# LAPORAN PRAKTIKUM

# MODUL VII
# QUEUE

**Disusun oleh:**
**Syafanida Khakiki**
**NIM: 2311102005**

**Dosen Pengampu:**
Muhammad Afrizal Amrustian, S. Kom., M. Kom

**PROGRAM STUDI TEKNIK INFORMATIKA**
**FAKULTAS INFORMATIKA**
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**
**PURWOKERTO**
**2023**

# BAB I

# TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

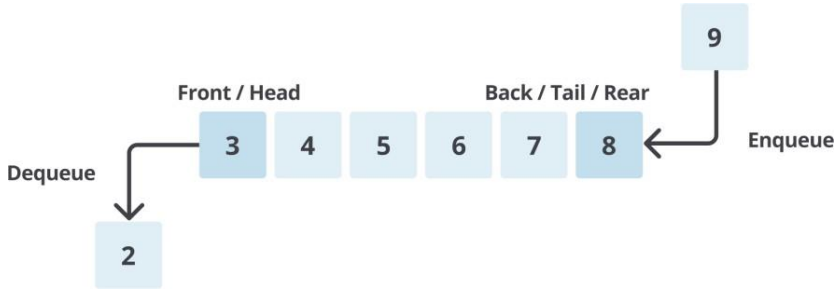
# BAB II

# DASAR TEORI

## a. Pengertian Queue

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep **antrian** pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. **Front/head** adalah pointer ke elemen pertama dalam queue dan **rear/tail/back** adalah pointer ke elemen terakhir dalam queue.




FIRST IN FIRST OUT (FIFO)

Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir diinputkan akan berada paling dengan dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.

Pada Queue, operasi tersebut dilakukan ditempat berbeda (melalui salah satu ujung) karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut **Enqueue** dan **Dequeue** pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

## b. Operasi pada Queue

- enqueue() : menambahkan data ke dalam queue.
- dequeue() : mengeluarkan data dari queue.
- peek() : mengambil data dari queue tanpa menghapusnya.
- isEmpty() : mengecek apakah queue kosong atau tidak.
- isFull() : mengecek apakah queue penuh atau tidak.
- size() : menghitung jumlah elemen dalam queue

# 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;
}
```

**Output :**

```
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
```

**Deskripsi program :**

Program ini mensimulasikan sistem antrian menggunakan queue.

**1. Enqueue (enqueue(string data))**:

- Fungsi ini menambahkan data baru ke dalam antrian.
- Data diwakili oleh string.
- Jika antrian penuh (maksimal 5 data), program akan menampilkan pesan "Antrian penuh".
- Jika antrian kosong, data akan ditempatkan pada posisi pertama dan variabel `front` dan `back` diubah.
- Jika antrian tidak kosong, data akan ditempatkan pada posisi terakhir dan variabel `back` diubah.

**2. Dequeue (dequeue())**:

- Fungsi ini menghapus data terdepan dari antrian.
- Jika antrian kosong, program akan menampilkan pesan "Antrian kosong".
- Jika antrian tidak kosong, data pada posisi pertama akan dihapus dan variabel `back` dikurangi.
- Data pada posisi selanjutnya akan digeser ke depan.

**3. Cek Antrian Penuh (isFull())**:

- Fungsi ini mengembalikan nilai `true` jika antrian penuh dan `false` jika antrian masih kosong.
- Antrian dianggap penuh jika variabel `back` sama dengan nilai maksimal antrian (`maksimalQueue`).

**4. Cek Antrian Kosong (isEmpty())**:

- Fungsi ini mengembalikan nilai `true` jika antrian kosong dan `false` jika antrian masih berisi data.
- Antrian dianggap kosong jika variabel `back` sama dengan 0.

**5. Hitung Jumlah Antrian (count())**:

- Fungsi ini menghitung dan mengembalikan jumlah data yang sedang mengantri.
- Jumlah antrian dihitung berdasarkan nilai variabel `back`.

**6. Hapus Semua Data (clear())**:

- Fungsi ini menghapus semua data dari antrian.
- Jika antrian kosong, program akan menampilkan pesan "Antrian kosong".
- Jika antrian tidak kosong, seluruh data dihapus dan variabel `back` dan `front` diubah menjadi 0.

**7. Tampilkan Data (view())**:

- Fungsi ini menampilkan data yang sedang mengantri beserta nomor urutnya.
- Data yang ditampilkan adalah string.
- Jika antrian kosong, program akan menampilkan pesan "Antrian kosong".

**Contoh Penggunaan:**

Pada contoh program yang diberikan, beberapa operasi antrian diilustrasikan:

1. Menambahkan data "Andi" dan "Maya" ke antrian.
2. Menampilkan data dan menghitung jumlah data.
3. Menghapus data terdepan dari antrian.
4. Menampilkan data dan menghitung jumlah data setelah penghapusan.
5. Menghapus semua data dari antrian.
6. Menampilkan data dan menghitung jumlah data setelah penghapusan semua data.

# BAB IV
# UNGUIDED

1. **Unguided 1**

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

**Source code :**

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

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

class Queue {
private:
    Node* front;
    Node* back;
public:
    Queue() {
        front = NULL;
        back = NULL;
    }

    void enqueueAntrian(string data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = NULL;
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        cout << "Antrian ditambahkan: " << data << endl;
    }

    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
```

```
        }
        Node* temp = front;
        cout << "Antrian dihapus: " << front->data << endl;
        front = front->next;
        delete temp;
    }

    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != NULL) {
            count++;
            current = current->next;
        }
        return count;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian dikosongkan" << endl;
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int position = 1;
        while (current != NULL) {
            cout << position << ". " << current->data << endl;
            current = current->next;
            position++;
        }
    }

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

int main() {
```

```
    Queue antrian;
    antrian.enqueueAntrian("Andi");
    antrian.enqueueAntrian("Maya");
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    antrian.dequeueAntrian();
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    antrian.clearQueue();
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    return 0;
}
```

