Laporan Praktikum
MODUL VIII
QUEUE

Disusun Oleh :
Dwi Candra Pratama/2211104035 SE 07 02

Asisten Praktikum :
Aldi Putra Andini Nur Hidayah

Dosen Pengampu :
Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA TELKOM UNIVERSITY PURWOKERTO
2024

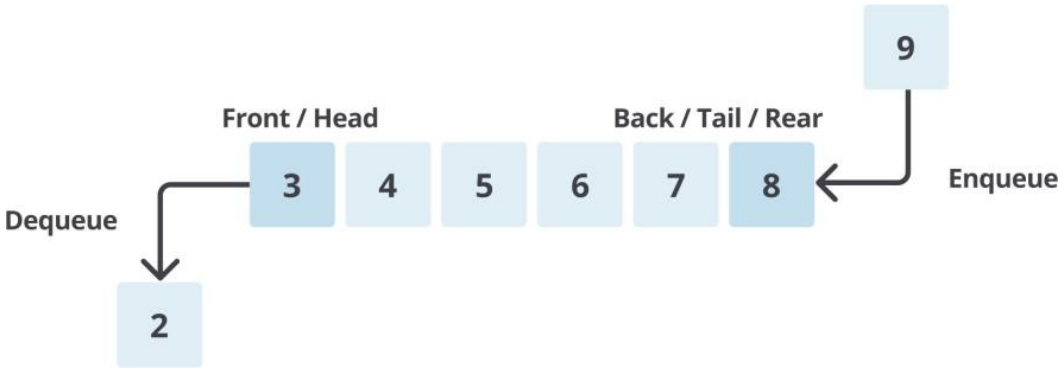
A. TUJUAN PRAKTIKUM

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. DASAR TEORI

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, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

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



FIRST IN FIRST OUT (FIFO)

Perbedaan antara *stack* dan *queue* terletak pada aturan penambahan dan penghapusan elemen. Pada *stack*, operasi penambahan dan penghapusan elemen dilakukan di satu ujung yang disebut *top* (ujung atas). Elemen yang terakhir kali dimasukkan ke dalam *stack* akan berada di posisi paling atas dan akan menjadi elemen pertama yang dihapus. Sifat ini dikenal dengan istilah *LIFO* (Last In, First Out). Contoh analogi sederhana dari *stack* adalah tumpukan piring, di mana piring terakhir yang ditambahkan berada di posisi paling atas dan akan diambil atau dihapus terlebih dahulu.

Sebaliknya, pada *queue*, operasi penambahan dan penghapusan elemen dilakukan di dua ujung yang berbeda. Elemen baru ditambahkan di ujung belakang (*rear* atau *tail*), dan elemen dihapus dari ujung depan (*front* atau *head*). Proses ini mengikuti prinsip *FIFO* (First In, First Out), yang berarti elemen pertama yang dimasukkan ke dalam *queue* akan menjadi elemen pertama yang dikeluarkan. Dalam konteks *queue*, operasi

penambahan elemen dikenal sebagai ***Enqueue***, dan operasi penghapusan elemen disebut ***Dequeue***.

Pada *Enqueue*, elemen ditambahkan di belakang *queue* setelah elemen terakhir yang ada, sementara pada *Dequeue*, elemen paling depan (*head*) dihapus, dan posisi *head* akan bergeser ke elemen berikutnya. Contoh penggunaan *queue* dalam kehidupan sehari-hari adalah antrean di kasir, di mana orang pertama yang datang adalah yang pertama dilayani.

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.

