# LAPORAN PRAKTIKUM

# MODUL VI
# QUEUE

**Disusun oleh:**
**Muhammad Rifki Fadhilah**
**NIM: 2311102032**

**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

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
2. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

# BAB II

# DASAR TEORI

Queue adalah struktur data yang mirip dengan antrian dalam kehidupan sehari-hari. Elemen-elemen dalam queue dimasukkan pada satu ujung (rear) dan dihapus pada ujung lain (front). Prinsip ini dikenal sebagai FIFO (First-In-First-Out). Queue digunakan untuk memproses data secara berurutan, di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar.

Operasi pada Queue:

1. Enqueue : Menambahkan elemen ke dalam antrian dari belakang
2. Dequeue : Menghapus elemen ke dalam antrian dari depan
3. IsFull : Memeriksa apakah antrian sudah penuh
4. IsEmpty : Memeriksa apakah antrian kosong atau tidak
5. Display : Melihat semua elemen di antrian
6. Clear : Menghapus semua elemen di antrian

Implementasi pada Queue:

1. Menggunakan Array: Elemen-elemen queue disimpan dalam array. Diperlukan variabel untuk menyimpan posisi rear dan front queue.
   Kelemahan: Ukuran array harus ditentukan sebelumnya, dan jika queue penuh, tidak dapat menambahkan elemen baru meskipun ada ruang kosong setelah elemen yang sudah dihapus.

2. Menggunakan Linked List: Elemen-elemen queue disimpan dalam linked list. Diperlukan dua pointer, yaitu pointer front dan rear.
   Kelebihan: Dinamis dalam ukuran, tidak ada pemborosan memori karena hanya digunakan saat diperlukan, dan tidak ada batasan ukuran yang harus ditentukan sebelumnya.

Contoh penerapana Queue:

3. Antrian pada kasir supermarket, bank, atau tiket
4. Penjadwalan proses pada system operasi untuk menentukan urutan eksekusi tugas
5. Buffer pada system komputuer untuk mengatur akses ke sumber daya bersama

# BAB III

## GUIDED

### 1. Guided 1

#### Source code

```
#include<iostream>

using namespace std;

const int maksimalQueue = 5; //Maksimal antrian
int front = 0; //Pennanda 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(){//Antrian 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{// Antriannya 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 untuk menghitung banyak antrian
    return back;
}

void clearQueue(){//Fungsi menghapus semua antrian
    if (isEmpty())
    {
        cout << "Antrian Kosong" <<endl;
```

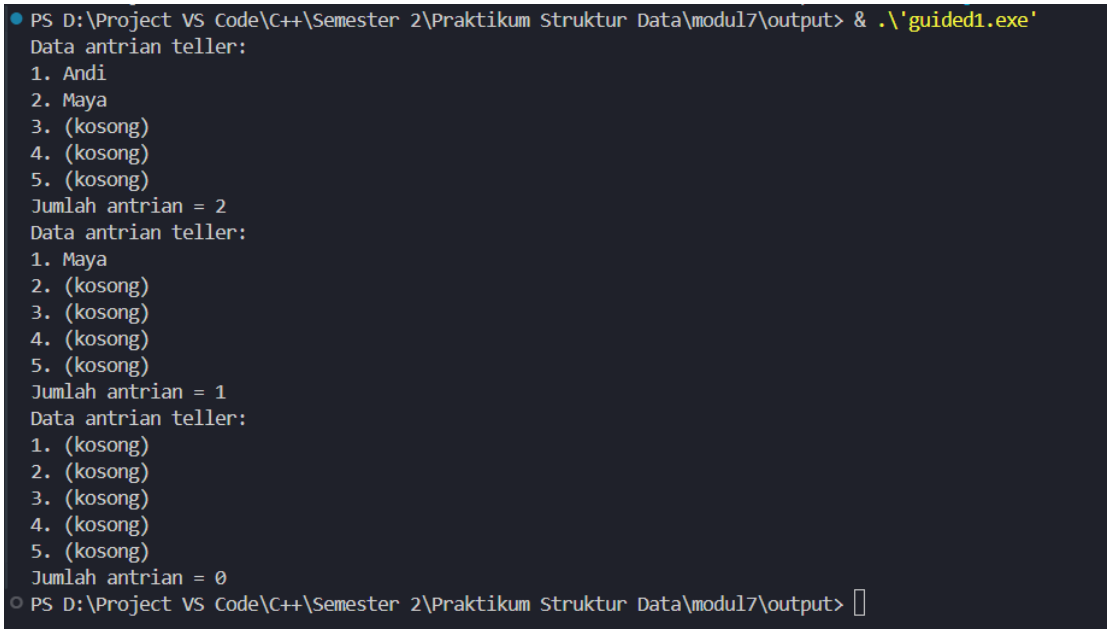
```
    }else{
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

void viewQueue(){
    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;
```

```
}
```

