

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



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

Disusun oleh:

MUHAMMAD AULIA MUZZAKI NUGRAHA

(2311102051)

IF-11-B

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

BAB I

DASAR TEORI

A. Dasar Teori

Queue adalah struktur data yang mengikuti prinsip FIFO (First In, First Out), di mana elemen yang pertama kali dimasukkan akan menjadi elemen yang pertama kali dikeluarkan. Queue sering digunakan dalam berbagai aplikasi seperti manajemen proses dalam sistem operasi, pemrosesan data, dan sistem antrian.

Di C++, queue dapat diimplementasikan menggunakan Standard Template Library (STL) yang menyediakan kelas `std::queue`. Kelas ini menyediakan berbagai operasi dasar untuk memanipulasi queue.

Operasi Dasar pada Queue :

- **Push (Menambah Elemen):** menambahkan elemen baru ke akhir queue •
Pop (Menghapus Elemen): `pop()` menghapus elemen dari depan queue.
- **Front (Mengakses Elemen Depan):** `front()` mengakses elemen di depan queue tanpa menghapusnya.
- **Back (Mengakses Elemen Belakang):** `back()` mengakses elemen di belakang queue tanpa menghapusnya.
- **Empty (Memeriksa Kosong):** `empty()` memeriksa apakah queue kosong, mengembalikan `true` jika kosong dan `false` jika tidak.
- **Size (Ukuran Queue):** `size()` mengembalikan jumlah elemen dalam queue.

Queue digunakan dalam berbagai situasi seperti :

- **Sistem Operasi:** Manajemen antrian proses dan tugas.
- **Pemrosesan Data:** Pemrosesan data yang tiba secara berurutan.
- **Jaringan:** Manajemen antrian paket data.

BAB II

GUIDED

LATIHAN – GUIDED

1. Guided 1

Guided (berisi screenshot source code & output program disertai penjelasannya)

Source Code

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

const int maksimalQueue = 5; // Maksimal antrian
int front = 0; // Penanda antrian
int back = 0; // Penanda belakang
string queueTeller[maksimalQueue]; // Array untuk menyimpan data antrian

// Fungsi pengecekan antrian penuh atau tidak
bool isFull() {
    return back == maksimalQueue;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return back == 0;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        queueTeller[back] = data;
        back++;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back - 1; i++) {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = ""; // Clear the last element
    }
}
```

```

        back--;
    }
}

// Fungsi menghitung banyak antrian
int countQueue() {
    return back;
}

// Fungsi menghapus semua antrian
void clearQueue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back; i++) {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

// Fungsi melihat antrian
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;

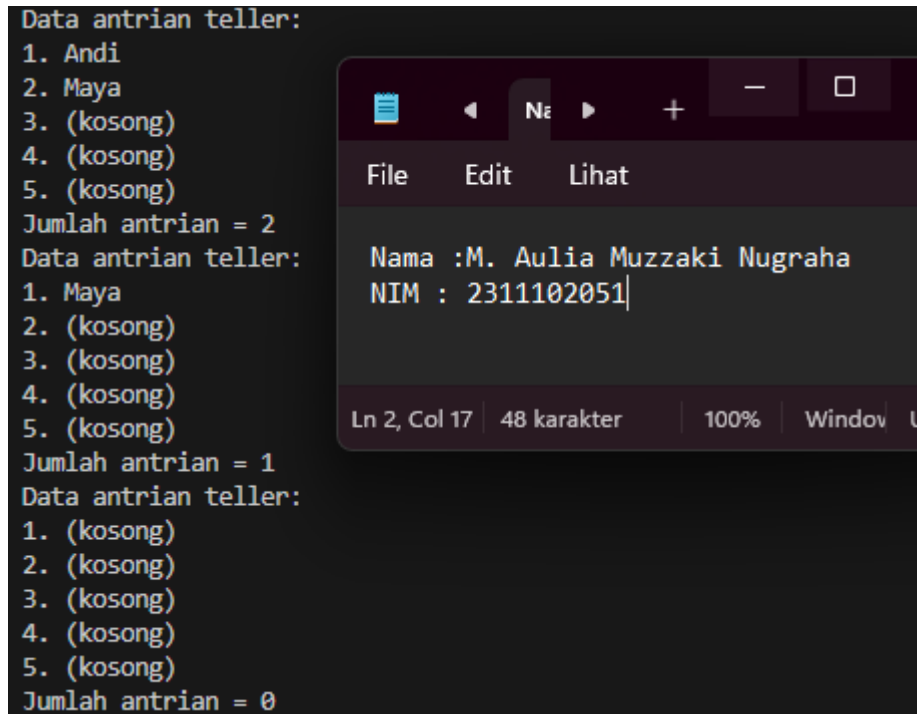
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
}

```