**Output :**

```
Antrian ditambahkan: Andi
Antrian ditambahkan: Maya
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Antrian dihapus: Andi
Data antrian teller:
1. Maya
Jumlah antrian = 1
Antrian dihapus: Maya
Antrian dikosongkan
Antrian kosong
Jumlah antrian = 0
```

**Deskripsi Program :**

Program ini mensimulasikan sistem antrian sederhana menggunakan struktur data Linked List. Program ini memungkinkan pengelolaan data dalam antrian dengan fungsi-fungsi berikut:

**1. enqueueAntrian (string data)**

- Menambahkan data baru ke bagian belakang antrian.
- Membuat Node baru untuk menyimpan data.
- Memeriksa apakah antrian kosong.
    - Jika kosong, Node baru menjadi elemen pertama (front) dan terakhir (back) dalam antrian.

    - Jika tidak kosong, Node baru ditambahkan di belakang elemen terakhir (back) dan memperbarui back menjadi Node baru.
- Menampilkan pesan konfirmasi bahwa data berhasil ditambahkan ke antrian.

**2. dequeueAntrian ()**

- Menghapus data terdepan dari antrian.
- Memeriksa apakah antrian kosong.
    - Jika kosong, program menampilkan pesan "Antrian kosong".
- Jika tidak kosong, data pada elemen terdepan (front) disimpan sementara dan elemen tersebut dihapus.
- Elemen selanjutnya di antrian menjadi elemen terdepan yang baru (front).
- Membebaskan memori yang digunakan oleh Node yang dihapus.
- Menampilkan pesan konfirmasi bahwa data telah dihapus dari antrian.

**3. countQueue ()**

- Menghitung dan mengembalikan jumlah data yang sedang berada di dalam antrian.
- Melakukan iterasi melalui Linked List dimulai dari elemen terdepan (front) hingga elemen terakhir (penanda NULL).
- Menambah counter untuk setiap Node yang dilewati.
- Mengembalikan nilai counter sebagai jumlah data dalam antrian.

**4. clearQueue ()**

- Menghapus semua data yang ada di dalam antrian.
- Memanfaatkan fungsi dequeueAntrian secara berulang hingga antrian kosong.
- Menampilkan pesan konfirmasi bahwa antrian telah dikosongkan.

**5. viewQueue ()**

- Menampilkan seluruh data yang sedang berada di dalam antrian beserta nomor urutnya.
- Memeriksa apakah antrian kosong.
    - Jika kosong, program menampilkan pesan "Antrian kosong".
- Jika tidak kosong, program menampilkan judul "Data antrian teller".
- Melakukan iterasi melalui Linked List dimulai dari elemen terdepan (front).
- Menampilkan nomor urut dan data pada setiap Node.
- Nomor urut dimulai dari 1 dan incremented pada setiap iterasi.

