LAPORAN PRAKTIKUM

Modul 8

ANTREAN

Disusun Oleh:

Adhiansyah Muhammad Pradana Farawowan - 2211104038

S1SE-07-02

Asisten Praktikum:

Aldi Putra

Andini Nur Hidayah

Dosen:

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

PROGRAM STUDI S1 REKAYASAN PERANGKAT LUNAK

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM PURWOKERTO

2024

A. Tujuan

Laporan praktikum ini memiliki tujuan di bawah berikut.

1. Memperkenalkan konsep *queue*, selanjutnya akan disebut *antrean*, sebagai salah satu struktur data
2. Memahami dan mengelola cara kerja antrean
3. Mengimplementasikan antrean dalam bahasa C++

B. Landasan Teori

Antrean (*queue*) adalah struktur data yang bekerja dengan "data yang masuk pertama dulu adalah yang akan keluar dulu" (*first in, first out*, FIFO). Antrean memiliki variabel *front* dan *back* untuk melacak dua ujung strukturnya. Antrean memiliki beberapa operasi di bawah berikut.

1. Menambah data ke antrean
   Menambah data ke antrean dilakukan oleh operasi `enqueue()`.

2. Mengurangi data antrean
   Mengurangi data antrean dilakukan oleh operasi `dequeue()`.

3. Melihat data paling depan
   Melihat data paling depan dilakukan oleh operasi `peek()`.

4. Mengecek apakah antrean kosong
   Mengecek apakah antrean kosong dilakukan oleh operasi `is_empty()`.

5. Mengecek apakah antrean penuh
   Mengecek apakah antrean penuh dilakukan oleh operasi `is_full()`.

6. Melihat jumlah banyaknya data dalam antrean

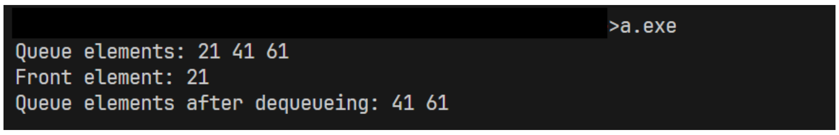
C. Bimbingan (*guided*)

Bimbingan hari ini adalah mengimplementasikan sebuah antrean dengan larik dan daftar berantai, dengan modifikasi sendiri.

guided_1.cpp

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

// Kelas untuk antrian
class Queue
{
private:
    int front, rear;
    int arr[MAX];

public:
    Queue()
    {
        front = -1;
        rear = -1;
    }
    bool is_full()
    {
        return rear == MAX - 1;
    }

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

    void enqueue(int x)
    {
        if (is_full())
        {
            std::cout << "Queue is full!\n";
            return;
        }
        if (front == -1)
            front = 0;
        arr[++rear] = x;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Queue is empty!\n";
            return;
        }
        front++;
    }

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

    void display()
    {
        if (is_empty())
        {
            std::cout << "Queue is empty!\n";
            return;
        }
        for (int i = front; i <= rear; i = i + 1)
        {
            std::cout << arr[i] << " ";
        }
        std::cout << "\n";
    }
};

int main()
{
    Queue q;

    q.enqueue(21);
    q.enqueue(41);
    q.enqueue(61);

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

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

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

    return 0;
}
```

Output dari guided_1.cpp

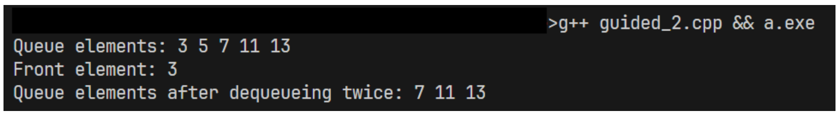


```
[redacted]>a.exe
Queue elements: 21 41 61
Front element: 21
Queue elements after dequeueing: 41 61
```

guided_2.cpp

```cpp
#include <iostream>

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

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

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

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

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

    void enqueue(int x)
    {
        Node *newNode = new Node(x);

        if (is_empty())
        {
            front = rear = newNode;
            return;
        }
        rear->next = newNode;
        rear = newNode;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Queue is empty!\n";
            return;
        }
        Node *temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr)
            rear = nullptr;
    }

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

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

int main()
{
    Queue q;

    q.enqueue(3);
    q.enqueue(5);
    q.enqueue(7);
    q.enqueue(11);
    q.enqueue(13);

