LAPORAN PRAKTIKUM
Modul 8
QUEUE

Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
S1SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

1. **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

2. **Landasan Teori**

    Queue adalah struktur data yang menyimpan elemen dalam urutan linier dengan prinsip “First-In-First-Out” (FIFO), di mana elemen yang pertama kali dimasukkan adalah yang pertama kali dikeluarkan. Queue sering digunakan dalam berbagai aplikasi, seperti manajemen antrian dalam sistem operasi dan pengelolaan tugas dalam pemrograman.

    Queue adalah struktur data yang sangat penting dalam pemrograman dan pengolahan data. Memahami cara kerja dan penerapannya dapat membantu dalam mengelola alur data secara efisien dalam berbagai aplikasi teknologi.

    Melalui dua operasi dasar, yaitu enqueue dan dequeue, queue memungkinkan penambahan dan penghapusan elemen secara teratur, yang mendukung pengolahan data yang efisien. Implementasi queue dapat dilakukan menggunakan struktur data seperti array atau linked list, masing-masing dengan kelebihan dan kekurangan tersendiri. Queue dengan array menawarkan akses cepat tetapi memiliki ukuran tetap, sedangkan queue dengan linked list lebih fleksibel dalam hal penambahan elemen, meskipun memerlukan lebih banyak memori untuk menyimpan penunjuk.

    Secara keseluruhan, queue adalah alat yang sangat penting dalam pengembangan perangkat lunak dan sistem informasi, dan pemahaman yang baik tentangnya akan sangat bermanfaat dalam praktik pemrograman dan pengelolaan data.

3. **Guided**

    Guided 1

    Outputnya:

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
```

    Code programnya

```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;
}
```

Guided 2

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
```

Code programnya

Telkom University Purwokerto

```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;
}
```

Guided 3

Code programnya

Telkom University Purwokerto

```cpp
#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;
}
```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3
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
```

4. Unguided

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

   Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\
1. Andi
2. Maya
Jumlah antrian = 2
1. Maya
Jumlah antrian = 1
Queue kosong
Jumlah antrian = 0
```

   Code programnya

Telkom University Purwokerto

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

class QueueLinkedList {
private:
    struct Node {
        string data;
        Node* next;
        Node(string val) : data(val), next(nullptr) {}
    };
    Node *front, *rear;
    int size;

public:
    QueueLinkedList() : front(nullptr), rear(nullptr), size(0) {}

    void enqueue(string data) {
        Node* newNode = new Node(data);
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
        size++;
    }

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

    void viewQueue() {
        if (isEmpty()) {
            cout << "Queue kosong" << endl;
            return;
        }
        Node* current = front;
        int i = 1;
        while (current) {
            cout << i++ << ". " << current->data << endl;
            current = current->next;
        }
    }

    bool isEmpty() { return front == nullptr; }
    int getSize() { return size; }

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

int main() {
    QueueLinkedList queue;
    queue.enqueue("Andi");
    queue.enqueue("Maya");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.getSize() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.getSize() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.getSize() << endl;
    return 0;
}
```

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

Code programnya

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

class StudentQueue {
private:
    struct Student {
        string nama;
        string nim;
        Student* next;
        Student(string n, string id) : nama(n), nim(id), next(nullptr) {}
    };
    Student* front;
    Student* rear;

public:
    StudentQueue() : front(nullptr), rear(nullptr) {}

    void enqueue() {
        string nama, nim;
        cout << "Masukkan nama mahasiswa: ";
        cin.ignore(); // Membersihkan buffer sebelum getline
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;

        Student* newStudent = new Student(nama, nim);
        if (isEmpty()) {
            front = rear = newStudent;
        } else {
            rear->next = newStudent;
            rear = newStudent;
        }
        cout << "Mahasiswa " << nama << " ditambahkan ke antrian." << endl;
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }
        Student* temp = front;
        front = front->next;
        cout << "Mahasiswa " << temp->nama << " dikeluarkan dari antrian." << endl;
        delete temp;
        if (front == nullptr) rear = nullptr;
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }
        Student* current = front;
        int i = 1;
        while (current) {
            cout << i++ << ". Nama: " << current->nama
                 << ", NIM: " << current->nim << endl;
            current = current->next;
        }
    }

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

int main() {
    StudentQueue queue;
    int pilihan;
    do {
        cout << "\nMenu Antrian Mahasiswa:" << endl;
        cout << "1. Tambah Mahasiswa" << endl;
        cout << "2. Hapus Mahasiswa" << endl;
        cout << "3. Lihat Antrian" << endl;
        cout << "4. Keluar" << endl;
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch(pilihan) {
            case 1: queue.enqueue(); break;
            case 2: queue.dequeue(); break;
            case 3: queue.viewQueue(); break;
            case 4: cout << "Keluar dari program." << endl; break;
            default: cout << "Pilihan tidak valid!" << endl; break;
        }
    } while (pilihan != 4);

    return 0;
}
```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3\

