Telkom University Purwokerto

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHubyang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. AsistenPraktikum:4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folderdalam repositoryakan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwalpraktikumadalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktuini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem- solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

LAPORAN PRAKTIKUM
MODUL 8
QUEUE

Disusun Oleh :

Zaenarif Putra ‘Ainurdin – 2311104049

Kelas :

SE-07-02

Dosen :

Wahyu Andi Saputra, S.pd,M.Eng

PROGRAM STUDI SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

I. 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

II. LANDASAN TEORI

Queue adalah struktur data yang menggunakan prinsip FIFO, atau First-In First-Out, di mana elemen pertama yang dimasukkan akan dikeluarkan. Sisi depan menunjuk elemen pertama dan sisi belakang menunjuk elemen terakhir. Enqueue (menambahkan elemen di belakang), dequeue (menghapus elemen di depan), peek (mengambil elemen depan tanpa menghapusnya), isEmpty (mengecek apakah queue kosong), isFull (mengecek apakah queue penuh), dan size adalah operasi dasar yang dilakukan pada queue. Queue memungkinkan penambahan dan penghapusan elemen di ujung yang berbeda dari stack yang menggunakan metode LIFO (Last-In First-Out). Ini membuatnya ideal untuk aplikasi seperti sistem penjadwalan atau antrian pelanggan.

III. GUIDE

1. Guide1

   a. Syntax

```
#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;
```

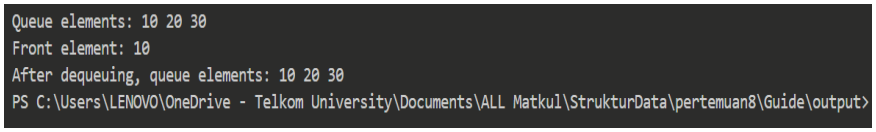
```
    }

    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;
}
```

b. Penjelasan syntax

- Kelas Queue memiliki dua variabel anggota front dan rear untuk melacak posisi elemen pertama dan terakhir dalam antrian, serta array arr untuk menyimpan elemen-elemen antrian.
- Konstruktor Queue diinisialisasi dengan front dan rear diatur ke -1, menandakan bahwa antrian kosong.
- isFull(): Mengembalikan true jika antrian sudah penuh (rear mencapai batas maksimum). isEmpty(): Mengembalikan true jika antrian kosong (front -1 atau front lebih besar dari rear).
- enqueue(int x): Menambahkan elemen x ke belakang antrian. Jika antrian penuh, menampilkan pesan "Queue Overflow". Jika antrian kosong, front diatur ke 0. Elemen ditambahkan ke array dan rear ditingkatkan.
- dequeue(): Menghapus elemen dari depan antrian. Jika antrian kosong, menampilkan pesan "Queue Underflow". Jika tidak, front ditingkatkan untuk menunjukkan elemen berikutnya.
- peek(): Mengembalikan elemen di depan antrian tanpa menghapusnya. Jika antrian kosong, menampilkan pesan dan mengembalikan -1.
- display(): Menampilkan semua elemen dalam antrian dari front hingga rear. Jika antrian kosong, menampilkan pesan.
- Di dalam fungsi main, objek Queue dibuat. Tiga elemen (10, 20, 30) ditambahkan ke antrian menggunakan enqueue. Kemudian, elemen antrian ditampilkan, elemen di depan ditampilkan menggunakan peek, dan setelah itu, elemen di depan dihapus menggunakan dequeue. Namun, tidak ada panggilan dequeue di sini, jadi tampaknya ada kesalahan dalam pernyataan "setelah dequeuing" karena tidak ada elemen yang dihapus.