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

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

    std::cout << "Queue elements after dequeueing twice: ";
    q.display();

    return 0;
}
```

Output dari guided_2.cpp



```
[redacted]>g++ guided_2.cpp && a.exe
Queue elements: 3 5 7 11 13
Front element: 3
Queue elements after dequeueing twice: 7 11 13
```

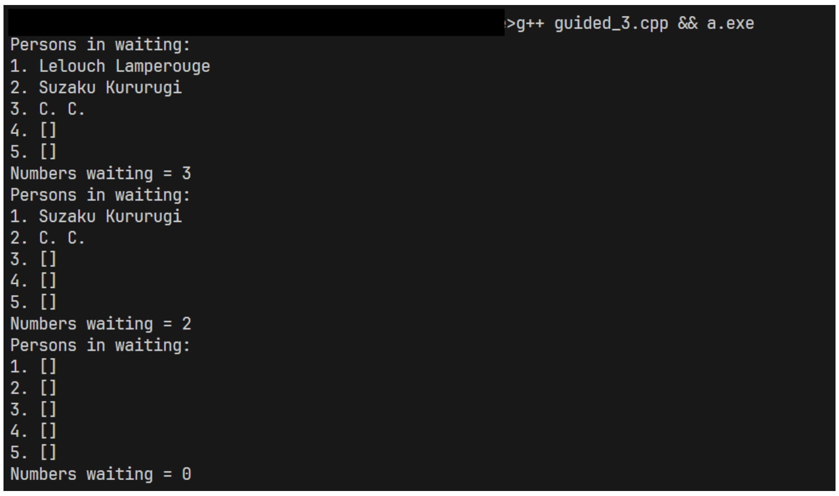
Pada `guided_3.cpp` terdapat kesalahan kode yang diberikan oleh asisten praktikum pada fungsi untuk menambah antrean. Instruksi untuk menambah data jika isi tidak kosong secara tidak sengaja ditulis dalam komentar, membuat jumlah isi dalam antrean tetap satu (bisa dilihat pada beberapa laporan teman-teman). Perbaikan sudah disertakan dalam berkas ini.

`guided_3.cpp`

```
#include <iostream>

const int MAX = 5;
int front = 0;
int back = 0;
std::string teller_queue[5];
```

```cpp
bool is_full()
{
    if (back == MAX)
    {
        return true;
    }
    else
    {
        return false;
    }
}

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

void enqueue(std::string data)
{
    if (is_full())
    {
        std::cout << "Queue is full!" << '\n';
    }
    else
    {
        if (is_empty())
        {
            teller_queue[0] = data;
            front++;
            back++;
        }
        else
        {
            teller_queue[back] = data; back++;
        }
    }
}

void dequeue()
{
    if (is_empty())
    {
        std::cout << "Queue is empty!" << '\n';
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            teller_queue[i] = teller_queue[i + 1];
        }
        back--;
    }
}

int count_queue_elements()
{
    return back;
}

void clear_queue()
{
    if (is_empty())
    {
        std::cout << "Queue is empty!" << '\n';
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            teller_queue[i] = "";
        }
        back = 0;
        front = 0;
    }
}

void display_queue()
{
    std::cout << "Persons in waiting:" << '\n';
    for (int i = 0; i < MAX; i++)
    {
        if (teller_queue[i] != "")
        {
            std::cout << i + 1 << ". " << teller_queue[i] << '\n';
        }
        else
        {
            std::cout << i + 1 << ". " << "[]" << '\n';
        }
    }
}

int main()
{
    enqueue("Lelouch Lamperouge");
    enqueue("Suzaku Kururugi");
    enqueue("C. C.");

    display_queue();
    std::cout << "Numbers waiting = " << count_queue_elements() << '\n';

    dequeue();
    display_queue();
    std::cout << "Numbers waiting = " << count_queue_elements() << '\n';

    clear_queue();
    display_queue();
    std::cout << "Numbers waiting = " << count_queue_elements() << '\n';

    return 0;
}
{
        data = value;
        next = nullptr;
    }
};

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

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

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

    void enqueue(int x)
    {
        Node *newNode = new Node(x);

        if (is_empty())
        {
            front = rear = newNode;
            return;
        }
        rear->next = newNode;
        rear = newNode;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Queue is empty!\n";
            return;
        }
        Node *temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr)
            rear = nullptr;
    }

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

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

int main()
{
    Queue q;

    q.enqueue(3);
    q.enqueue(5);
    q.enqueue(7);
    q.enqueue(11);
    q.enqueue(13);

