkan semua elemen di Queue
    void display() {
        if (isEmpty()) {
           cout << "Queue is empty\n";
            return;
        }
        Node* current = front; // Mulai dari depan
        while (current) {      // Iterasi sampai akhir
           cout << current->data << " ";
            current = current->next;
        }
       cout << "\n";
    }
};

// Fungsi utama untuk menguji Queue
int main() {
    Queue q;

    // Menambahkan elemen ke Queue
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    // Menampilkan elemen di Queue
   cout << "Queue elements: ";
    q.display();

    // Menampilkan elemen depan
   cout << "Front element: " << q.peek() << "\n";

    // Menghapus elemen dari depan Queue
    q.dequeue();
   cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}
```

Output :

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS D:\Praktikum STD_2211104049>
```

3. Guided 3

Source Code :

```
#include<iostream>

using namespace std;

const 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;
}
}

bool isEmpty() { // Antriannya kosong atau tidak
if (back == 0) { return true;
} else {
return false;
}
}

void enqueueAntrian(string data) { // Fungsi menambahkan antrian
if (isFull()) {
cout << "Antrian penuh" << endl;
} else {
if (isEmpty()) { // Kondisi ketika queue kosong
queueTeller[0] = data; front++;
back++;
} else { // Antrianya ada isi queueTeller[back] = data; back++;
}
}
}

void dequeueAntrian() { // Fungsi mengurangi antrian
if (isEmpty()) {
cout << "Antrian kosong" << endl;
} else {
for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
}
back--;
}
}

int countQueue() { // Fungsi menghitung banyak antrian
return back;
}

void clearQueue() { // Fungsi menghapus semua antrian
if (isEmpty()) {
cout << "Antrian kosong" << endl;
} else {
for (int i = 0; i < back; i++) { queueTeller[i] = "";
}
back = 0;
front = 0;
}
}

void viewQueue() { // Fungsi melihat antrian
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. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
Antrian kosong
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS D:\Praktikum STD_2211104049>
```

D. Unguided

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa
3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

   Noted : Untuk data mahasiswa dan nim dimasukan oleh user

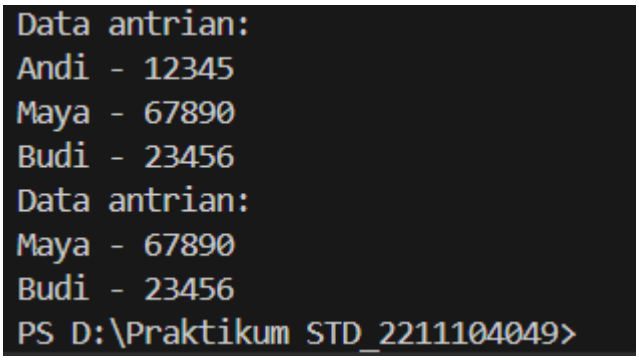
1. Soal 1

   main.cpp

   Source Code :

```cpp
#include<iostream>

using namespace std;

// Struktur Node untuk Linked List
struct Node {
    string nama;
    string nim;
    Node* next;
};

// Struktur Queue
struct Queue {
    Node* front;
    Node* back;

    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

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

        if (isEmpty()) {
            front = back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

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

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
        } else {
            Node* current = front;
            cout << "Data antrian: " << endl;
            while (current != nullptr) {
                cout << current->nama << " - " << current->nim << endl;
                current = current->next;
            }
        }
    }
};

int main() {
    Queue queue;
    queue.enqueue("Andi", "12345");
    queue.enqueue("Maya", "67890");
    queue.enqueue("Budi", "23456");

    queue.viewQueue();
    queue.dequeue();
    queue.viewQueue();

    return 0;
}
```

Output :


Data antrian:
Andi - 12345
Maya - 67890
Budi - 23456
Data antrian:
Maya - 67890
Budi - 23456
PS D:\Praktikum STD_2211104049>

2. Soal 2

main.cpp

Source Code :

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

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

struct Queue {
    Node* front;
    Node* back;

    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

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

        if (isEmpty()) {
            front = back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

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

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
        } else {
            Node* current = front;
            cout << "Data antrian: " << endl;
            while (current != nullptr) {
                cout << current->nama << " - " << current->nim << endl;
                current = current->next;
            }
        }
    }
};

int main() {
    Queue queue;
    string nama, nim;

    cout << "Masukkan data mahasiswa untuk antrian:" << endl;

    int jumlahMahasiswa;
    cout << "Berapa jumlah mahasiswa yang ingin dimasukkan ke dalam antrian? ";
    cin >> jumlahMahasiswa;

    for (int i = 0; i < jumlahMahasiswa; i++) {
        cout << "\nMasukkan nama mahasiswa: ";
        cin.ignore(); // Untuk membersihkan buffer input
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;
        queue.enqueue(nama, nim);
    }

    queue.viewQueue();

    return 0;
}
```

Output :

```
Menu Antrian:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Aji
Masukkan NIM mahasiswa: 2211104049
Data antrian Aji dengan NIM 2211104049 berhasil ditambahkan.