## Screenshoot program




## Deskripsi program

Program diatas adalah implementasi queue dengan array. Pertama mendefinisikan maksimalQueue sebagai ukuran maksimal antrian, serta variabel front dan back sebagai penunjuk posisi depan dan belakang. Array queueTeller digunakan untuk menyimpan elemen-elemen dalam antrian. Fungsi isFull untuk memeriksa apakah antrian penuh atau tidak sedangkan fungsi isEmpty untuk memeriksa apakah antrian kosong atau tidak. Fungsi enqueueAntrian untuk menambahkan elemen ke dalam antrian, jika antrian penuh maka akan mengeluarkan pesan “antrian penuh” dan jika antrian kosong, elemen ditambahkan paling depan/indeks 0. Jika antrian tidak kosong, elemen ditambahkan di belakang elemen yang sebelumnya ditambahkan.Fungsi dequeueAntrian digunakan untuk menghapus antrian paling depan dan menggeser

elemen yang di belakang ke depan.Fungsi countQueue mengembalikan jumlah elemen dalam antrian. Fungsi clearQueue untuk menghapus semua elemen di antrian. Fungsi viewQueue digunakan untuk melihat semua elemen di antrian. Pada fungsi utama yaitu main, ada beberapa operasi queue yang digunakan seperti enqueue,dequeue, countQueue, viewQueue, dan viewQueue.

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

**Source code**

```
#include<iostream>

using namespace std;

struct antrianLL
{
    string nama;
    antrianLL *next;
};

int maksimalAntrianLinkedList = 5;
antrianLL *head, *tail, *cur, *del, *newNode;

int countLinkedList(){
    if (head == NULL)
    {
        return 0;
    }else{
        int banyak = 0;
        cur = head;
        while (cur != NULL)
        {
            cur = cur-> next;
            banyak++;
        }
        return banyak;
    }
}
```

```
//isFull linked list
bool isFullLinkedList(){
    if (countLinkedList() == maksimalAntrianLinkedList)
    {
        return true;
    }else{
        return false;
    }
}

bool isEmptyLinkedList(){
    if (countLinkedList() == 0)
    {
        return true;
    }else {
        return false;
    }

}

void enqueueLinkedList(string nama){
    if (isFullLinkedList())
    {
        cout << "Antrian Penuh" << endl;
    }else{
        if (isEmptyLinkedList())
        {
            head = new antrianLL();
            head->nama = nama;
            head->next = NULL;
            tail = head;
        }else{
            newNode = new antrianLL();
```

```
            newNode->nama = nama;

            newNode->next = NULL;
            tail->next = newNode;
            tail = newNode;
        }
    }
}

void dequeueLinkedList()
{
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" <<endl;
    }else{
        del = head;
        head = head->next;
        del->next = NULL;
        delete del;
    }
}

void displayLinkedList()
{
    cout << "Data Antrian Teller : " << endl;
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" << endl;
    }else
    {
        cout << "Banyak data antrian : " << countLinkedList() <<
endl;
        cur = head;
        int nomor = 1;
```

```
        while (nomor <= maksimalAntrianLinkedList)
        {
            if (cur != NULL)
            {
                cout << nomor << ". " << cur->nama << endl;
                cur = cur->next;
            }else
            {
                cout << nomor << ". " << "(Kosong)" << endl;
            }
            nomor++;
        }
        cout << " " << endl;
    }

}

void clearLinkedList()
{
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" <<endl;
    }else{
        cur = head;
        while (cur != NULL)
        {
            del = cur;
            cur = cur->next;
            del->next =  NULL;
            delete del;
        }
        head = NULL;
        tail = NULL;
```

