

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
PEMROGRAGAMAN**

**MODUL VII
QUEUE**



Disusun oleh :

MOHAMMAD HARITS T

2311102028

IF-11-A

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

- Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
- Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
- Mahasiswa 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.

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.

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

```
#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;
}
```

OUTPUT

```
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS D:\KULIAH\SEMESTER2\struktur data praktikum> █
```

DESKRIPSI PROGRAM

Program ini adalah implementasi sederhana dari struktur data antrian (queue) menggunakan array menggunakan C++. Berikut adalah penjelasan singkat dari masing-masing komponen program: Variabel dan Konstanta. MaksimalQueue menentukan kapasitas antrian maksimum, 5 dalam hal ini. Front dan Back menandai posisi depan dan belakang antrian. QueueTeller menyimpan elemen dalam antrian.

Fungsi Parameter isFull

menunjukkan apakah antrian penuh atau kosong. enqueueAntrian: Menambahkan elemen ke antrian; dequeueAntrian: Menghapus elemen dari antrian; countQueue: Menghitung jumlah elemen dalam antrian; clearQueue: Menghapus semua elemen dari antrian; viewQueue: Menampilkan semua elemen dalam antrian.

Fungsi utama Menambahkan Elemen: Dalam antrian, tambahkan dua elemen, "Andi" dan "Maya". Antrian Menampilkan: Menampilkan elemen dalam antrian; Antrian Menghitung: Menampilkan jumlah elemen dalam antrian. Menghapus Elemen: Anda dapat menyingkirkan satu elemen dari daftar.

BAB IV

Unguided

Unguided 1

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

// Definisi struktur Node
struct Node {
    string data;
    Node* next;
};

// Deklarasi variabel untuk front dan back
Node* front = nullptr;
Node* back = nullptr;

// Fungsi untuk mengecek apakah queue penuh (tidak relevan untuk linked list)
bool isFull() {
    return false; // Linked list tidak terbatas ukuran
}

// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string data) {
    Node* newNode = new Node(); // Membuat node baru
    newNode->data = data; // Mengisi data pada node baru
    newNode->next = nullptr; // Node baru menunjuk ke nullptr

    // Jika queue kosong, set front dan back ke node baru
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode; // Menambahkan node baru di belakang queue
        back = newNode; // Update back ke node baru
    }
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front; // Menyimpan node depan untuk dihapus
        front = front->next; // Memindahkan front ke node berikutnya
        if (front == nullptr) {
            back = nullptr; // Jika queue menjadi kosong, set back ke nullptr
        }
        delete temp; // Menghapus node depan
    }
}
```



```

    }
}

// Fungsi untuk menghitung jumlah elemen dalam queue
int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi untuk menghapus semua elemen dalam queue
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi untuk menampilkan elemen dalam queue
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    while (temp != nullptr) {
        cout << index << ". " << temp->data << endl;
        temp = temp->next;
        index++;
    }
    if (index == 1) {
        cout << "Antrian kosong" << endl;
    }
}

int main() {
    // Menambahkan elemen ke dalam queue
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    // Menghapus elemen dari queue
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    // Menghapus semua elemen dari queue
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    return 0;
}

```

OUTPUT