**6. isEmpty ()**

- Memeriksa apakah antrian dalam keadaan kosong.
- Mengembalikan nilai `true` jika elemen terdepan (front) bernilai NULL (antrian kosong).
- Mengembalikan nilai `false` jika elemen terdepan (front) memiliki nilai (antrian tidak kosong).

**Contoh Penggunaan:**

Program utama mendemonstrasikan penggunaan fungsi-fungsi yang telah dijelaskan.

- Menambahkan data "Andi" dan "Maya" ke dalam antrian.
- Menampilkan isi antrian dan menghitung jumlah data.
- Menghapus data terdepan dari antrian.
- Menampilkan isi antrian dan menghitung jumlah data setelah penghapusan.
- Menghapus semua data dari antrian.
- Mencoba menampilkan isi antrian dan menghitung jumlah data setelah pengosongan (akan menampilkan pesan "Antrian kosong").

2. **Unguided 2**
   Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

**Source Code :**

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

// Struktur untuk merepresentasikan setiap elemen dalam linked list
struct Node {
    string nama;
    string nim;
    Node* next;
};

class Queue {
private:
    Node* front; // Pointer ke depan antrian
    Node* back;  // Pointer ke belakang antrian
public:
    Queue() {
        front = NULL;
        back = NULL;
    }

    // Fungsi menambahkan elemen ke belakang antrian (enqueue)
    void enqueueAntrian(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;
        if (isEmpty()) {
            front = newNode;
            back = newNode;
```

```
        } else {
            back->next = newNode;
            back = newNode;
        }
        cout << "Antrian ditambahkan: Nama: " << nama << ", NIM: " << nim <<
endl;
    }

    // Fungsi menghapus elemen dari depan antrian (dequeue)
    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Antrian dihapus: Nama: " << front->nama << ", NIM: " <<
front->nim << endl;
        front = front->next;
        delete temp;
    }

    // Fungsi menghitung banyak elemen dalam antrian
    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != NULL) {
            count++;
            current = current->next;
        }
        return count;
    }

    // Fungsi menghapus semua elemen dari antrian
    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian dikosongkan" << endl;
    }

    // Fungsi untuk melihat antrian
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
```

```
        }
        cout << "Data antrian mahasiswa:" << endl;
        Node* current = front;
        int position = 1;
        while (current != NULL) {
            cout << position << ". Nama: " << current->nama << ", NIM: " <<
current->nim << endl;
            current = current->next;
            position++;
        }
    }

    // Fungsi untuk memeriksa apakah antrian kosong
    bool isEmpty() {
        return front == NULL;
    }
};

int main() {
    Queue antrian;
    antrian.enqueueAntrian("SYAFANIDA KHAKIKI", "2311102005");
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    antrian.dequeueAntrian();
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    antrian.clearQueue();
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;
    return 0;
}
```

**Output :**

