

LAPORAN PRAKTIKUM

MODUL VII QUEUE



Disusun oleh:
Geranada Saputra Priambudi
NIM: 2311102008

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

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024**

BAB I

TUJUAN PRAKTIKUM

- a. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue.
- b. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue.
- c. Mahasiswa mampu menerapkan operasi tampil data pada queue.

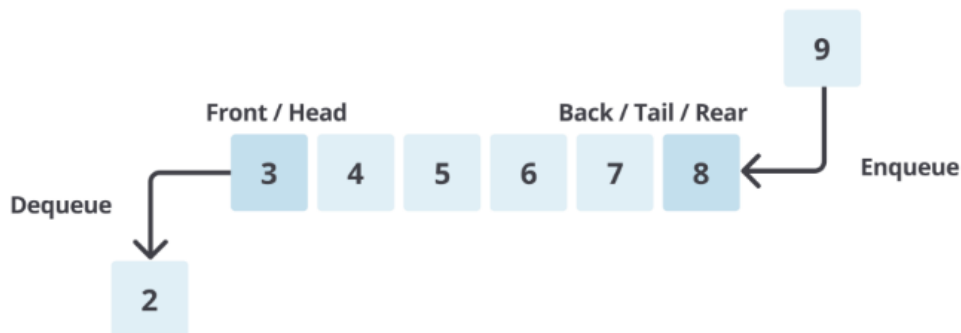
BAB II

DASAR TEORI

Queue atau dalam bahasa Indonesia yang berarti antrean adalah struktur data yang menyusun elemen-elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrean akan menjadi yang pertama pula untuk dikeluarkan.

Caranya bekerja adalah seperti jejeran orang yang sedang menunggu antrean di supermarket di mana orang pertama yang datang adalah yang pertama dilayani (First In, First Out). Pada struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut Front atau Head. Sebaliknya, data pada urutan terakhir (data yang baru saja ditambahkan) disebut Back, Rear, atau Tail. Proses untuk menambahkan data pada antrean disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrean disebut dengan Dequeue.

Queue Data Structure



Queue memiliki peran yang penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrean tugas atau operasi secara efisien. Dalam sistem komputasi, ia digunakan untuk menangani tugas-tugas seperti penjadwalan proses, antrean pesan, dan manajemen sumber daya.

BAB III

GUIDED

1. Guided 1

Source code

```
#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;
}

```

Screenshoot program

```

PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7> cd "d:\Telkom University\Teknik Informatika\Semester 2\Praktikum
Struktur data dan Algoritma\Modul7\\" ; if ($?) { g++ Guided.cpp -o Guided } ; if ($?) { .\Guided }
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
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
PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7>

```

Deskripsi program

Kode tersebut merupakan implementasi sederhana dari sebuah antrian (queue) dengan batas maksimal lima elemen, menggunakan array untuk menyimpan data antrian. Program ini memperlihatkan operasi dasar dari

sebuah antrian seperti menambah, menghapus, menampilkan, dan menghitung jumlah elemen dalam antrian.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

using namespace std;

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

class Queue {
private:
    Node* front;
    Node* rear;
    int count; // untuk melacak jumlah elemen dalam antrian
    const int maksimalQueue;

public:
    Queue(int size) : front(nullptr), rear(nullptr), count(0),
maksimalQueue(size) {}

    bool isFull() {
        return count == maksimalQueue;
    }

    bool isEmpty() {
        return count == 0;
    }
}
```

```

void enqueue(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        Node* newNode = new Node{data, nullptr};
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
        count++;
    }
}

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

int countQueue() {
    return count;
}

void clearQueue() {

```

```

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

    void viewQueue() {
        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". " << current->data << endl;
            current = current->next;
            index++;
        }
        for (; index <= maksimalQueue; index++) {
            cout << index << ". (kosong)" << endl;
        }
    }
};

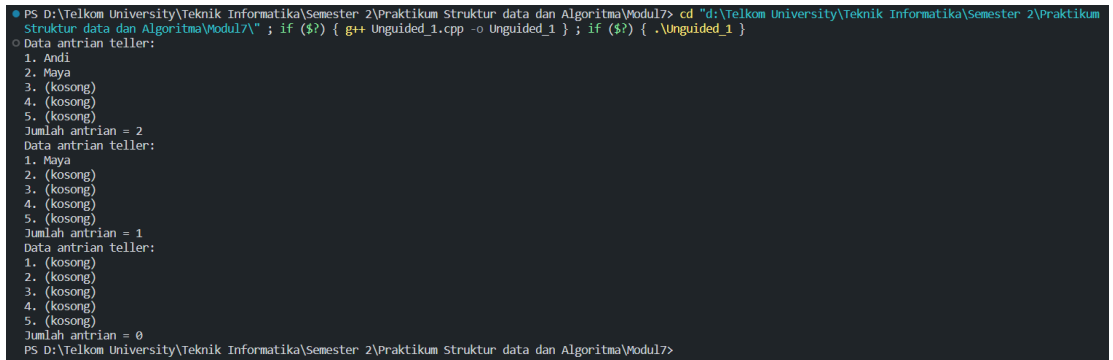
int main() {
    Queue queue(5); // membuat antrian dengan maksimal 5 elemen

    queue.enqueue("Andi");
    queue.enqueue("Maya");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
}

```

