LAPORAN PRAKTIKUM

Modul 8

“QUEUE”

Disusun Oleh:

Rengganis Tantri Pramudita - 2311104065

SE0702

Dosen :

Wahyu Andy Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan

- Mahasiswa mampu menjelaskan definisi dan konsep dari queue
- Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
- Mahasiswa mampu menerapkan operasi tampil data pada queue

2. Landasan Teori

Queue (antrian) adalah salah satu struktur data yang berfungsi sebagai tempat penyimpanan elemen-elemen yang diakses berdasarkan prinsip *First In, First Out* (FIFO). Artinya, elemen yang pertama kali dimasukkan ke dalam queue akan menjadi elemen pertama yang dikeluarkan. Konsep ini menyerupai antrian dalam kehidupan nyata, seperti antrian di loket atau kasir.

Karakteristik Queue:

1. Operasi Utama:
   - Enqueue: Menambahkan elemen baru ke belakang antrian.
   - Dequeue: Menghapus elemen dari depan antrian.
2. Akses Elemen: Hanya dua ujung queue yang dapat diakses, yaitu:
   - Front: Elemen pertama di antrian (tempat elemen dihapus).
   - Rear: Elemen terakhir di antrian (tempat elemen ditambahkan).

3. 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;
}
```

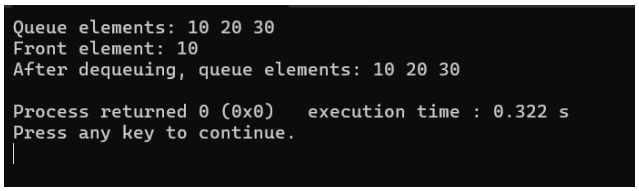
Keterangan:

Kode diatas mengimplementasikan struktur data Queue (antrian) menggunakan array di C++. Queue mengikuti prinsip First In, First Out (FIFO), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar.

Operasi dasar yag digunakan:

1. Enqueue (Menambahkan Elemen ke Antrian)
2. Dequeue (Menghapus Elemen dari Antrian)
3. Peek (Melihat Elemen Terdepan Tanpa Menghapusnya)
4. Display (Menampilkan Semua Elemen dalam Antrian)

Outputnya



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;
}
```

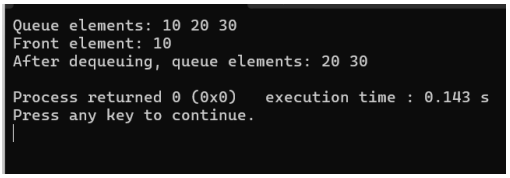
Kode di atas adalah implementasi Queue menggunakan linked list

Keterangan:

Operasi dasar yang digunakan

1. Enqueue (Menambahkan Elemen ke Queue)
2. Dequeue (Menghapus Elemen dari Queue)
3. Peek (Melihat Elemen Depan Tanpa Menghapusnya)
4. Display (Menampilkan Semua Elemen Queue)

Outputnya:



Guided 3

```
#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;
}
```

Keterangan

Kode di atas adalah implementasi Queue menggunakan array di C++, dengan fungsi tambahan seperti menghitung jumlah elemen, membersihkan antrian, dan menampilkan isi antrian

Operasi dasarnya

1. Enqueue (Menambahkan Elemen ke Queue)
2. Dequeue (Menghapus Elemen dari Queue)
3. View Queue (Menampilkan Elemen Queue)
4. Clear Queue (Menghapus Semua Elemen dalam Queue)
5. Count Queue (Menghitung Jumlah Elemen dalam Queue)

Outputnya

4. Unguided

Soal 1

Kode Programnya

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

using namespace std;

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

Node *front = nullptr;
Node *back = nullptr;

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

```
void enqueueAntrian(string data)
{
   Node *newNode = new Node();
   newNode->data = data;
   newNode->next = nullptr;

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

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

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

```
}

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

int main()
{
   enqueueAntrian("Andi");
   enqueueAntrian("Maya");

   viewQueue();

   dequeueAntrian();
   viewQueue();

   clearQueue();
   viewQueue();

   return 0;
}
```

Outputnya

```
Data antrian:
Andi
Maya
Data antrian:
Maya
Antrian kosong

Process returned 0 (0x0)   execution time : 0.464 s
Press any key to continue.
```

Soal 2

Kode Programnya

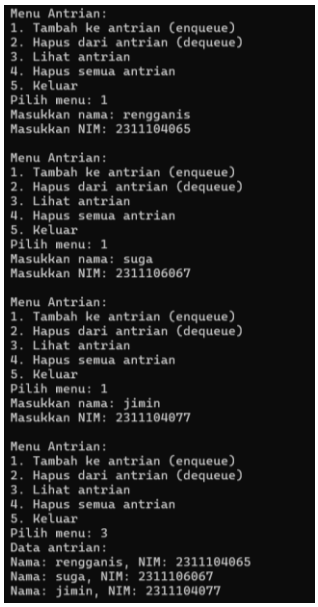
```
#include <iostream>
#include <string>

using namespace std;