c. Output



2. Guide2

a. Syntax :

```
#include <iostream>

using namespace std;

class Node {
public:
    int data;
    Node* next;

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

class Queue {
private:
    Node* front;
    Node* rear;

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

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

    void enqueue(int x) {
        Node* newNode = new Node(x);
        if (isEmpty()) {
            front = rear = newNode;
            return;
        }
        rear->next = newNode;
        rear = newNode;
    }

    void dequeue() {
        if (isEmpty()) {
           cout << "Queue Underflow\n";
            return;
        }
        Node* temp = front;
        front = front->next;
```

```
        delete temp;
        if (front == nullptr)
            rear = nullptr;
    }

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

    void display() {
        if (isEmpty()) {
           cout << "Queue is empty\n";
            return;
        }
        Node* current = front;
        while (current) {
           cout << current->data << " ";
            current = current->next;
        }
       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";

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

    return 0;
}
```

b. Penjelasan Syntax :

- Kelas Node merepresentasikan elemen dalam antrian. Memiliki dua atribut: data: Menyimpan nilai dari elemen antrian. next: Pointer yang

menunjuk ke node berikutnya dalam antrian. Konstruktor Node(int value) digunakan untuk menginisialisasi data dengan nilai yang diberikan dan next diatur ke nullptr (tidak ada node berikutnya).

- Kelas Queue merepresentasikan antrian itu sendiri. Memiliki dua pointer: front: Menunjuk ke elemen pertama dalam antrian. rear: Menunjuk ke elemen terakhir dalam antrian. Konstruktor Queue() menginisialisasi front dan rear ke nullptr, menandakan bahwa antrian kosong.
- Memeriksa apakah antrian kosong dengan mengembalikan true jika front adalah nullptr.
- enqueue(int x): Menambahkan elemen baru ke belakang antrian. Membuat node baru (newNode) dengan nilai x. Jika antrian kosong, front dan rear diatur ke node baru. Jika tidak kosong, node baru ditambahkan di belakang antrian dengan mengubah pointer next dari rear, kemudian memperbarui rear ke node baru.
- dequeue(): Menghapus elemen dari depan antrian. Jika antrian kosong, menampilkan pesan "Queue Underflow". Jika tidak kosong, menyimpan pointer ke node front dalam temp, kemudian memperbarui front ke node berikutnya. Node yang dihapus (temp) dibebaskan dari memori. Jika setelah penghapusan front menjadi nullptr, maka rear juga diatur ke nullptr.
- peek(): Mengembalikan nilai dari elemen di depan antrian tanpa menghapusnya. Jika antrian tidak kosong, mengembalikan nilai data dari front. Jika kosong, menampilkan pesan dan mengembalikan -1.
- display(): Menampilkan semua elemen dalam antrian dari front hingga rear. Jika antrian kosong, menampilkan pesan. Menggunakan pointer current untuk iterasi melalui node-node dan mencetak nilai data masing-masing.
- Fungsi main() adalah titik awal program. Membuat objek Queue bernama q. Menambahkan elemen 10, 20, dan 30 ke dalam antrian menggunakan enqueue. Menampilkan elemen antrian saat ini dengan memanggil display(). Mengambil elemen di depan antrian menggunakan peek() dan menampilkannya. Menghapus elemen terdepan dengan dequeue() dan menampilkan elemen antrian setelah penghapusan.

c. Output :

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan8\Guide\output>
```

3. Guide3

a. Syntax

```
#include<iostream>

using namespace std;
```

```
const int maksimalQueue = 5;
int front = 0;
int back = 0;
string queueTeller[5];

bool isFull() {
if (back == maksimalQueue) { return true;
} else {
return false;
}
}

bool isEmpty() {
if (back == 0) { return true;
} else {
return false;
}
}

void enqueueAntrian(string data) {
if (isFull()) {
cout << "Antrian penuh" << endl;
} else {
if (isEmpty()) {
queueTeller[0] = data; front++;
back++;
} else {
}
}
}

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

int countQueue() {
return back;
}

void clearQueue() {
if (isEmpty()) {
cout << "Antrian kosong" << endl;
} else {
for (int i = 0; i < back; i++) { queueTeller[i] = "";
```