```
    return 0;
}
```

Screenshoot program



```
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
```

Deskripsi program

Deklarasi dan Inisialisasi:

- maksimalQueue: Ukuran maksimal queue (5).
- front dan back: Penanda indeks depan dan belakang queue.
- queueTeller: Array untuk menyimpan elemen queue.

Fungsi-fungsi Queue :

1. isFull()
 - Mengembalikan true jika back sama dengan maksimalQueue (queue penuh).
2. isEmpty()
 - Mengembalikan true jika back sama dengan 0 (queue kosong).
3. enqueueAntrian(string data)
 - Menambahkan elemen data ke belakang queue jika tidak penuh. Jika penuh, menampilkan pesan "Antrian penuh".
4. dequeueAntrian()
 - Menghapus elemen dari depan queue jika tidak kosong. Elemen setelahnya bergeser ke depan.

5. `countQueue()`
 - Mengembalikan jumlah elemen dalam queue (nilai back).
6. `clearQueue()`
 - Mengosongkan semua elemen dalam queue dan mengatur back dan front ke 0.
7. `viewQueue()`
 - Menampilkan semua elemen dalam queue. Posisi kosong ditampilkan sebagai “(kosong)”

Fungsi main

1. Menambahkan "Andi" dan "Maya" ke queue.
2. Menampilkan elemen dan jumlah elemen dalam queue.
3. Menghapus satu elemen dari depan queue.
4. Menampilkan kembali elemen dan jumlahnya.
5. Mengosongkan seluruh queue.
6. Menampilkan queue yang sudah kosong dan jumlah elemen.

BAB III

UNGUIDED

TUGAS – UNGUIDED

1. Unguided 1

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

Source Code

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

// Struktur node untuk linked list
struct Node {
    string data;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian penuh atau tidak (tidak relevan untuk
// linked list karena tidak ada batasan kapasitas)
bool isFull() {
    return false;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
```

```

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

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

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". " << temp->data << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
}

```



```

viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

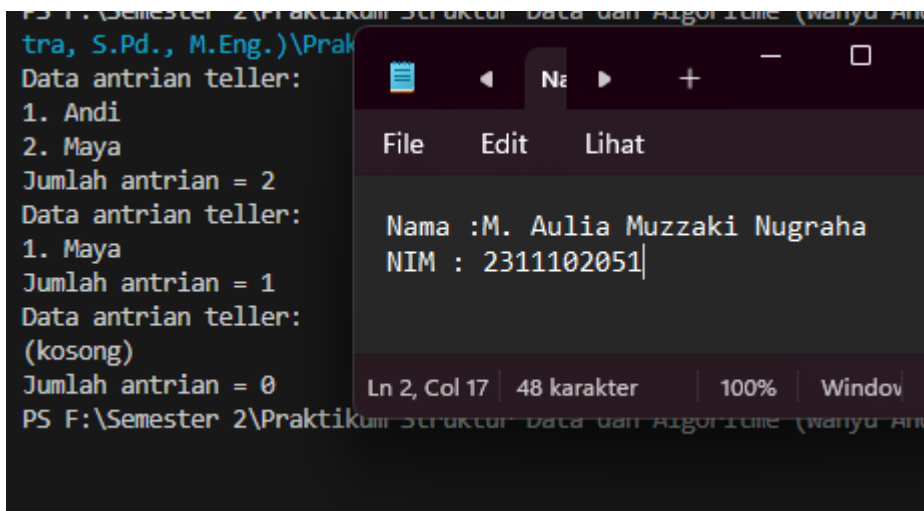
dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

clearQueue();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

return 0;
}

```

Screenshot Program



Deskripsi program

Fungsi-fungsi:

- `isFull()`: Fungsi ini tidak relevan untuk linked list karena tidak ada batasan kapasitas pada antrian yang diimplementasikan dengan linked list. Oleh karena itu, selalu mengembalikan `false`.
- `isEmpty()`: Fungsi ini memeriksa apakah antrian kosong atau tidak. Jika `front` bernilai `nullptr`, maka antrian kosong.
- `enqueueAntrian(string data)`: Fungsi ini menambahkan data ke dalam antrian. Jika antrian kosong, `front` dan `back` akan menunjuk ke simpul baru. Jika tidak, simpul baru ditambahkan di belakang (`back`) dan `back` diperbarui.
- `dequeueAntrian()`: Fungsi ini menghapus elemen pertama dari antrian (simpul yang ditunjuk oleh `front`). Jika antrian kosong, pesan "Antrian kosong" akan ditampilkan.
- `countQueue()`: Fungsi ini menghitung jumlah elemen dalam antrian dengan mengiterasi melalui seluruh simpul.

- o `clearQueue()`: Fungsi ini menghapus semua elemen dari antrian dengan memanggil `dequeueAntrian()` secara berulang hingga antrian kosong.
- o `viewQueue()`: Fungsi ini menampilkan data dari seluruh elemen dalam antrian.