```
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Antrian kosong
Jumlah antrian = 0
PS D:\tugas kuliah> □
```

DESKRIPSI PROGRAM

Struktur Variabel dan Data Node adalah struktur data elemen antrian. Node memiliki data (menyimpan nilai) dan next (menunjuk ke elemen berikutnya). Front dan back juga merupakan pointer yang menunjuk ke elemen pertama dan terakhir dalam antrian. Fungsi Karena ukuran daftar yang terhubung tidak terbatas, isFull selalu mengembalikan false. isEmpty menentukan apakah antrian kosong dengan memeriksa apakah front adalah nullptr. Menambahkan elemen baru ke antrian. Front dan back diperbarui jika antrian kosong; jika tidak, elemen baru ditambahkan di akhir antrian dan back diperbarui. dequeueAntrian Menghapus elemen pertama. Jika antrian kosong, ditampilkan pesan "Antrian kosong"; jika tidak, elemen pertama dihapus dan depan diperbarui. Jika setelah penghapusan antrian menjadi kosong, kembali diatur ke nullptr. countQueue menghitung dan mengembalikan jumlah elemen dalam antrian menggunakan iterasi dari depan ke belakang ClearQueue: Panggil dequeueQueue terus menerus untuk menghapus semua elemen dalam antrian sampai antrian kosong. Menampilkan semua elemen dalam antrian. Jika ada antrian kosong, akan ditampilkan pesan "Antrian kosong".

Unguided 2

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

// Definisi struktur Node dengan atribut nama mahasiswa dan NIM mahasiswa
struct Node {
    string nama;
    string nim;
    Node *next;
};

// Deklarasi variabel untuk front dan back
Node *front = nullptr;
Node *back = nullptr;

// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string nama, string nim) {
    Node *newNode = new Node(); // Membuat node baru
    newNode->nama = nama; // Mengisi nama mahasiswa pada node baru
    newNode->nim = nim; // Mengisi NIM mahasiswa pada node baru
    newNode->next = nullptr; // Node baru menunjuk ke nullptr

    if (isEmpty()) {
        front = back = newNode; // Jika queue kosong, set front dan back ke node baru
    } else {
        back->next = newNode; // Menambahkan node baru di belakang queue
        back = newNode; // Update back ke node baru
    }
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node *temp = front; // Menyimpan node depan untuk dihapus
        front = front->next; // Memindahkan front ke node berikutnya
        if (front == nullptr) {
            back = nullptr; // Jika queue menjadi kosong, set back ke nullptr
        }
        delete temp; // Menghapus node depan
    }
}
```

```

// Fungsi untuk menghitung jumlah elemen dalam queue
int countQueue() {
    int count = 0;
    Node *temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi untuk menghapus semua elemen dalam queue
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi untuk menampilkan elemen dalam queue
void viewQueue() {
    cout << "Data antrian mahasiswa:" << endl;
    Node *temp = front;
    int index = 1;
    while (temp != nullptr) {
        cout << index << ". Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
        temp = temp->next;
        index++;
    }
    if (index == 1) {
        cout << "Antrian kosong" << endl;
    }
}

int main() {
    enqueueAntrian("harits", "2311102016");
    enqueueAntrian("emel", "231110277");
    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 mahasiswa:
1. Nama: harits, NIM: 2311102016
2. Nama: emel, NIM: 231110277
Jumlah antrian = 2
Data antrian mahasiswa:
1. Nama: emel, NIM: 231110277
Jumlah antrian = 1
Data antrian mahasiswa:
Antrian kosong
Jumlah antrian = 0
PS D:\tugas kuliah> 
```

DESKRIPSI PROGRAM

Program ini mengimplementasikan antrian (queue) menggunakan struktur data linked list di C++. Antrian digunakan untuk menyimpan data mahasiswa dengan atribut nama dan NIM. Struktur Variabel dan Data Node adalah struktur data elemen antrian. Node memiliki data (menyimpan nilai) dan next (menunjuk ke elemen berikutnya). Front dan back juga merupakan pointer yang menunjuk ke elemen pertama dan terakhir dalam antrian.

BAB V

KESIMPULAN

Queue atau antrian adalah struktur data yang mengikuti prinsip FIFO (First In First Out), di mana elemen yang masuk pertama kali akan keluar pertama kali. Dalam C++, queue diimplementasikan sebagai container adapter yang menggunakan objek terenkapsulasi dari kelas container tertentu (seperti deque atau list) sebagai wadah utama. Berikut adalah beberapa kegunaan utama queue:

- Mensimulasikan antrian dunia nyata: Queue bisa digunakan untuk meniru antrian di kehidupan sehari-hari, seperti di kasir, pompa bensin, atau bandara.
- Menyimpan data untuk diproses kemudian: Queue berfungsi untuk menyimpan data yang akan diproses nanti, misalnya input dari pengguna, data yang dihasilkan oleh bagian lain dari program, atau data yang perlu diprioritaskan.
- Mengimplementasikan algoritma: Queue digunakan dalam berbagai algoritma seperti pencarian breadth-first, sorting, dan scheduling.