```
    return 0;
}
```

Screenshoot program



```
PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7> cd "d:\telkom university\Teknik Informatika\Semester 2\Praktikum
Struktur data dan Algoritma\Modul7\" ; if ($?) { g++ Unguided_1.cpp -o Unguided_1 } ; if ($?) { .\Unguided_1 }
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
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
PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7>
```

Deskripsi program

Kode tersebut merupakan implementasi dari antrian (queue) menggunakan struktur data linked list dalam bahasa C++. Program ini memperlihatkan operasi dasar dari antrian menggunakan linked list, seperti menambah, menghapus, menghitung, mengosongkan, dan menampilkan elemen antrian.

2. Unguided 2

Source code

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

using namespace std;

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

class Queue {
private:
    Node* front;
    Node* rear;
    int count; // untuk melacak jumlah elemen dalam antrian
    const int maksimalQueue;

public:
    Queue(int size) : front(nullptr), rear(nullptr), count(0),
maksimalQueue(size) {}

    bool isFull() {
        return count == maksimalQueue;
    }

    bool isEmpty() {
        return count == 0;
    }

    void enqueue(string nama, string nim) {
        if (isFull()) {
```

```

        cout << "Antrian penuh" << endl;
    } else {
        Node* newNode = new Node{nama, nim, nullptr};
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
        count++;
    }
}

void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        cout << "Dequeued: " << temp->nama << " (" << temp->
nim << ")" << endl;
        delete temp;
        count--;
        if (front == nullptr) {
            rear = nullptr;
        }
    }
}

int countQueue() {
    return count;
}

void clearQueue() {

```

```

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

    void viewQueue() {
        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". Nama: " << current->nama << ",
NIM: " << current->nim << endl;
            current = current->next;
            index++;
        }
        for (; index <= maksimalQueue; index++) {
            cout << index << ". (kosong)" << endl;
        }
    }
};

int main() {
    Queue queue(5); // membuat antrian dengan maksimal 5 elemen

    queue.enqueue("Andi", "21102320");
    queue.enqueue("Maya", "2211102003");
    queue.enqueue("Toyo", "20112032");
    queue.enqueue("Dodo", "2311103146");
    queue.enqueue("Agus", "21102183");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
}

```

```

queue.clearQueue();

queue.viewQueue();

cout << "Jumlah antrian = " << queue.countQueue() << endl;

return 0;

}

```

Screenshot program

```

PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7> cd "d:\Telkom University\Teknik Informatika\Semester 2\Praktikum
Struktur data dan Algoritma\Modul7\" ; if ($?) { g++ Unguided_2.cpp -o Unguided_2 } ; if ($?) { .\Unguided_2 }
Data antrian teller:
1. Nama: Andi, NIM: 21102320
2. Nama: Maya, NIM: 2211102003
3. Nama: Toyo, NIM: 20112032
4. Nama: Dodo, NIM: 2311103146
5. Nama: Agus, NIM: 21102183
Jumlah antrian = 5
Dequeued: Andi (21102320)
Data antrian teller:
1. Nama: Maya, NIM: 2211102003
2. Nama: Toyo, NIM: 20112032
3. Nama: Dodo, NIM: 2311103146
4. Nama: Agus, NIM: 21102183
5. (kosong)
Jumlah antrian = 4
Dequeued: Maya (2211102003)
Dequeued: Toyo (20112032)
Dequeued: Dodo (2311103146)
Dequeued: Agus (21102183)
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS D:\Telkom University\Teknik Informatika\Semester 2\Praktikum Struktur data dan Algoritma\Modul7>

```

Deskripsi program

Kode ini mengimplementasikan antrian (queue) menggunakan struktur data linked list dalam bahasa C++. Antrian ini menyimpan data mahasiswa yang terdiri dari nama dan NIM (Nomor Induk Mahasiswa). Program ini memperlihatkan operasi dasar dari antrian menggunakan linked list yang menyimpan data mahasiswa berupa nama dan NIM, termasuk menambah, menghapus, menghitung, mengosongkan, dan menampilkan elemen antrian.

BAB IV

KESIMPULAN

Praktikum ini bertujuan untuk memahami dan mengimplementasikan struktur data Queue menggunakan linked list. Queue adalah struktur data yang mengikuti prinsip FIFO (First In First Out), di mana elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan.

Pada praktikum ini, kami mengimplementasikan Queue menggunakan struktur data linked list. Implementasi dilakukan dalam bahasa pemrograman C++ dengan beberapa fungsi utama, yaitu:

`enqueue(nama, nim)`: Menambahkan elemen baru ke akhir antrian.

`dequeue()`: Menghapus elemen dari depan antrian dan menampilkan elemen yang dihapus.

`isFull()`: Memeriksa apakah antrian penuh.

`isEmpty()`: Memeriksa apakah antrian kosong.

`countQueue()`: Menghitung jumlah elemen dalam antrian.

`clearQueue()`: Mengosongkan antrian.

`viewQueue()`: Menampilkan semua elemen dalam antrian.

Prinsip FIFO: Implementasi queue menggunakan linked list ini menunjukkan prinsip dasar FIFO dengan baik. Elemen yang pertama kali ditambahkan menjadi elemen pertama yang dikeluarkan.

Dinamika Antrian: Penggunaan linked list untuk implementasi antrian menunjukkan fleksibilitas dalam mengelola elemen secara dinamis tanpa perlu memperhatikan ukuran tetap seperti pada array.

Manajemen Memori: Dengan linked list, manajemen memori menjadi lebih efisien karena elemen hanya dialokasikan saat diperlukan dan dibebaskan ketika tidak lagi digunakan.

Fungsi Pendukung: Fungsi-fungsi pendukung seperti `isFull`, `isEmpty`, `countQueue`, `clearQueue`, dan `viewQueue` memberikan kemudahan dalam mengelola antrian secara keseluruhan.