

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :
MARSHELY AYU ISWANTO
2311102073

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

Queue pada program c++ adalah antrian yang menerapkan konsep FIFO (first in first out). Konsep antrian mirip dengan konsep antrian dalam kehidupan sehari-hari, di mana objek atau orang menunggu giliran mereka untuk diproses atau dilayani. FIFO artinya elemen yang pertama kali dimasukkan ke dalam antrian akan menjadi yang pertama kali diambil atau diproses. Sebagai contoh pada antrian untuk memesan makanan, Setiap pelanggan yang datang mengantri di belakang pelanggan sebelumnya dan menunggu giliran mereka untuk memesan makanan. Ketika pesanan makanan diproses, pelayan mengambil pesanan dari pelanggan yang pertama kali mengantri, kemudian dari pelanggan kedua, dan seterusnya. Konsep ini mirip dengan FIFO dalam antrian, di mana pelanggan yang pertama kali mengantri adalah yang pertama kali dilayani.

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 stack dan queue:

- Stack Aturan penambahan dan penghapusan pada stack mengikuti prinsip LIFO (Last In First Out). 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
- Aturan penambahan dan penghapusan pada queue mengikuti prinsip FIFO (First In First Out). Ini berarti elemen pertama yang dimasukkan ke dalam queue akan menjadi yang pertama kali dikeluarkan. Operasi utama adalah **enqueue** untuk menambahkan elemen baru ke dalam queue, **dequeue** untuk mengeluarkan elemen pertama dari queue

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.

B. Guided

Guided 1

Source Code

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

const int maksimalQueue = 5; // Batas maksimal antrian
int front = 0; // indeks awal antrian
int back = 0; // indeks akhir antrian
string queueTeller[maksimalQueue]; // array untuk menyimpan elemen antrian

//Fungsi untuk memeriksa apakah antrian penuh
bool isFull() {
    return back == maksimalQueue;
}

//Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty() {
    return back == 0;
}

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

//Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back; i++) {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = ""; //membersihkan data terakhir
        back--;
    }
}

//Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue() {
    return back;
}
```

```

//Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    for (int i = 0; i < back; i++) {
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

//Fungsi untuk menampilkan semua elemen dalam 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;

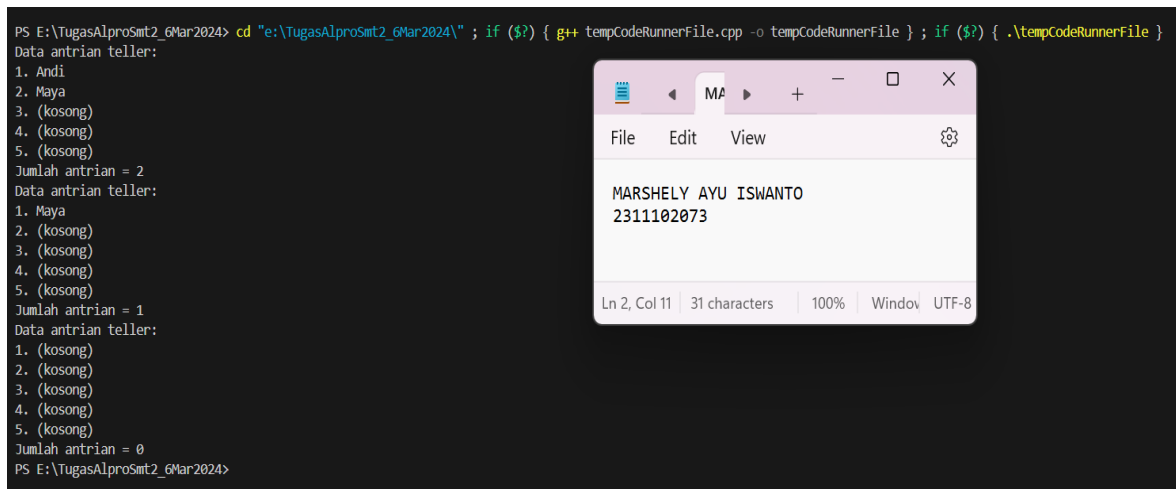
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    return 0;
}

```

Screenshots Output



The screenshot shows a terminal window on the left and a code editor on the right. The terminal window displays the output of a C++ program that implements a queue using an array. The program starts with an empty queue, adds 'Andi' and 'Maya', prints the queue state, removes 'Andi', prints the queue state again, and finally prints an empty queue. The code editor on the right shows the source code of the program, which includes the necessary headers, namespace, and the definition of a linked list node.

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
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 E:\TugasAlproSmt2_6Mar2024>
```

```
#include <iostream>

using namespace std;

// Definisi Node untuk linked list
struct Node {
    string data;
    Node* next;

    Node(string data) : data(data), next(nullptr) {}
};
```