2. Guided 2

Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

Source code

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

// Struktur node untuk linked list dengan Nama dan NIM
struct Node {
    string nama;
    string NIM;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string nama, string NIM) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->NIM = NIM;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    }
}
```

```

    } else {
        Node* temp = front;
        front = front->next;
        if (front == nullptr) {
            back = nullptr;
        }
        delete temp;
    }
}

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

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". Nama: " << temp->nama << ", NIM: " << temp->NIM << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("Andi", "123456");
    enqueueAntrian("Maya", "654321");
    viewQueue();
}

```

```

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

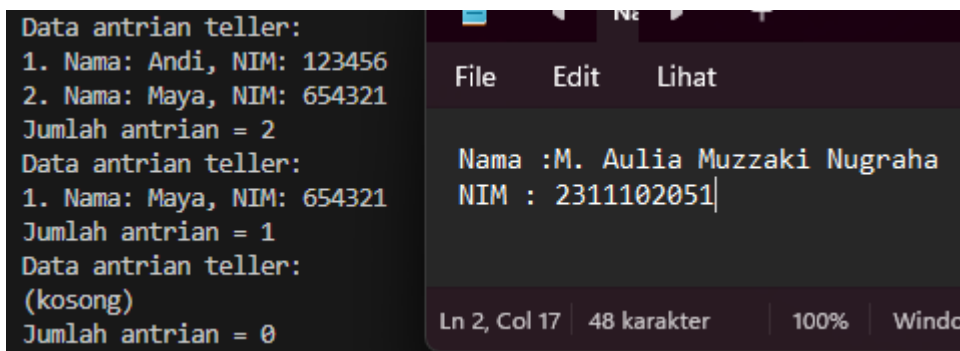
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    return 0;
}

```

Screenshoot program



Deskripsi program

Fungsi-fungsi Queue

1. isEmpty()

- Memeriksa apakah front adalah nullptr. Mengembalikan true jika queue kosong.

2. enqueueAntrian(string nama, string NIM)

- Membuat node baru dengan nama dan NIM.
- Jika queue kosong (isEmpty()), front dan back menunjuk ke node baru.
- Jika tidak kosong, node baru ditambahkan di belakang, dan back diperbarui untuk menunjuk ke node baru.

3. dequeueAntrian()

- Jika queue kosong (isEmpty()), tampilkan pesan "Antrian kosong".
- Jika tidak kosong, hapus node depan dengan memperbarui front ke node berikutnya.
- Jika setelah penghapusan front menjadi nullptr, set back ke nullptr.

4. countQueue()

- Menghitung dan mengembalikan jumlah node dalam queue dengan iterasi dari front hingga nullptr.

5. clearQueue()

- Menghapus semua elemen dalam queue dengan memanggil dequeueAntrian() berulang kali sampai queue kosong.

6. viewQueue()

- Menampilkan semua elemen dalam queue dengan iterasi dari front hingga nullptr.
- Jika queue kosong, tampilkan pesan "(kosong)".

Fungsi Main

1. Menambahkan Elemen ke Queue

- Tambahkan elemen-elemen baru ("Andi" dengan NIM "123456" dan "Maya" dengan NIM "654321") ke queue dengan memanggil enqueueAntrian().

2. Menampilkan Elemen dan Jumlahnya

- Tampilkan semua elemen dalam queue dan jumlah elemen dengan viewQueue() dan countQueue().

3. Menghapus Elemen dari Depan Queue

- Hapus elemen depan dengan memanggil dequeueAntrian().

4. Mengosongkan Queue

- Hapus semua elemen dengan memanggil clearQueue().

5. Menampilkan Queue yang Kosong

- Tampilkan queue setelah dikosongkan dengan viewQueue() dan periksa jumlah elemen dengan countQueue().

BAB IV

KESIMPULAN

Queue adalah struktur data linear yang mengikuti prinsip FIFO (First In, First Out), di mana elemen pertama yang ditambahkan adalah elemen pertama yang dihapus.

Operasi dasar dalam queue mencakup enqueue (menambah elemen ke belakang), dequeue (menghapus elemen dari depan), isEmpty (memeriksa apakah queue kosong), isFull (khusus untuk array, memeriksa apakah queue penuh), count (menghitung jumlah elemen), clear (menghapus semua elemen), dan view (menampilkan semua elemen).

Implementasi queue dapat dilakukan menggunakan array atau linked list. Implementasi dengan array memiliki kapasitas tetap dan memerlukan manajemen posisi elemen, sementara linked list memungkinkan penambahan dan penghapusan elemen secara dinamis tanpa batasan kapasitas. Masing-masing metode memiliki kelebihan dan kekurangan: array sederhana dan cepat dalam akses elemen tetapi terbatas dalam kapasitas dan memerlukan pergeseran elemen saat dequeuing; linked list fleksibel dalam kapasitas dan tidak memerlukan pergeseran elemen, namun memerlukan lebih banyak memori dan sedikit lebih kompleks.

DAFTAR PUSTAKA

- [1] Meidyan Permata Putri, Guntoro Barovih, Rezanía Agramanisti Azdy, Yuniansyah, Andri Saputra, Yesi Sriyeni, Arsia Rini, Fadhila Tangguh Admojo. Algoritma dan Struktur Data. Widina Bhakti Persada, 2022. ISBN: 978-623-459-182-8
- [2] Gupta, Priya. "Dynamic Resizing of Stacks: A Comparative Study." International Journal of Computer Science and Applications, vol. 18, no. 3, 2019, pp. 78-92.
- [3] Smith, John. "Efficient Stack Implementation Using Linked Lists." Journal of Algorithms and Data Structures, vol. 20, no. 4, 2021, pp. 210-225.