```
}
back = 0;
front = 0;
}
}

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("Gonzales");

enqueueAntrian("Maradona");

viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

clearQueue();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

return 0;
}
```

b. Penjelasan Syntax :

- maksimalQueue: Menentukan kapasitas maksimum dari antrian, yaitu 5. front: Menunjukkan posisi depan antrian. back: Menunjukkan posisi belakang antrian. queueTeller: Array yang menyimpan elemen-elemen antrian.
- isFull(): Mengembalikan true jika antrian sudah penuh (jumlah elemen sama dengan kapasitas maksimum). isEmpty(): Mengembalikan true jika antrian kosong (tidak ada elemen).

- Fungsi enqueueAntrian digunakan untuk menambahkan elemen ke dalam antrian. Jika antrian penuh, akan menampilkan pesan "Antrian penuh". Jika antrian kosong, elemen pertama akan ditambahkan ke posisi 0 dan front serta back akan diincrement. Namun, bagian untuk menambahkan elemen jika antrian tidak kosong belum lengkap.
- Fungsi dequeueAntrian digunakan untuk menghapus elemen dari depan antrian. Jika antrian kosong, akan menampilkan pesan "Antrian kosong".Jika tidak kosong, elemen di depan dihapus dan semua elemen lainnya digeser ke depan satu posisi.
- Fungsi countQueue digunakan untuk mengembalikan jumlah elemen saat ini dalam antrian.
- Fungsi clearQueue Menghapus semua elemen dalam antrian. Jika antrian kosong, akan menampilkan pesan "Antrian kosong". Jika tidak kosong, semua elemen diatur menjadi string kosong, dan front serta back direset ke 0.
- Fungsi viewQueue Menampilkan semua elemen dalam antrian. Jika elemen ada, ditampilkan; jika tidak, ditampilkan pesan "(kosong)".
- Fungsi main adalah titik awal program.Dua elemen, "Gonzales" dan "Maradona", ditambahkan ke antrian menggunakan enqueueAntrian.
- Kemudian, antrian ditampilkan dengan viewQueue, dan jumlah elemen ditampilkan menggunakan countQueue. Elemen di depan antrian dihapus dengan dequeueAntrian, dan antrian ditampilkan lagi. Setelah itu, antrian dibersihkan dengan clearQueue, dan ditampilkan kembali untuk menunjukkan bahwa antrian sudah kosong.

c. Output

```
Data antrian teller:
1. Gonzales
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 C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan8\Guide\output>
```

IV. UNGUIDED

1. Unguided1

a. Syntax :

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

```
struct Node {
    string data;
    Node* next;
};

struct Queue {
    Node* front;
    Node* rear;
    int count;
};

Queue* createQueue() {
    Queue* queue = new Queue();
    queue->front = nullptr;
    queue->rear = nullptr;
    queue->count = 0;
    return queue;
}

bool isEmpty(Queue* queue) {
    return (queue->front == nullptr);
}

void enqueue(Queue* queue, string data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;

    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
    queue->count++;
    cout << data << " ditambahkan ke antrian." << endl;
}

void dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        cout << "Antrian kosong." << endl;
        return;
    }
    Node* temp = queue->front;
    cout << temp->data << " dikeluarkan dari antrian." << endl;
    queue->front = queue->front->next;
    delete temp;
    queue->count--;
```

```
    if (queue->front == nullptr) {
        queue->rear = nullptr;
        cout << "Antrian telah dikosongkan." << endl;
    }
}

void viewQueue(Queue* queue) {
    Node* current = queue->front;
    cout << "Data antrian :" << endl;
    int index = 1;
    while (current != nullptr) {
        cout << index++ << ". " << current->data << endl;
        current = current->next;
    }
    cout << "Jumlah antrian = " << queue->count << endl;
}