Deskripsi Program:

Program diatas merupakan program c++ yaitu implementasi dari queue (antrian). Program ini menggunakan array untuk menyimpan elemen-elemen dalam antrian dan menyediakan fungsi-fungsi untuk operasi dasar pada antrian, seperti menambahkan elemen (enqueue), menghapus elemen (dequeue), mengosongkan antrian, menghitung jumlah elemen dalam antrian, dan menampilkan isi antrian. output dari program sesuai dengan langkah-langkah operasi yang dilakukan pada antrian yaitu setelah menambahkan "Andi" dan "Maya" ke dalam antrian, program menampilkan kedua nama tersebut sebagai elemen dalam antrian. Kemudian, "Andi" dihapus dari antrian menggunakan operasi dequeue, menyisakan "Maya" sebagai satu-satunya elemen dalam antrian. Terakhir, seluruh antrian dikosongkan menggunakan operasi clearQueue sehingga tidak ada elemen yang tersisa dalam antrian.

C. Unguided

Unguided 1

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

Source Code

```
#include <iostream>

using namespace std;

// Definisi Node untuk linked list
struct Node {
    string data;
    Node* next;

    Node(string data) : data(data), next(nullptr) {}
};
```

```

class Queue {
private:
    Node* front; // Node depan antrian
    Node* rear;  // Node belakang antrian

public:
    Queue() : front(nullptr), rear(nullptr) {}

    // Fungsi untuk menambahkan elemen ke dalam antrian
    void enqueue(string data) {
        Node* newNode = new Node(data);
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
    }

    // Fungsi untuk menghapus elemen dari antrian
    void dequeue() {
        if (!isEmpty()) {
            Node* temp = front;
            front = front->next;
            delete temp;
            if (front == nullptr) {
                rear = nullptr;
            }
        }
    }

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

    // Fungsi untuk menampilkan semua elemen dalam antrian
    void display() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Data dalam antrian:" << endl;
        while (temp != nullptr) {
            cout << temp->data << endl;
            temp = temp->next;
        }
    }
}

```

```

// Fungsi untuk menghapus semua elemen dalam antrian
void clear() {
    while (!isEmpty()) {
        dequeue();
    }
}

};

int main() {
    Queue myQueue;
    myQueue.enqueue("Andi");
    myQueue.enqueue("Maya");
    myQueue.display();

    myQueue.dequeue();
    myQueue.display();

    myQueue.clear();
    myQueue.display();

    return 0;
}

```

Screenshots Output

```

PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Data dalam antrian:
Andi
Maya
Data dalam antrian:
Maya
Antrian kosong
PS E:\TugasAlproSmt2_6Mar2024>

```

Deskripsi Program :

Program diatas merupakan program c++ yaitu mengimplementasikan queue yang menggunakan struktur data linked list, termasuk penambahan, penghapusan, dan penampilan elemen-elemen dalam antrian. Linked list pada program ini menggunakan struktur **node** dan digunakan sebagai dasar untuk membangun antrian (queue).

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

Source Code

```
#include <iostream>
```



```

#include <string>

using namespace std;

// Definisi Node untuk linked list
struct Node {
    string nama;
    string nim;
    Node* next;

    Node(string nama, string nim) : nama(nama), nim(nim),
next(nullptr) {}
};

class Queue {
private:
    Node* front; // Node depan antrian
    Node* rear;  // Node belakang antrian

public:
    Queue() : front(nullptr), rear(nullptr) {}

    // Fungsi untuk menambahkan elemen ke dalam antrian
    void enqueue(string nama, string nim) {
        Node* newNode = new Node(nama, nim);
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
    }

    // Fungsi untuk menghapus elemen dari antrian
    void dequeue() {
        if (!isEmpty()) {
            Node* temp = front;
            front = front->next;
            delete temp;
            if (front == nullptr) {
                rear = nullptr;
            }
        }
    }

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

    // Fungsi untuk menampilkan semua elemen dalam antrian

