

# **LAPORAN PRAKTIKUM**

## **MODUL 7**

### **QUEUE**



**Disusun oleh:**

**Annisa Al Jauhar**

**NIM: 2311102014**

**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. TUJUAN PRAKTIKUM**

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

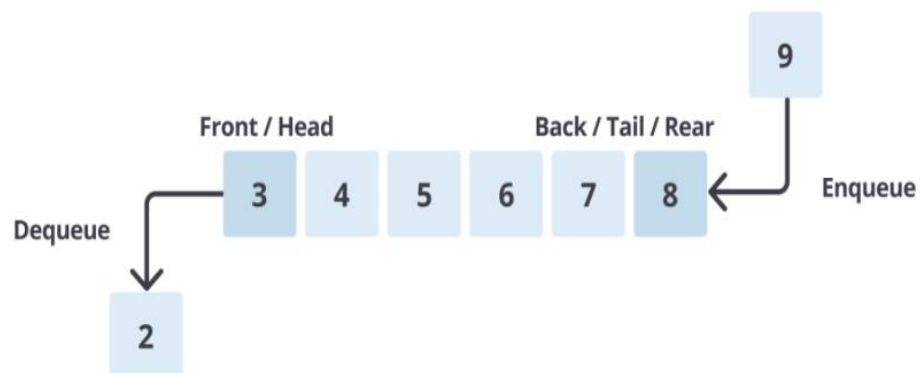
## BAB II

### DASAR TEORI

#### B. DASAR TEORI

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.



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.

#### 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 { // 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 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

```

mester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7\ ; if ($?) { g++ guided-1.cpp -o guided-1 } ; if ($?) { .\guided-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 C:\Users\Yannis\OneDrive\Desktop\Semester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7>

```

## Deskripsi program

Program di atas merupakan implementasi sederhana dari struktur data antrian (queue) menggunakan array statis dengan kapasitas tetap sebesar 5 elemen. Program ini memiliki beberapa fungsi dasar untuk mengelola antrian, termasuk menambah elemen ke antrian (enqueue), menghapus elemen dari antrian (dequeue), memeriksa apakah antrian penuh atau kosong, menghitung jumlah elemen dalam antrian, mengosongkan seluruh elemen dalam antrian, dan menampilkan isi antrian.

Fungsi isFull memeriksa apakah antrian telah mencapai kapasitas maksimal, sedangkan fungsi isEmpty memeriksa apakah antrian kosong. Fungsi enqueueAntrian menambah elemen ke antrian jika masih ada ruang, dan menampilkan pesan jika antrian penuh. Fungsi dequeueAntrian menghapus elemen dari antrian dan menggeser

elemen-elemen yang tersisa ke depan, serta menampilkan pesan jika antrian kosong. Fungsi `countQueue` mengembalikan jumlah elemen dalam antrian. Fungsi `clearQueue` menghapus semua elemen dalam antrian dan mengatur ulang penanda depan dan belakang. Fungsi `viewQueue` menampilkan semua elemen dalam antrian, menunjukkan posisi yang kosong dengan label "(kosong)".



## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### A. Source code

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

using namespace std;

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

class Queue {
private:
    Node* front;
    Node* rear;

public:
    Queue() {
        front = nullptr;
        rear = nullptr;
    }

    bool isFull() {
        return false;
    }

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

```

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

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

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

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

```

```

    }

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

    void view() {
        cout << "Queue contents:" << endl;
        Node* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". " << current->data << endl;
            current = current->next;
            index++;
        }
        if (index == 1) {
            cout << "(empty)" << endl;
        }
    }
};

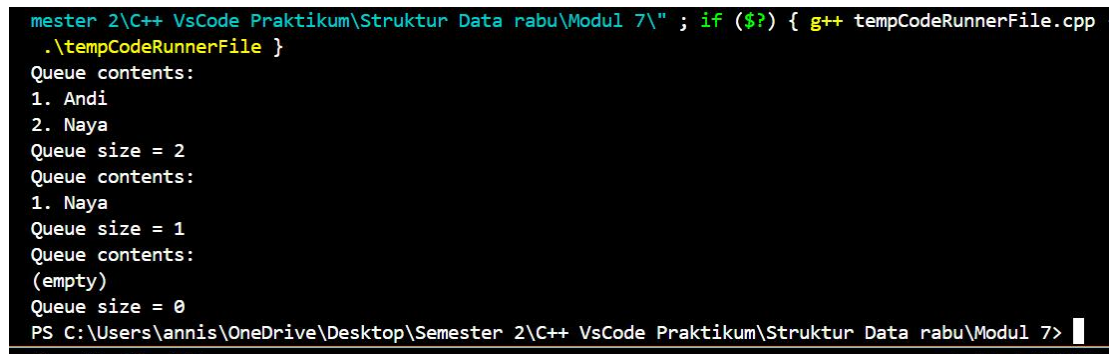
int main() {
    Queue q;
    q.enqueue("Andi");
    q.enqueue("Naya");
    q.view();
    cout << "Queue size = " << q.count() << endl;

    q.dequeue();
    q.view();
    cout << "Queue size = " << q.count() << endl;
}

```

```
q.clear();  
q.view();  
cout << "Queue size = " << q.count() << endl;  
  
return 0;  
}
```

## Screenshoot program



```
mester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7\" ; if ($?) { g++ tempCodeRunnerFile.cpp  
.\tempCodeRunnerFile }  
Queue contents:  
1. Andi  
2. Naya  
Queue size = 2  
Queue contents:  
1. Naya  
Queue size = 1  
Queue contents:  
(empty)  
Queue size = 0  
PS C:\Users\annis\OneDrive\Desktop\Semester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7>
```

## Deskripsi program

Program ini mendefinisikan struktur Node yang menyimpan data string dan pointer ke node berikutnya. Kelas Queue memiliki dua pointer, front dan rear, yang menunjukkan elemen pertama dan terakhir dari antrian. Kelas ini menyediakan beberapa metode utama:

- isFull: Memeriksa apakah antrian penuh. Dalam konteks linked list, antrian tidak pernah penuh, sehingga selalu mengembalikan false.