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

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

    std::cout << "Queue elements after dequeueing twice: ";
    q.display();

    return 0;
}
```

Output dari guided_3.cpp



```
[redacted]>g++ guided_3.cpp && a.exe
Persons in waiting:
1. Lelouch Lamperouge
2. Suzaku Kururugi
3. C. C.
4. []
5. []
Numbers waiting = 3
Persons in waiting:
1. Suzaku Kururugi
2. C. C.
3. []
4. []
5. []
Numbers waiting = 2
Persons in waiting:
1. []
2. []
3. []
4. []
5. []
Numbers waiting = 0
```

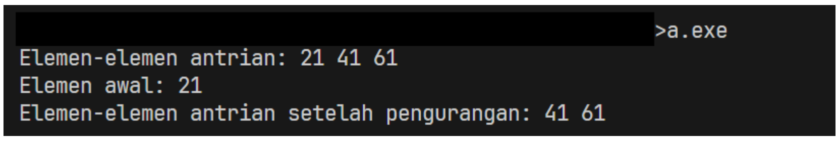
D. Tugas mandiri (*unguided)*

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

unguided_1.cpp

```
// Diadaptasi ulang dari kode-kode GUIDED
#include <iostream>

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

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

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

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

    bool is_empty()
    {
        return front == nullptr;
    }
    void enqueue(int x)
    {
        Node *newNode = new Node(x);

        if (is_empty())
        {
            front = rear = newNode;
            return;
        }
        rear->next = newNode;
        rear = newNode;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }
        Node *temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr)
            rear = nullptr;
    }

    int peek()
    {
        if (!is_empty())
        {
            return front->data;
        }
        std::cout << "Antrian kosong!\n";
        return -1;
    }

    void display()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }
        Node *current = front;
        while (current)
        {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << "\n";
    }
};

int main()
{
    Queue q;

    q.enqueue(21);
    q.enqueue(41);
    q.enqueue(61);

    std::cout << "Elemen-elemen antrian: ";
    q.display();

    std::cout << "Elemen awal: " << q.peek() << "\n";
    q.dequeue();

    std::cout << "Elemen-elemen antrian setelah pengurangan: ";
    q.display();

    return 0;
}
```

Output dari unguided_1.cpp



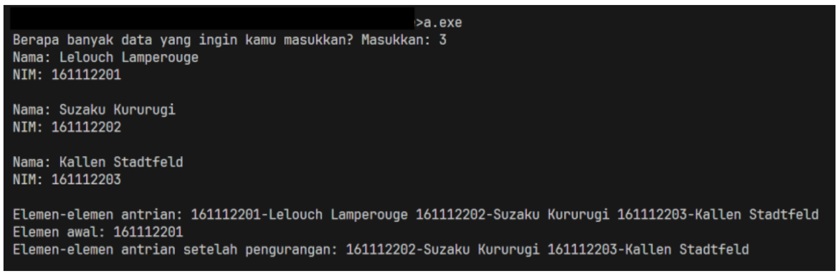
```
[redacted]>a.exe
Elemen-elemen antrian: 21 41 61
Elemen awal: 21
Elemen-elemen antrian setelah pengurangan: 41 61
```

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

unguided_2.cpp

```
// Diadaptasi ulang dari kode-kode GUIDED
#include <iostream>
#include <string>

struct Mhs
{
    int nim;
    std::string nama;
};

class Node
{
public:
    struct Mhs data;
    Node *next;

    Node(struct Mhs value)
    {
        data = value;
        next = nullptr;
    }
};

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

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

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

    void enqueue(struct Mhs x)
    {
        Node *new_node = new Node(x);

        if (is_empty())
        {
            front = new_node;
            back = new_node;
            return;
        }
        back->next = new_node;
        back = new_node;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }
        Node *temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr)
            back = nullptr;
    }

    int peek()
    {
        if (!is_empty())
        {
            return front->data.nim;
        }
        std::cout << "Antrian kosong!\n";
        return -1;
    }

    void display()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }
        Node *current = front;
        while (current)
        {
            std::cout << current->data.nim << "-" << current->data.nama << " ";
            current = current->next;
        }
        std::cout << "\n";
    }
};

int main()
{
    Queue q;

    int much_data = 0;

    struct Mhs mahasiswa;
    mahasiswa.nama = "";
    mahasiswa.nim = 0;

    std::cout << "Berapa banyak data yang ingin kamu masukkan? Masukkan: ";
    std::cin >> much_data;
    std::cin.ignore();

    for (int i = 0; i < much_data; i = i + 1)
    {
        std::cout << "Nama: ";
        std::getline(std::cin, mahasiswa.nama);

        std::cout << "NIM: ";
        std::cin >> mahasiswa.nim;
        std::cin.ignore();

        q.enqueue(mahasiswa);
        std::cout << '\n';
    }

    std::cout << "Elemen-elemen antrian: ";
    q.display();

    std::cout << "Elemen awal: " << q.peek() << "\n";
    q.dequeue();

    std::cout << "Elemen-elemen antrian setelah pengurangan: ";
    q.display();