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Andi
Masukkan NIM mahasiswa: 2311104054
Mahasiswa Andi ditambahkan ke antrian.

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Maya
Masukkan NIM mahasiswa: 2311104065
Mahasiswa Maya ditambahkan ke antrian.

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 3
1. Nama: Andi, NIM: 2311104054
2. Nama: Maya, NIM: 2311104065

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Andi dikeluarkan dari antrian.
```

```
Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Maya dikeluarkan dari antrian.

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 4
Keluar dari program.
```

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

Telkom University Purwokerto

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

class PrioritizedStudentQueue {
private:
    struct Student {
        string nama;
        string nim;
        Student* next;
        Student(string n, string id) : nama(n), nim(id), next(nullptr) {}
    };
    Student* front;
    Student* rear;

public:
    PrioritizedStudentQueue() : front(nullptr), rear(nullptr) {}

    void enqueue() {
        string nama, nim;
        cout << "Masukkan nama mahasiswa: ";
        cin.ignore(); // Membersihkan buffer sebelum getline
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;

        Student* newStudent = new Student(nama, nim);

        // Queue kosong
        if (isEmpty()) {
            front = rear = newStudent;
            return;
        }

        // NIM lebih kecil dari front, masukkan di depan
        if (nim < front->nim) {
            newStudent->next = front;
            front = newStudent;
            return;
        }

        // Cari posisi yang tepat berdasarkan NIM
        Student* current = front;
        while (current->next && current->next->nim < nim) {
            current = current->next;
        }

        newStudent->next = current->next;
        current->next = newStudent;

        // Update rear jika diperlukan
        if (newStudent->next == nullptr) {
            rear = newStudent;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }
        Student* temp = front;
        front = front->next;
        cout << "Mahasiswa " << temp->nama << " dikeluarkan dari antrian." << endl;
        delete temp;
        if (front == nullptr) rear = nullptr;
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }
        Student* current = front;
        int i = 1;
        while (current) {
            cout << i++ << ". Nama: " << current->nama
                 << ", NIM: " << current->nim << endl;
            current = current->next;
        }
    }

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

int main() {
    PrioritizedStudentQueue queue;
    int pilihan;
    do {
        cout << "\nMenu Antrian Mahasiswa Prioritas:" << endl;
        cout << "1. Tambah Mahasiswa" << endl;
        cout << "2. Hapus Mahasiswa" << endl;
        cout << "3. Lihat Antrian" << endl;
        cout << "4. Keluar" << endl;
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch(pilihan) {
            case 1: queue.enqueue(); break;
            case 2: queue.dequeue(); break;
            case 3: queue.viewQueue(); break;
            case 4: cout << "Keluar dari program." << endl; break;
            default: cout << "Pilihan tidak valid!" << endl; break;
        }
    } while (pilihan != 4);

    return 0;
}
```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Andi
Masukkan NIM mahasiswa: 2311104054

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Maya
Masukkan NIM mahasiswa: 2311104065

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 3
1. Nama: Andi, NIM: 2311104054
2. Nama: Maya, NIM: 2311104065

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Andi dikeluarkan dari antrian.
```

```
Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Maya dikeluarkan dari antrian.

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 4
Keluar dari program.
```

5. **Kesimpulan**

   Pada praktikum ini, kita mempelajari tentang struktur data Queue, termasuk konsep dasar, karakteristik, serta cara kerjanya. Selain itu, kita juga akan memahami bagaimana menerapkan Queue dalam berbagai kasus nyata, seperti sistem antrian, manajemen proses dalam sistem operasi, atau penanganan data secara terorganisir. Dengan memahami materi ini, kita diharapkan mampu mengimplementasikan Queue secara efektif dalam program dan memanfaatkan keunggulannya untuk memecahkan masalah secara efisien.