- isEmpty: Memeriksa apakah antrian kosong dengan memeriksa apakah front adalah nullptr.

-enqueue: Menambah elemen baru ke antrian. Jika antrian kosong, elemen baru menjadi elemen pertama dan terakhir. Jika tidak, elemen baru ditambahkan ke belakang antrian.

-dequeue: Menghapus elemen dari depan antrian. Jika antrian kosong, menampilkan pesan bahwa antrian kosong. Jika tidak, menghapus elemen depan dan mengupdate pointer front.

count: Menghitung jumlah elemen dalam antrian dengan iterasi melalui seluruh elemen dari depan hingga belakang.

-clear: Mengosongkan antrian dengan menghapus semua elemen satu per satu.

-view: Menampilkan semua elemen dalam antrian.

Dalam fungsi main, program menguji fungsionalitas antrian dengan menambah dua elemen ("Andi" dan "Naya"), menampilkan isi antrian dan ukurannya, menghapus satu elemen, menampilkan isi antrian dan ukurannya lagi, mengosongkan antrian, serta menampilkan isi antrian dan ukurannya setelah dikosongkan. Program ini menunjukkan bagaimana menggunakan linked list untuk mengelola antrian dinamis yang dapat menambah dan menghapus elemen tanpa batasan ukuran tetap.

## 2. Unguided 2

### B. Source code

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

using namespace std;

struct Node {
    string name;
    string studentID;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* rear;

public:
    Queue() {
        front = nullptr;
        rear = nullptr;
    }

    bool isFull() {
        return false;
    }

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

    void enqueue(string name, string studentID) {
        Node* newNode = new Node();
```

```

        newNode->name = name;
        newNode->studentID = studentID;
        newNode->next = nullptr;

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

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

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

```

```

    }

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

    void view() {
        cout << "Queue Student Data:" << endl;
        Node* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". Name: " << current->name << ",
NIM: " << current->studentID << endl;
            current = current->next;
            index++;
        }
        if (index == 1) {
            cout << "(empty)" << endl;
        }
    }
};

int main() {
    Queue q;
    q.enqueue("Andi", "2023001");
    q.enqueue("Naya", "2023002");
    q.view();
    cout << "Queue Size = " << q.count() << endl;

    q.dequeue();
    q.view();
    cout << "Queue Size = " << q.count() << endl;
}

```



```

        q.clear();
        q.view();
        cout << "Queue Size = " << q.count() << endl;

        return 0;
    }

```

### Screenshoot program

```

PS C:\Users\annis\OneDrive\Desktop\Semester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7> c
mester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7\ ; if ($?) { g++ tempCodeRunnerFile.cp
.\tempCodeRunnerFile }
Queue Student Data:
1. Name: Andi, NIM: 2023001
2. Name: Naya, NIM: 2023002
Queue Size = 2
Queue Student Data:
1. Name: Naya, NIM: 2023002
Queue Size = 1
Queue Student Data:
(empty)
Queue Size = 0
PS C:\Users\annis\OneDrive\Desktop\Semester 2\C++ VsCode Praktikum\Struktur Data rabu\Modul 7>

```

### Deskripsi program

Struktur Node menyimpan informasi nama dan nomor induk mahasiswa (NIM) serta pointer ke node berikutnya. Kelas Queue memiliki dua pointer, front dan rear, untuk menunjukkan elemen pertama dan terakhir dalam antrian

Dalam fungsi main, program menguji fungsionalitas antrian dengan menambah dua elemen ("Andi" dengan NIM "2023001" dan "Naya" dengan NIM "2023002"), menampilkan isi antrian dan ukurannya, menghapus satu elemen, menampilkan isi antrian dan ukurannya lagi, mengosongkan antrian, serta menampilkan isi antrian dan ukurannya setelah dikosongkan. Program ini

menunjukkan penggunaan linked list untuk mengelola antrian dinamis yang dapat menambah dan menghapus elemen tanpa batasan ukuran tetap.

## **BAB IV**

### **KESIMPULAN**

Kesimpulannya, Queue (antrian) adalah salah satu jenis list linier dalam struktur data yang bekerja dengan prinsip First In First Out (FIFO), di mana elemen pertama yang masuk adalah elemen pertama yang keluar. Data dalam antrian bisa bertipe integer, real, atau record dalam bentuk sederhana atau terstruktur. Penyisipan data dalam queue dilakukan di satu ujung, sementara penghapusan dilakukan di ujung lainnya. Ujung untuk penyisipan disebut rear atau tail, dan ujung untuk penghapusan disebut front atau head.

Sebuah queue dalam program setidaknya harus memiliki tiga variabel: head untuk menandai bagian depan antrian, tail untuk menandai bagian belakang antrian, dan array data untuk menyimpan elemen-elemen yang dimasukkan ke dalam queue tersebut.

Ada beberapa operasi dasar pada queue, yaitu: prosedur create untuk membuat queue baru yang kosong, fungsi isEmpty untuk memeriksa apakah queue kosong, fungsi isFull untuk memeriksa apakah queue penuh, prosedur enqueue untuk menambahkan data ke dalam queue, prosedur dequeue untuk menghapus elemen dari posisi head pada queue, fungsi clear untuk menghapus semua elemen dalam queue, dan prosedur tampil untuk menampilkan elemen-elemen yang ada dalam queue.