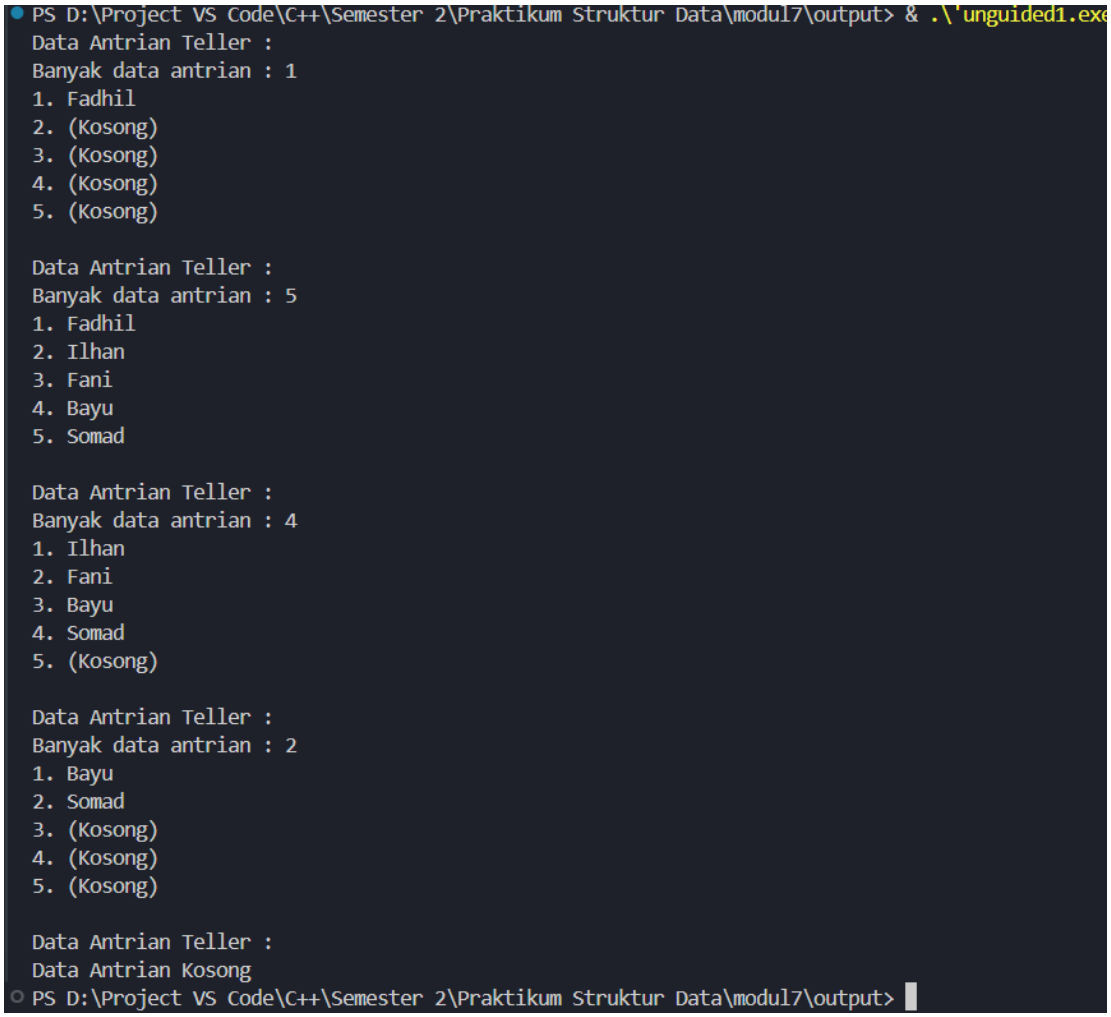
```
    }
}


int main(){
    enqueueLinkedList("Fadhil");
    displayLinkedList();
    enqueueLinkedList("Ilhan");
    enqueueLinkedList("Fani");
    enqueueLinkedList("Bayu");
    enqueueLinkedList("Somad");
    displayLinkedList();

    dequeueLinkedList();
    displayLinkedList();
    dequeueLinkedList();
    dequeueLinkedList();
    displayLinkedList();

    clearLinkedList();
    displayLinkedList();
}
```