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

Node *front = nullptr;
```

```
Node *back = nullptr;

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

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

void viewQueue()
{
   if (isEmpty())
   {
      cout << "Antrian kosong" << endl;
   }
   else
   {
      Node *temp = front;
```

```
        cout << "Data antrian:" << endl;
        while (temp != nullptr)
        {
            cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
            temp = temp->next;
        }
    }
}

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

int main()
{
    int pilihan;
    string nama, nim;

    do
    {
        cout << "\nMenu Antrian:" << endl;
        cout << "1. Tambah ke antrian (enqueue)" << endl;
        cout << "2. Hapus dari antrian (dequeue)" << endl;
        cout << "3. Lihat antrian" << endl;
        cout << "4. Hapus semua antrian" << endl;
        cout << "5. Keluar" << endl;
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            cout << "Masukkan nama: ";
            cin.ignore(); // Membersihkan buffer input
            getline(cin, nama);
            cout << "Masukkan NIM: ";
            getline(cin, nim);
            enqueueAntrian(nama, nim);
            break;
        case 2:
            dequeueAntrian();
            break;
        case 3:
            viewQueue();
            break;
        case 4:
```

```
            clearQueue();
            cout << "Semua antrian telah dihapus." << endl;
            break;
        case 5:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid." << endl;
        }
    } while (pilihan != 5);

    return 0;
}
```

Outpunya



Soal 3

Kode Program

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

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

class PriorityQueue {
private:
    Node* depan;

public:
```

```
    PriorityQueue() {
        depan = nullptr;
    }

    bool kosong() {
        return depan == nullptr;
    }

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

        if (kosong() || nim < depan->nim) {
            nodeBaru->berikut = depan;
            depan = nodeBaru;
        } else {
            Node* sementara = depan;
            while (sementara->berikut != nullptr && sementara->berikut->nim <= nim) {
                sementara = sementara->berikut;
            }
            nodeBaru->berikut = sementara->berikut;
            sementara->berikut = nodeBaru;
        }
    }

    void dequeue() {
        if (kosong()) {
            cout << "Antrian kosong (Underflow)\n";
            return;
        }

        Node* sementara = depan;
        depan = depan->berikut;
        delete sementara;
    }

    void tampilkan() {
        if (kosong()) {
            cout << "Antrian kosong\n";
            return;
        }

        Node* sementara = depan;
        while (sementara != nullptr) {
            cout << "Nama: " << sementara->nama << ", NIM: " << sementara->nim <<
"\n";
            sementara = sementara->berikut;
        }
```

```
    }
};

int main() {
    PriorityQueue antrian;
    int jumlah;

    cout << "Masukkan jumlah mahasiswa yang ingin diinput: ";
    cin >> jumlah;

    for (int i = 0; i < jumlah; i++) {
        string nama, nim;
        cout << "Masukkan nama mahasiswa: ";
        cin >> ws;
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;
        antrian.enqueue(nama, nim);
    }

    cout << "\nData antrian mahasiswa (berdasarkan prioritas NIM):\n";
    antrian.tampilkan();

    return 0;
}
```

Outputnya

```
Masukkan jumlah mahasiswa yang ingin diinput: 3
Masukkan nama mahasiswa: rengganis
Masukkan NIM mahasiswa: 2311104065
Masukkan nama mahasiswa: suga
Masukkan NIM mahasiswa: 2311104061
Masukkan nama mahasiswa: jimin
Masukkan NIM mahasiswa: 2311104010

Data antrian mahasiswa (berdasarkan prioritas NIM):
Nama: jimin, NIM: 2311104010
Nama: suga, NIM: 2311104061
Nama: rengganis, NIM: 2311104065
```

5. Kesimpulan

Queue adalah struktur data linear dengan prinsip FIFO (First In, First Out), di mana elemen pertama yang masuk adalah yang pertama keluar. Implementasi queue dapat dilakukan menggunakan array yang sederhana namun terbatas ukurannya, atau linked list yang lebih fleksibel dengan alokasi memori dinamis. Operasi dasar queue meliputi enqueue untuk menambah elemen, dequeue untuk menghapus elemen, serta fungsi tambahan seperti menghitung jumlah, mengosongkan, dan menampilkan elemen. Queue banyak digunakan dalam berbagai aplikasi seperti sistem antrian, penjadwalan, dan manajemen data.