C. GUIDED

Guided 1

```
#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;
}
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8> cd 'd:\Semester5\StrukturData\Pr
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output> & .\'Guided1.exe'
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output>
```

Guided 2

```
#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;
}
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8> cd 'd:\Semester5\StrukturData\P
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output> & .\'Guided2.exe'
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output>
```

Guided 3

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

using namespace std;

const int MAKSIMAL_QUEUE = 5; // Maksimal antrian
int front = 0;              // Penanda antrian depan
int back = 0;               // Penanda antrian belakang
string queueTeller[MAKSIMAL_QUEUE]; // Array untuk menyimpan antrian

// Cek apakah antrian penuh
bool isFull() {
   return back == MAKSIMAL_QUEUE;
}

// Cek apakah antrian kosong
bool isEmpty() {
   return back == 0;
}

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

// Fungsi mengurangi antrian
void dequeueAntrian() {
   if (isEmpty()) {
      cout << "Antrian kosong" << endl;
   } else {
      // Geser elemen ke depan
```

```
        for (int i = 0; i < back - 1; 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 {
        // Reset semua elemen
        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 < MAKSIMAL_QUEUE; i++) {
        if (queueTeller[i] != "") {
            cout << i + 1 << ". " << queueTeller[i] << endl;
        } else {
            cout << i + 1 << ". (kosong)" << endl;
        }
    }
}

int main() {
    // Contoh penggunaan fungsi-fungsi queue
    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;
}
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8> cd 'd:\Semester5\StrukturData\Pr
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output> & .\'Guided3.exe'
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\Guided\output>
```

D. UNGUIDED

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

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

// Node untuk linked list
struct Node {
    string data;
    Node* next;
};

// Struktur Queue
struct Queue {
    Node* front; // Elemen pertama
    Node* back;  // Elemen terakhir

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

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

    // Menambahkan elemen ke queue
    void enqueue(string data) {
        Node* newNode = new Node{data, nullptr};
        if (isEmpty()) {
            front = back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

    // Menghapus elemen dari queue
    void dequeue() {
```

```
        if (isEmpty()) {
            cout << "Queue kosong\n";
            return;
        }
        Node* temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr) back = nullptr;
    }

    // Menampilkan elemen dalam queue
    void display() {
        if (isEmpty()) {
            cout << "Queue kosong\n";
            return;
        }
        Node* temp = front;
        cout << "Isi queue: ";
        while (temp) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    Queue q;
    int jumlah;

    cout << "Masukkan jumlah data: ";
    cin >> jumlah;

    for (int i = 1; i <= jumlah; i++) {
        string data;
        cout << "Masukkan data ke-" << i << ": ";
        cin >> data;
        q.enqueue(data);
    }

    cout << "\nData dalam queue setelah input:\n";
    q.display();

    cout << "\nProses dequeue...\n";
```

```
        q.dequeue();
        q.display();

        return 0;
}
```

```
an 8\UnGuided\output'
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output> & .\'Soal1.exe'
Masukkan jumlah data: 3
Masukkan data ke-1: Dirmo
Masukkan data ke-2: Rani
Masukkan data ke-3: Aris

Data dalam queue setelah input:
Isi queue: Dirmo Rani Aris

Proses dequeue...
Isi queue: Rani Aris
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output>
```

2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

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

// Node untuk linked list
struct Node {
    string nama;
    string nim;
    Node* next;
};

// Struktur Queue untuk mahasiswa
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{nama, nim, nullptr};
        if (isEmpty()) {
            front = back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue kosong" << endl;
            return;
        }
        Node* temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr) back = nullptr;
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Data mahasiswa dalam queue:" << endl;
        while (temp) {
            cout << "Nama: " << temp->nama << ", NIM: " << temp->nim
     << endl;
            temp = temp->next;
        }
    }
};

int main() {
    Queue q;
    int n;
    string nama, nim;

    cout << "Masukkan jumlah mahasiswa: ";
    cin >> n;
    cin.ignore();  // Menghapus karakter newline setelah input integer
```