Menu Antrian:
1. Tambah Antrian
2. Hapus Antrian
3. Lihat Antrian
4. Jumlah Antrian
5. Hapus Semua Antrian
6. Keluar
Pilih menu: 6
Keluar dari program...
PS D:\Praktikum STD_2211104049>
```

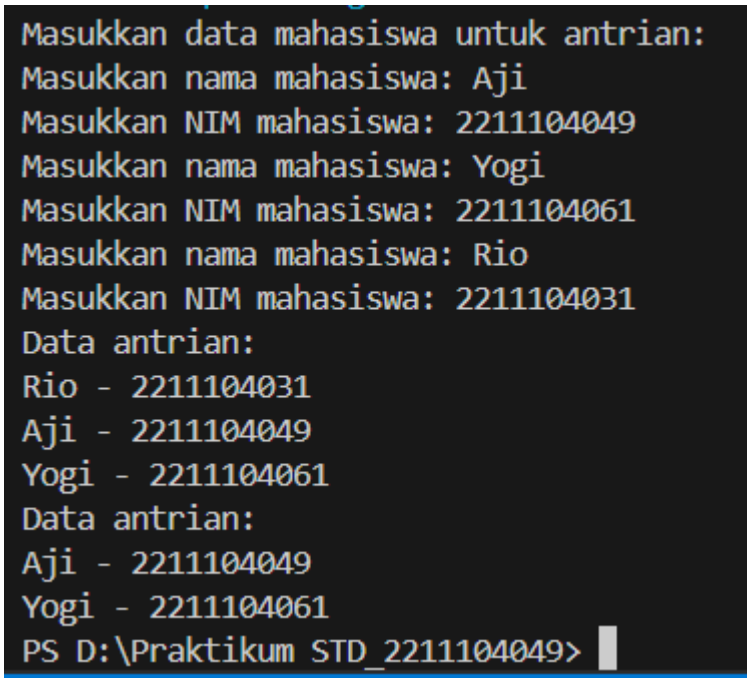
3. Soal 3

Main.cpp

Source Code :

```cpp
#include<iostream>
#include<string>

using namespace std;

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

struct Queue {
    Node* front;
    Node* back;

    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

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

        // Jika queue kosong
        if (isEmpty()) {
            front = back = newNode;
        } else {
            // Jika NIM mahasiswa baru lebih kecil dari NIM yang ada di front
            if (newNode->nim < front->nim) {
                newNode->next = front;
                front = newNode;
            } else {
                Node* current = front;
                while (current->next != nullptr && current->next->nim < newNode->nim) {
                    current = current->next;
                }
                newNode->next = current->next;
                current->next = newNode;
                if (newNode->next == nullptr) {
                    back = newNode;
                }
            }
        }
    }

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

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
        } else {
            Node* current = front;
            cout << "Data antrian: " << endl;
            while (current != nullptr) {
                cout << current->nama << " - " << current->nim << endl;
                current = current->next;
            }
        }
    }
};

int main() {
    Queue queue;
    string nama, nim;

    cout << "Masukkan data mahasiswa untuk antrian:" << endl;

    for (int i = 0; i < 3; i++) {
        cout << "Masukkan nama mahasiswa: ";
        cin >> nama;
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;
        queue.enqueue(nama, nim);
    }

    queue.viewQueue();
    queue.dequeue();
    queue.viewQueue();

    return 0;
}
```

Output :


Masukkan data mahasiswa untuk antrian:
Masukkan nama mahasiswa: Aji
Masukkan NIM mahasiswa: 2211104049
Masukkan nama mahasiswa: Yogi
Masukkan NIM mahasiswa: 2211104061
Masukkan nama mahasiswa: Rio
Masukkan NIM mahasiswa: 2211104031
Data antrian:
Rio - 2211104031
Aji - 2211104049
Yogi - 2211104061
Data antrian:
Aji - 2211104049
Yogi - 2211104061
PS D:\Praktikum STD_2211104049>