int main() {
    Queue* queue = createQueue();
    enqueue(queue, "Putra");
    enqueue(queue, "Zahara");
    enqueue(queue, "Izaaty");
    viewQueue(queue);
    dequeue(queue);
    viewQueue(queue);
    dequeue(queue);
    dequeue(queue);
    return 0;
}
```

b. Penjelasan Syntax

- Node: Struktur ini merepresentasikan elemen dalam antrian. Setiap Node memiliki dua atribut: data: Menyimpan data (dalam hal ini, bertipe string). next: Pointer yang menunjuk ke Node berikutnya dalam antrian.
- Queue: Struktur ini merepresentasikan antrian itu sendiri. Memiliki tiga atribut: front: Pointer yang menunjuk ke elemen terdepan dalam antrian. rear: Pointer yang menunjuk ke elemen terakhir dalam antrian. count: Menyimpan jumlah elemen saat ini dalam antrian.
- Fungsi createQueue digunakan untuk membuat antrian baru. Mengalokasikan memori untuk Queue, menginisialisasi front dan rear dengan nullptr (menunjukkan bahwa antrian kosong), dan mengatur count ke 0. Mengembalikan pointer ke antrian yang baru dibuat.
- Fungsi isEmpty digunakan untuk memeriksa apakah antrian kosong dengan memeriksa apakah front adalah nullptr. Mengembalikan true jika antrian kosong, dan false jika tidak.
- Fungsi enqueue digunakan untuk menambahkan elemen ke dalam antrian. Membuat Node baru dan mengisi data dengan nilai yang diberikan.

- Jika antrian kosong, front dan rear diatur ke newNode. Jika tidak kosong, next dari rear diatur ke newNode, dan rear diperbarui. count diincrement untuk menunjukkan jumlah elemen yang bertambah. Menampilkan pesan bahwa data telah ditambahkan.
- Fungsi dequeue digunakan untuk menghapus elemen dari depan antrian. Jika antrian kosong, menampilkan pesan "Antrian kosong".
- Jika tidak kosong, menyimpan front dalam variabel sementara (temp), menampilkan data yang akan dikeluarkan, dan memperbarui front ke elemen berikutnya. Menghapus temp (elemen yang dikeluarkan) dari memori dan mengurangi count. Jika setelah penghapusan antrian menjadi kosong, rear juga diatur ke nullptr dan menampilkan pesan bahwa antrian telah kosong.
- Fungsi viewQueue digunakan untuk menampilkan semua elemen dalam antrian. Menggunakan pointer current untuk melintasi setiap Node mulai dari front. Menampilkan data setiap elemen dengan nomor urut, dan melanjutkan ke Node berikutnya hingga mencapai akhir antrian (nullptr). Setelah menampilkan semua elemen, juga menampilkan jumlah elemen dalam antrian.
- Fungsi main adalah titik awal program. Membuat antrian baru dengan createQueue. Menambahkan tiga elemen ("Putra", "Zahara", "Izaaty") ke dalam antrian menggunakan enqueue. Menampilkan antrian saat ini dengan viewQueue. Menghapus elemen dari antrian dengan dequeue dan menampilkan antrian setelah setiap penghapusan. Program berakhir setelah semua operasi selesai.

c. Output