```
Antrian ditambahkan: Nama: SYAFANIDA KHAKIKI, NIM: 2311102005
Data antrian mahasiswa:
1. Nama: SYAFANIDA KHAKIKI, NIM: 2311102005
Jumlah antrian = 1
Antrian dihapus: Nama: SYAFANIDA KHAKIKI, NIM: 2311102005
Antrian kosong
Jumlah antrian = 0
Antrian dikosongkan
Antrian kosong
Jumlah antrian = 0
```

**Deskripsi Program :**

Program ini mensimulasikan sistem antrian sederhana untuk mahasiswa menggunakan struktur data Linked List. Program ini memungkinkan pengelolaan data antrian mahasiswa dengan fungsi-fungsi berikut:

**1. enqueueAntrian (string nama, string nim)**

- Menambahkan data mahasiswa baru ke bagian belakang antrian.
- Membuat Node baru untuk menyimpan data nama dan NIM mahasiswa.
- Memeriksa apakah antrian kosong.
    - Jika kosong, Node baru menjadi elemen pertama (front) dan terakhir (back) dalam antrian.
    - Jika tidak kosong, Node baru ditambahkan di belakang elemen terakhir (back) dan memperbarui back menjadi Node baru.
- Menampilkan pesan konfirmasi bahwa data mahasiswa berhasil ditambahkan ke antrian.

**2. dequeueAntrian ()**

- Menghapus data mahasiswa terdepan dari antrian.
- Memeriksa apakah antrian kosong.
    - Jika kosong, program menampilkan pesan "Antrian kosong".
- Jika tidak kosong, data pada elemen terdepan (front) disimpan sementara dan elemen tersebut dihapus.
- Elemen selanjutnya di antrian menjadi elemen terdepan yang baru (front).
- Membebaskan memori yang digunakan oleh Node yang dihapus.
- Menampilkan pesan konfirmasi bahwa data mahasiswa telah dihapus dari antrian.

**3. countQueue ()**

- Menghitung dan mengembalikan jumlah mahasiswa yang sedang berada di dalam antrian.
- Melakukan iterasi melalui Linked List dimulai dari elemen terdepan (front) hingga elemen terakhir (penanda NULL).
- Menambah counter untuk setiap Node yang dilewati.
- Mengembalikan nilai counter sebagai jumlah mahasiswa dalam antrian.

**4. clearQueue ()**

- Menghapus semua data mahasiswa yang ada di dalam antrian.
- Memanfaatkan fungsi dequeueAntrian secara berulang hingga antrian kosong.
- Menampilkan pesan konfirmasi bahwa antrian telah dikosongkan.

**5. viewQueue ()**

- Menampilkan seluruh data mahasiswa yang sedang berada di dalam antrian beserta nomor urutnya.
- Memeriksa apakah antrian kosong.
    - Jika kosong, program menampilkan pesan "Antrian kosong".
- Jika tidak kosong, program menampilkan judul "Data antrian mahasiswa".
- Melakukan iterasi melalui Linked List dimulai dari elemen terdepan (front).
- Menampilkan nomor urut, nama, dan NIM pada setiap Node.
- Nomor urut dimulai dari 1 dan incremented pada setiap iterasi.

**6. isEmpty ()**

- Memeriksa apakah antrian dalam keadaan kosong.
- Mengembalikan nilai `true` jika elemen terdepan (front) bernilai NULL (antrian kosong).
- Mengembalikan nilai `false` jika elemen terdepan (front) memiliki nilai (antrian tidak kosong).

**Contoh Penggunaan:**

Program utama mendemonstrasikan penggunaan fungsi-fungsi yang telah dijelaskan.

- Menambahkan data mahasiswa "SYAFANIDA KHAKIKI" dengan NIM "2311102005" ke dalam antrian.
- Menampilkan isi antrian dan menghitung jumlah mahasiswa.
- Menghapus data mahasiswa terdepan dari antrian.
- Menampilkan isi antrian dan menghitung jumlah mahasiswa setelah penghapusan.
- Menghapus semua data mahasiswa dari antrian.
- Mencoba menampilkan isi antrian dan menghitung jumlah mahasiswa setelah pengosongan (akan menampilkan pesan "Antrian kosong").

# BAB IV

# KESIMPULAN

**Antrian** (queue) adalah struktur data yang menerapkan prinsip **First In, First Out (FIFO)**, di mana elemen yang pertama kali masuk ke dalam antrian akan menjadi elemen yang pertama kali keluar.

Kesimpulan mengenai Queue (Antrian) berdasarkan Program di atas :

Program diatas dari guided dan unguided merupakan implementasi sederhana dari konsep Queue (Antrian) menggunakan struktur data Linked List.

Konsep antrian didasarkan pada prinsip "First In, First Out" (FIFO), dimana elemen yang pertama kali masuk ke dalam antrian akan menjadi elemen yang pertama kali keluar.

Program diatas menggunakan fungsi / operasi dasar yang umum dilakukan pada antrian:

- **Enqueue (enqueueAntrian):** Menambahkan elemen baru ke bagian belakang antrian.
- **Dequeue (dequeueAntrian):** Menghapus elemen terdepan dari antrian.
- **Count (countQueue):** Menghitung jumlah elemen yang sedang berada di dalam antrian.
- **Clear (clearQueue):** Menghapus semua elemen dari antrian.
- **View (viewQueue):** Menampilkan seluruh elemen yang sedang berada di dalam antrian.
- **IsEmpty (isEmpty):** Memeriksa apakah antrian dalam keadaan kosong.

# DAFTAR PUSTAKA

Asisten Praktikum (2024). Queue. Learning Management System.