**Screenshoot program**




**Deskripsi program**

Program diatas adalah implementasi Queue dengan linked list. Struck antrianLL untuk mendefinisikan node dalam linked list. Setiap node memiliki dua bagian yaitu nama dan next untuk pointer yang menunjuk ke node berikutnya. Variabel head digunakan untuk menunjukkan node pertama sedangkan tail menunjukkan node terakhir dalam linked list. Variabel cur dan del digunakan untuk pointer sementara untuk traversal dan penghapusan node. Fungsi countLinkedList digunakan untuk menghitung jumlah elemen dalam antrian. Fungsi isFullLinkedList dan

isEmptyLinkedList digunakan untuk memeriksa apakah antrian kosong atau tidak. Fungsi enqueueLinkedList digunakan untuk menambahkan elemen ke dalam antrian. Jika antrian kosong node baru akan menjadi head dan tail, jika tidak kosong, node baru akan ditambahkan di belakang tail dan node baru tadi menjadi tail. Fungsi dequeueLinkedList digunakan untuk menghapus elemen pertama dari antrian dan menggeser node berikutnya. Fungsi displayLinkedList untuk menampilkan elemen-elemen dalam antrian.Fungsi clearLinkedList untuk menghapus semua elemen dalam antrian. Dalam fungsi utama atau main, program menambahkan 5 elemen ke atrian dengan menggunakan berbagai operasi yang sudah dijelaskan.

## 2. Unguided 2

### Source code

```
#include<iostream>

using namespace std;

struct antrianLL
{
    string nama;
    long long nim;
    antrianLL *next;
};

int maksimalAntrianLinkedList = 5;
antrianLL *head, *tail, *cur, *del, *newNode;

int countLinkedList(){
    if (head == NULL)
    {
        return 0;
    }else{
        int banyak = 0;
        cur = head;
        while (cur != NULL)
        {
            cur = cur-> next;
            banyak++;
        }
        return banyak;
    }
}


//isFull linked list
```

```
bool isFullLinkedList(){
    if (countLinkedList() == maksimalAntrianLinkedList)
    {
        return true;
    }else{
        return false;
    }
}

bool isEmptyLinkedList(){
    if (countLinkedList() == 0)
    {
        return true;
    }else {
        return false;
    }

}

void enqueueLinkedList(string nama, long long nim){
    if (isFullLinkedList())
    {
        cout << "Antrian Penuh" << endl;
    }else{
        if (isEmptyLinkedList())
        {
            head = new antrianLL();
            head->nama = nama;
            head->nim = nim;
            head->next = NULL;
            tail = head;
        }else{
            newNode = new antrianLL();
            newNode->nama = nama;
```

```
            newNode->nim = nim;

            newNode->next = NULL;
            tail->next = newNode;
            tail = newNode;
        }
    }
}

void dequeueLinkedList()
{
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" <<endl;
    }else{
        del = head;
        head = head->next;
        del->next = NULL;
        delete del;
    }
}

void displayLinkedList()
{
    cout << "Data Antrian : " << endl;
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" << endl;
    }else
    {
        cout << "Banyak data antrian : " << countLinkedList() <<
endl;
        cur = head;
        int nomor = 1;
```

```
        while (nomor <= maksimalAntrianLinkedList)
        {
            if (cur != NULL)
            {
                cout << nomor << ". " << cur->nim << ", " << cur-
>nama << endl;
                cur = cur->next;
            }else
            {
                cout << nomor << ". " << "(Kosong)" << endl;
            }
            nomor++;
        }
        cout << " " << endl;
    }

}

void clearLinkedList()
{
    if (isEmptyLinkedList())
    {
        cout << "Data Antrian Kosong" <<endl;
    }else{
        cur = head;
        while (cur != NULL)
        {
            del = cur;
            cur = cur->next;
            del->next =  NULL;
            delete del;
        }
        head = NULL;
        tail = NULL;
```