```
Putra ditambahkan ke antrian.
Zahara ditambahkan ke antrian.
Izaaty ditambahkan ke antrian.
Data antrian :
1. Putra
2. Zahara
3. Izaaty
Jumlah antrian = 3
Putra dikeluarkan dari antrian.
Data antrian :
1. Zahara
2. Izaaty
Jumlah antrian = 2
Zahara dikeluarkan dari antrian.
Izaaty dikeluarkan dari antrian.
Antrian telah dikosongkan.
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan8\Unguided\output>
```

2. Unguided2

a. Syntax

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

struct Node {
    string name;
    string nim;
    Node* next;
};
```

```
struct Queue {
    Node* front;
    Node* rear;
    int count;
};

Queue* createQueue() {
    Queue* queue = new Queue();
    queue->front = nullptr;
    queue->rear = nullptr;
    queue->count = 0;
    return queue;
}

bool isEmpty(Queue* queue) {
    return (queue->front == nullptr);
}

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

    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
    queue->count++;
      cout << "Mahasiswa " << name << " [NIM: " << nim << "] 
ditambahkan ke antrian." << endl;
}

void dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        cout << "Antrian kosong." << endl;
        return;
    }
    Node* temp = queue->front;
    cout << "Mahasiswa " << temp->name << " [NIM: " << temp->nim << 
"] dikeluarkan dari antrian." << endl;
    queue->front = queue->front->next;
    delete temp;
    queue->count--;

    if (queue->front == nullptr) {
```

```
        queue->rear = nullptr;
        cout << "Antrian telah dikosongkan." << endl;
    }
}

void viewQueue(Queue* queue) {
    Node* current = queue->front;
    cout << "Data antrian mahasiswa:" << endl;
    int index = 1;
    while (current != nullptr) {
        cout << index++ << ". Nama: " << current->name << ", NIM: " <<
current->nim << endl;
        current = current->next;
    }
    cout << "Jumlah antrian = " << queue->count << endl;
}

int main() {
    Queue* queue = createQueue();
    enqueue(queue, "Zaenarif Putra 'Ainurdin", "2311104049");
    enqueue(queue, "Izzaty Zahara", "2311104052");
    viewQueue(queue);
    dequeue(queue);
    viewQueue(queue);
    dequeue(queue);
    dequeue(queue);
    return 0;
}
```

b. Penjelasan Syntax

- Struktur Node: Struktur Node didefinisikan untuk merepresentasikan elemen dalam antrian. Setiap Node memiliki tiga atribut: name, yang menyimpan nama mahasiswa bertipe string; nim, yang menyimpan NIM (Nomor Induk Mahasiswa) mahasiswa juga bertipe string; dan next, yang merupakan pointer yang menunjuk ke Node berikutnya dalam antrian.
- Struktur Queue: Struktur Queue didefinisikan untuk merepresentasikan antrian itu sendiri. Terdapat tiga atribut dalam struktur ini: front, yang merupakan pointer yang menunjuk ke elemen terdepan dalam antrian; rear, yang merupakan pointer yang menunjuk ke elemen terakhir dalam antrian; dan count, yang menyimpan jumlah elemen saat ini dalam antrian.
- Fungsi createQueue: Fungsi ini digunakan untuk membuat antrian baru. Di dalam fungsi ini, memori dialokasikan untuk sebuah objek Queue, di mana atribut front dan rear diinisialisasi dengan nullptr untuk menunjukkan bahwa antrian kosong, dan count diatur ke 0. Setelah itu, fungsi ini mengembalikan pointer ke antrian yang baru dibuat.
- Fungsi isEmpty: Fungsi ini bertugas untuk memeriksa apakah antrian kosong. Hal ini dilakukan dengan memeriksa apakah front adalah

nullptr. Jika front adalah nullptr, maka antrian dianggap kosong dan fungsi ini mengembalikan nilai true; sebaliknya, jika ada elemen dalam antrian, fungsi ini mengembalikan nilai false.