    return 0;
}
```

Output dari unguided_2.cpp



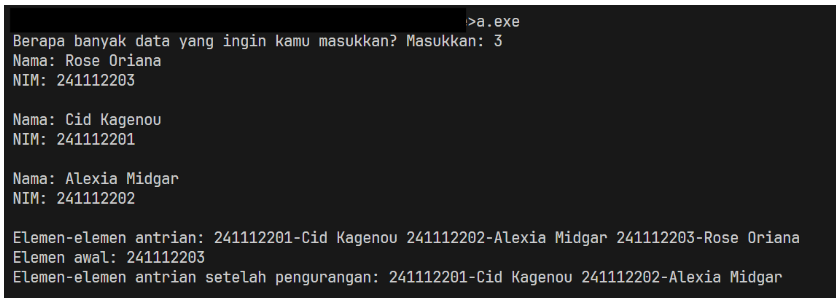
```
[redacted]>a.exe
Berapa banyak data yang ingin kamu masukkan? Masukkan: 3
Nama: Lelouch Lamperouge
NIM: 161112201

Nama: Suzaku Kururugi
NIM: 161112202

Nama: Kallen Stadtfeld
NIM: 161112203

Elemen-elemen antrian: 161112201-Lelouch Lamperouge 161112202-Suzaku Kururugi 161112203-Kallen Stadtfeld
Elemen awal: 161112201
Elemen-elemen antrian setelah pengurangan: 161112202-Suzaku Kururugi 161112203-Kallen Stadtfeld
```

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

unguided_3.cpp

```
// Diadaptasi ulang dari kode-kode GUIDED
#include <iostream>
#include <string>

struct Mhs
{
    int nim;
    std::string nama;
};

class Node
{
public:
    struct Mhs data;
    Node *next;

    Node(struct Mhs value)
    {
        data = value;
        next = nullptr;
    }
};

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

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

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

    void enqueue(struct Mhs x)
    {
        Node *new_node = new Node(x);

        if (is_empty())
        {
            front = new_node;
            back = new_node;
            return;
        }
        back->next = new_node;
        back = new_node;
    }

    void dequeue()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }
        Node *temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr)
            back = nullptr;
    }

    int peek()
    {
        if (!is_empty())
        {
            return front->data.nim;
        }
        std::cout << "Antrian kosong!\n";
        return -1;
    }

    void display()
    {
        if (is_empty())
        {
            std::cout << "Antrian kosong!\n";
            return;
        }

        // Biarkan antrean tetap utuh, ubah hanya pada outputnya
        Node *current = front;
        Node *sorted = nullptr;
        Node *temp = nullptr;

        while (current != nullptr)
        {
            Node *new_node = new Node(current->data);
            if (sorted == nullptr || sorted->data.nim >= new_node->data.nim)
            {
                new_node->next = sorted;
                sorted = new_node;
            }
            else
            {
                temp = sorted;
                while (temp->next != nullptr && temp->next->data.nim <
new_node->data.nim)
                {
                    temp = temp->next;
                }
                new_node->next = temp->next;
                temp->next = new_node;
            }
            current = current->next;
        }
        current = sorted;
        while (current != nullptr)
        {
            std::cout << current->data.nim << "-" << current->data.nama << " ";
            current = current->next;
        }
        std::cout << "\n";
    }
};

int main()
{
    Queue q;
```

```cpp
    int much_data = 0;

    struct Mhs mahasiswa;
    mahasiswa.nama = "";
    mahasiswa.nim = 0;

    std::cout << "Berapa banyak data yang ingin kamu masukkan? Masukkan: ";
    std::cin >> much_data;
    std::cin.ignore();

    for (int i = 0; i < much_data; i = i + 1)
    {
        std::cout << "Nama: ";
        std::getline(std::cin, mahasiswa.nama);

        std::cout << "NIM: ";
        std::cin >> mahasiswa.nim;
        std::cin.ignore();

        q.enqueue(mahasiswa);
        std::cout << '\n';
    }

    std::cout << "Elemen-elemen antrian: ";
    q.display();

    std::cout << "Elemen awal: " << q.peek() << "\n";
    q.dequeue();

    std::cout << "Elemen-elemen antrian setelah pengurangan: ";
    q.display();

    return 0;
}
```

Output dari unguided_3.cpp


```
[redacted]>a.exe
Berapa banyak data yang ingin kamu masukkan? Masukkan: 3
Nama: Rose Oriana
NIM: 241112203

Nama: Cid Kagenou
NIM: 241112201

Nama: Alexia Midgar
NIM: 241112202

Elemen-elemen antrian: 241112201-Cid Kagenou 241112202-Alexia Midgar 241112203-Rose Oriana
Elemen awal: 241112203
Elemen-elemen antrian setelah pengurangan: 241112201-Cid Kagenou 241112202-Alexia Midgar
```

Noted : Untuk data mahasiswa dan nim dimasukan oleh user