```
        for (int i = 0; i < n; i++) {
            cout << "Masukkan Nama Mahasiswa: ";
            getline(cin, nama);
            cout << "Masukkan NIM Mahasiswa: ";
            getline(cin, nim);
            q.enqueue(nama, nim);
        }

        cout << "\nData mahasiswa dalam queue:" << endl;
        q.display();

        cout << "\nMenghapus elemen dari queue..." << endl;
        q.dequeue();
        q.display();

        return 0;
}
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8> cd 'd:\Semester5\StrukturData\P
an 8\UnGuided\output'
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output> & .\'Soal2.exe'
Masukkan jumlah mahasiswa: 4
Masukkan Nama Mahasiswa: Sigit
Masukkan NIM Mahasiswa: 221110105
Masukkan Nama Mahasiswa: Adoeng
Masukkan NIM Mahasiswa: 221110103
Masukkan Nama Mahasiswa: Ciung
Masukkan NIM Mahasiswa: 221110102
Masukkan Nama Mahasiswa: Faix
Masukkan NIM Mahasiswa: 221110101

Data mahasiswa dalam queue:
Data mahasiswa dalam queue:
Nama: Sigit, NIM: 221110105
Nama: Adoeng, NIM: 221110103
Nama: Ciung, NIM: 221110102
Nama: Faix, NIM: 221110101

Menghapus elemen dari queue...
Data mahasiswa dalam queue:
Nama: Adoeng, NIM: 221110103
Nama: Ciung, NIM: 221110102
Nama: Faix, NIM: 221110101
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output>
```

3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

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

// Node untuk linked list
struct Node {
   string nama;
   string nim;
   Node* next;
};

// Struktur Queue dengan prioritas berdasarkan NIM
struct PriorityQueue {
   Node* front;

   PriorityQueue() {
      front = nullptr;
   }

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

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

      if (isEmpty() || nim < front->nim) {
         newNode->next = front;
         front = newNode;
      } else {
         Node* temp = front;
         while (temp->next != nullptr && temp->next->nim < nim) {
            temp = temp->next;
         }
         newNode->next = temp->next;
         temp->next = newNode;
      }
   }

   void dequeue() {
      if (isEmpty()) {
         cout << "Queue kosong" << endl;
```

```
            return;
        }
        Node* temp = front;
        front = front->next;
        delete temp;
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Data mahasiswa dalam queue:" << endl;
        while (temp) {
            cout << "Nama: " << temp->nama << ", NIM: " << temp->nim
     << endl;
            temp = temp->next;
        }
    }
};

int main() {
    PriorityQueue pq;
    int n;
    string nama, nim;

    cout << "Masukkan jumlah mahasiswa: ";
    cin >> n;
    cin.ignore();

    for (int i = 0; i < n; i++) {
        cout << "Masukkan Nama Mahasiswa: ";
        getline(cin, nama);
        cout << "Masukkan NIM Mahasiswa: ";
        getline(cin, nim);
        pq.enqueue(nama, nim);
    }

    cout << "\nData mahasiswa setelah diurutkan berdasarkan NIM:" <<
     endl;
    pq.display();

    return 0;
```

```
}
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8> cd 'd:\Semester5\
an 8\UnGuided\output'
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output> &
Masukkan jumlah mahasiswa: 4
Masukkan Nama Mahasiswa: Hanif
Masukkan NIM Mahasiswa: 2211101003
Masukkan Nama Mahasiswa: Jasmin
Masukkan NIM Mahasiswa: 2211101001
Masukkan Nama Mahasiswa: Mali
Masukkan NIM Mahasiswa: 2211101004
Masukkan Nama Mahasiswa: Ariska
Masukkan NIM Mahasiswa: 2211101002

Data mahasiswa setelah diurutkan berdasarkan NIM:
Data mahasiswa dalam queue:
Nama: Jasmin, NIM: 2211101001
Nama: Ariska, NIM: 2211101002
Nama: Hanif, NIM: 2211101003
Nama: Mali, NIM: 2211101004
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 8\UnGuided\output>
```

Noted : Untuk data mahasiswa dan nim dimasukan oleh user