- Fungsi enqueue: Fungsi ini digunakan untuk menambahkan elemen (mahasiswa) ke dalam antrian. Di dalam fungsi ini, sebuah Node baru dibuat dan diisi dengan name dan nim yang diberikan sebagai parameter. Jika antrian kosong, maka front dan rear diatur untuk menunjuk ke newNode. Namun, jika antrian sudah berisi elemen, maka next dari rear diatur untuk menunjuk ke newNode, dan rear diperbarui untuk menunjuk ke newNode. Setelah penambahan, atribut count diincrement untuk mencerminkan jumlah elemen yang bertambah, dan sebuah pesan ditampilkan untuk mengonfirmasi bahwa mahasiswa telah ditambahkan ke antrian.
- Fungsi dequeue: Fungsi ini digunakan untuk menghapus elemen dari depan antrian. Pertama, fungsi ini memeriksa apakah antrian kosong. Jika kosong, pesan "Antrian kosong" ditampilkan. Jika tidak, elemen di depan antrian disimpan dalam variabel sementara (temp), dan nama serta NIM mahasiswa yang dikeluarkan ditampilkan. Kemudian, front diperbarui untuk menunjuk ke elemen berikutnya, dan temp dihapus dari memori. Atribut count dikurangi untuk mencerminkan jumlah elemen yang berkurang. Jika setelah penghapusan antrian menjadi kosong, rear juga diatur ke nullptr, dan pesan bahwa antrian telah dikosongkan ditampilkan.
- Fungsi viewQueue: Fungsi ini digunakan untuk menampilkan semua elemen dalam antrian. Di dalam fungsi ini, pointer current digunakan untuk melintasi setiap Node mulai dari front. Selama current tidak nullptr, nama dan NIM dari setiap mahasiswa ditampilkan dengan nomor urut. Setelah semua elemen ditampilkan, fungsi ini juga mencetak jumlah elemen yang tersisa dalam antrian.
- Fungsi main: Fungsi main adalah titik awal program. Di dalam fungsi ini, antrian baru dibuat dengan memanggil createQueue. Kemudian, beberapa mahasiswa ditambahkan ke dalam antrian menggunakan fungsi enqueue, dan antrian ditampilkan dengan memanggil viewQueue. Setelah itu, elemen di depan antrian dihapus menggunakan dequeue, dan antrian ditampilkan kembali untuk menunjukkan perubahan. Proses penghapusan dilanjutkan hingga semua elemen dihapus.

c. Output

```
Mahasiswa Zaenarif Putra 'Ainurdin [NIM: 2311104049] ditambahkan ke antrian.
Mahasiswa Izzaty Zahara [NIM: 2311104052] ditambahkan ke antrian.
Data antrian mahasiswa:
1. Nama: Zaenarif Putra 'Ainurdin, NIM: 2311104049
2. Nama: Izzaty Zahara, NIM: 2311104052
Jumlah antrian = 2
Mahasiswa Zaenarif Putra 'Ainurdin [NIM: 2311104049] dikeluarkan dari antrian.
Data antrian mahasiswa:
1. Nama: Izzaty Zahara, NIM: 2311104052
Jumlah antrian = 1
Mahasiswa Izzaty Zahara [NIM: 2311104052] dikeluarkan dari antrian.
Antrian telah dikosongkan.
Antrian kosong.
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan8\Unguided\output>
```

3. Unguided3

a. Syntax

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

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

struct Queue {
    Node* front;
    Node* rear;
    int count;
};

Queue* createQueue() {
    Queue* queue = new Queue();
    queue->front = nullptr;
    queue->rear = nullptr;
    queue->count = 0;
    return queue;
}

bool isEmpty(Queue* queue) {
    return (queue->front == nullptr);
}

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

    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        if (nim < queue->front->nim) {
            newNode->next = queue->front;
            queue->front = newNode;
        } else {
            Node* current = queue->front;
            while (current->next != nullptr && current->next->nim < nim) {
                current = current->next;
```

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

            if (newNode->next == nullptr) {
                queue->rear = newNode;
            }
        }
    }
    queue->count++;
    cout << "Mahasiswa " << name << " (NIM: " << nim << ") ditambahkan ke antrian." << endl;
}

void dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        cout << "Antrian kosong." << endl;
        return;
    }
    Node* temp = queue->front;
    cout << "Mahasiswa " << temp->name << " (NIM: " << temp->nim << ") dikeluarkan dari antrian." << endl;
    queue->front = queue->front->next;
    delete temp;
    queue->count--;

    if (queue->front == nullptr) {
        queue->rear = nullptr;
        cout << "Antrian telah dikosongkan." << endl;
    }
}

void viewQueue(Queue* queue) {
    Node* current = queue->front;
    cout << "Data antrian mahasiswa:" << endl;
    int index = 1;
    while (current != nullptr) {
        cout << index++ << ". Nama: " << current->name << ", NIM: " << current->nim << endl;
        current = current->next;
    }
    cout << "Jumlah antrian = " << queue->count << endl;
}

int main() {
    Queue* queue = createQueue();
    int n;

    cout << "Masukkan jumlah mahasiswa yang ingin ditambahkan: ";
    cin >> n;
```

```
    cin.ignore();

    for (int i = 0; i < n; i++) {
        string name, nim;
        cout << "Masukkan nama mahasiswa: ";
        getline(cin, name);
        cout << "Masukkan NIM mahasiswa: ";
        getline(cin, nim);
        enqueue(queue, name, nim);
    }

    viewQueue(queue);

    while (!isEmpty(queue)) {
        dequeue(queue);
        viewQueue(queue);
    }

    return 0;
}
```