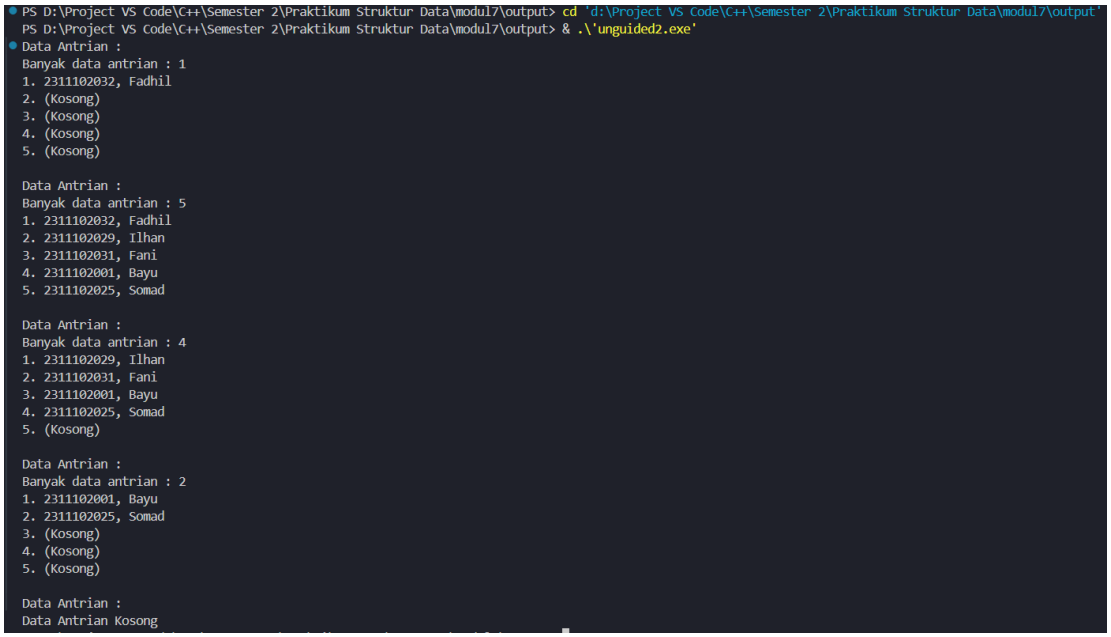
```
    }
}


int main(){
    enqueueLinkedList("Fadhil", 2311102032);
    displayLinkedList();
    enqueueLinkedList("Ilhan", 2311102029);
    enqueueLinkedList("Fani", 2311102031);
    enqueueLinkedList("Bayu", 2311102001);
    enqueueLinkedList("Somad", 2311102025);
    displayLinkedList();

    dequeueLinkedList();
    displayLinkedList();
    dequeueLinkedList();
    dequeueLinkedList();
    displayLinkedList();

    clearLinkedList();
    displayLinkedList();
}
```

**Screenshoot program**




**Deskripsi program**

Program diatas adalah implementasi Queue dengan linked list seperti unguided1. Struck antrianLL untuk mendefinisikan node dalam linked list. Setiap node memiliki tiga bagian yaitu nama,nim untuk menyimpan data antrian dan next untuk pointer yang menunjuk ke node berikutnya. Variabel head digunakan untuk menunjukkan node pertama sedangkan tail menunjukkan node terakhir dalam linked list. Variabel cur dan del digunakan untuk pointer sementara untuk traversal dan penghapusan node. Fungsi countLinkedList digunakan untuk menghitung jumlah elemen dalam antrian. Fungsi isFullLinkedList dan isEmptyLinkedList digunakan untuk memeriksa apakah antrian kosong atau tidak. Fungsi enqueueLinkedList digunakan untuk menambahkan elemen ke dalam antrian. Jika antrian kosong node baru akan menjadi head dan tail, jika tidak kosong, node baru akan ditambahkan di belakang tail dan node baru tadi menjadi tail. Fungsi dequeueLinkedList digunakan untuk menghapus elemen pertama dari antrian dan menggeser node berikutnya. Fungsi displayLinkedList untuk menampilkan elemen-elemen

dalam antrian.Fungsi clearLinkedList untuk menghapus semua elemen dalam antrian. Dalam fungsi utama atau main, program menambahkan 5 elemen ke atrian dengan menggunakan berbagai operasi yang sudah dijelaskan.

# BAB IV

# KESIMPULAN

Queue adalah struktur data yang mirip dengan antrian dalam kehidupan sehari-hari, di mana elemen-elemen dimasukkan pada satu ujung (back) dan dihapus pada ujung lain (front), mengikuti aturan FIFO (First-In-First-Out). Implementasi queue dapat dilakukan menggunakan array atau linked list. Array cocok untuk implementasi queue dengan ukuran tetap, sementara linked list lebih fleksibel karena ukurannya dapat dinamis. Keuntungan queue antara lain memudahkan dalam menangani data secara berurutan, berguna dalam penjadwalan, simulasi, dan pengelolaan antrian seperti pada sistem kasir dan penjadwalan proses pada sistem operasi.