b. Penjelasan Syntax

- Header dan Namespace: Kode dimulai dengan mengimpor pustaka standar C++ untuk input dan output menggunakan #include <iostream> dan #include <string>. Penggunaan using namespace std; memungkinkan kita untuk menggunakan elemen dari pustaka standar tanpa perlu menulis std:: setiap kali, sehingga membuat kode lebih bersih dan mudah dibaca.
- Struktur Node: Struktur Node didefinisikan untuk merepresentasikan elemen dalam antrian. Setiap Node memiliki tiga atribut: name, yang menyimpan nama mahasiswa (bertipe string); nim, yang menyimpan NIM mahasiswa (juga bertipe string); dan next, yang merupakan pointer yang menunjuk ke Node berikutnya dalam antrian. Struktur ini memungkinkan kita untuk menyimpan informasi tentang setiap mahasiswa yang ada dalam antrian.
- Struktur Queue: Struktur Queue didefinisikan untuk merepresentasikan antrian itu sendiri. Dalam struktur ini terdapat tiga atribut: front, yang merupakan pointer yang menunjuk ke elemen terdepan dalam antrian; rear, yang merupakan pointer yang menunjuk ke elemen terakhir dalam antrian; dan count, yang menyimpan jumlah elemen saat ini dalam antrian. Struktur ini memungkinkan kita untuk mengelola antrian dengan efisien.
- Fungsi createQueue: Fungsi ini digunakan untuk membuat antrian baru. Di dalam fungsi ini, memori dialokasikan untuk sebuah objek Queue, di mana atribut front dan rear diinisialisasi dengan nullptr untuk menunjukkan bahwa antrian kosong, dan count diatur ke 0. Setelah itu, fungsi ini mengembalikan pointer ke antrian yang baru dibuat.
- Fungsi isEmpty: Fungsi ini bertugas untuk memeriksa apakah antrian kosong. Hal ini dilakukan dengan memeriksa apakah front adalah

nullptr. Jika front adalah nullptr, maka antrian dianggap kosong dan fungsi ini mengembalikan nilai true; sebaliknya, jika ada elemen dalam antrian, fungsi ini mengembalikan nilai false.

- Fungsi enqueue: Fungsi ini digunakan untuk menambahkan elemen (mahasiswa) ke dalam antrian dengan urutan berdasarkan NIM. Di dalam fungsi ini, sebuah Node baru dibuat dan diisi dengan name dan nim yang diberikan sebagai parameter. Jika antrian kosong, maka front dan rear diatur untuk menunjuk ke newNode. Jika antrian tidak kosong, fungsi ini memeriksa apakah NIM mahasiswa yang baru lebih kecil dari NIM mahasiswa di depan antrian. Jika iya, newNode akan menjadi elemen pertama, dan front diupdate. Jika tidak, fungsi ini akan melintasi antrian untuk menemukan posisi yang tepat berdasarkan urutan NIM, dan menyisipkan newNode di posisi yang sesuai. Jika newNode ditambahkan di akhir, maka rear juga diperbarui. Setelah penambahan, atribut count diincrement untuk mencerminkan jumlah elemen yang bertambah, dan sebuah pesan ditampilkan untuk mengonfirmasi bahwa mahasiswa telah ditambahkan ke antrian.
- Fungsi dequeue: Fungsi ini digunakan untuk menghapus elemen dari depan antrian. Pertama, fungsi ini memeriksa apakah antrian kosong. Jika kosong, pesan "Antrian kosong" ditampilkan. Jika tidak, elemen di depan antrian disimpan dalam variabel sementara (temp), dan nama serta NIM mahasiswa yang dikeluarkan ditampilkan. Kemudian, front diperbarui untuk menunjuk ke elemen berikutnya, dan temp dihapus dari memori. Atribut count dikurangi untuk mencerminkan jumlah elemen yang berkurang. Jika setelah penghapusan antrian menjadi kosong, rear juga diatur ke nullptr, dan pesan bahwa antrian telah dikosongkan ditampilkan.
- Fungsi viewQueue: Fungsi ini digunakan untuk menampilkan semua elemen dalam antrian. Di dalam fungsi ini, pointer current digunakan untuk melintasi setiap Node mulai dari front. Selama current tidak nullptr, nama dan NIM dari setiap mahasiswa ditampilkan dengan nomor urut. Setelah semua elemen ditampilkan, fungsi ini juga mencetak jumlah elemen yang tersisa dalam antrian.
- Fungsi main: Fungsi main adalah titik awal program. Di dalam fungsi ini, antrian baru dibuat dengan memanggil createQueue. Kemudian, beberapa mahasiswa ditambahkan ke dalam antrian menggunakan fungsi enqueue, dan antrian ditampilkan dengan memanggil viewQueue. Setelah itu, elemen di depan antrian dihapus menggunakan dequeue, dan antrian ditampilkan kembali untuk menunjukkan perubahan. Proses penghapusan dilanjutkan hingga semua elemen dihapus.

c. Output

```
Data antrian mahasiswa:
1. Nama: Riiij, NIM: 2311104001
2. Nama: Zuhyoo, NIM: 2311104050
3. Nama: Maharaja, NIM: 2311104088
4. Nama: Sriiij, NIM: 2311104090
Jumlah antrian = 4
Mahasiswa Riiij (NIM: 2311104001) dikeluarkan dari antrian.
Data antrian mahasiswa:
1. Nama: Zuhyoo, NIM: 2311104050
2. Nama: Maharaja, NIM: 2311104088
3. Nama: Sriiij, NIM: 2311104090
Jumlah antrian = 3
Mahasiswa Zuhyoo (NIM: 2311104050) dikeluarkan dari antrian.
Data antrian mahasiswa:
1. Nama: Maharaja, NIM: 2311104088
2. Nama: Sriiij, NIM: 2311104090
Jumlah antrian = 2
Mahasiswa Maharaja (NIM: 2311104088) dikeluarkan dari antrian.
Data antrian mahasiswa:
1. Nama: Sriiij, NIM: 2311104090
Jumlah antrian = 1
Mahasiswa Sriiij (NIM: 2311104090) dikeluarkan dari antrian.
Antrian telah dikosongkan.
Data antrian mahasiswa:
Jumlah antrian = 0
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan8\Unguided\output>
```

V. KESIMPULAN

Sebagai hasil dari praktik ini, siswa telah memperoleh pemahaman tentang konsep dasar dan bagaimana struktur queue data digunakan. Struktur ini mencakup prinsip FIFO (First-In, First-Out), serta operasi utama seperti enqueue dan dequeue, antara lain. Selain mendorong pengembangan logika pemrograman melalui tugas memodifikasi program untuk prioritas tertentu, praktik ini mengajarkan penerapan queue menggunakan array dan daftar terhubung. Setelah praktikum ini selesai, diharapkan siswa dapat menggunakan konsep queue untuk menyelesaikan masalah yang memerlukan pengelolaan data yang terorganisir dan efisien.