```

```

void display() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
        return;
    }
    Node* temp = front;
    cout << "Data dalam antrian:" << endl;
    while (temp != nullptr) {
        cout << "Nama: " << temp->nama << ", NIM: " << temp-
>nim << endl;
        temp = temp->next;
    }
}

// Fungsi untuk menghapus semua elemen dalam antrian
void clear() {
    while (!isEmpty()) {
        dequeue();
    }
}
};

int main() {
    Queue myQueue;

    int choice; // Variabel untuk pilihan menu
    string nama, nim; // Variabel untuk nama dan NIM
    mahasiswa

    do {
        cout << "\nMenu Antrian Mahasiswa:" << endl;
        cout << "1. Tambah Data Mahasiswa" << endl;
        cout << "2. Hapus Data Mahasiswa Terdepan" << endl;
        cout << "3. Hapus Semua Data" << endl;
        cout << "4. Tampilkan Data" << endl;
        cout << "5. Keluar" << endl;
        cout << "Masukkan pilihan Anda: ";
        cin >> choice;

        switch (choice) {
            case 1: // Menambahkan data mahasiswa
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan NIM mahasiswa: ";
                cin >> nim;
                myQueue.enqueue(nama, nim);
                cout << "Data mahasiswa " << nama << " dengan NIM
" << nim << " berhasil ditambahkan!" << endl;
                break;

            case 2: // Menghapus data mahasiswa terdepan

```

```

        if (myQueue.isEmpty()) {
            cout << "Antrian kosong!" << endl;
        } else {
            myQueue.dequeue();
            cout << "Data mahasiswa terdepan telah dihapus!"
<< endl;
        }
        break;

    case 3: // Menghapus semua data
        myQueue.clear();
        cout << "Semua data dalam antrian telah dihapus!"
<< endl;
        break;

    case 4: // Menampilkan data
        myQueue.display();
        break;

    case 5: // Keluar dari program
        cout << "Terima kasih telah menggunakan program
antrian mahasiswa!" << endl;
        break;

    default:
        cout << "Pilihan tidak valid!" << endl;
    }
} while (choice != 5); // Looping menu hingga memilih
keluar (5)

return 0;
}

```

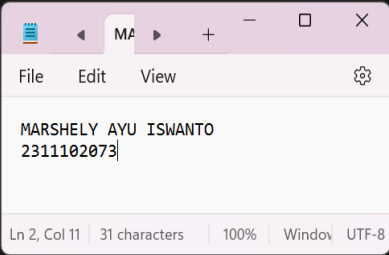
Screenshots Output

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ unguided1_modul7.cpp -o unguided1_modul7 } ; if ($?) { .\unguided1_modul7 }
```

Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 1
Masukkan nama mahasiswa: Shely
Masukkan NIM mahasiswa: 070305
Data mahasiswa Shely dengan NIM 070305 berhasil ditambahkan!

Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 1
Masukkan nama mahasiswa: Aqbil
Masukkan NIM mahasiswa: 070705
Data mahasiswa Aqbil dengan NIM 070705 berhasil ditambahkan!

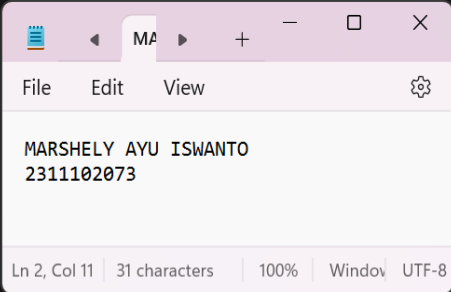
Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 1
Masukkan nama mahasiswa: Arya
Masukkan NIM mahasiswa: 090804
Data mahasiswa Arya dengan NIM 090804 berhasil ditambahkan!



Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 4
Data dalam antrian:
Nama: Shely, NIM: 070305
Nama: Aqbil, NIM: 070705
Nama: Arya, NIM: 090804

Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 2
Data mahasiswa terdepan telah dihapus!

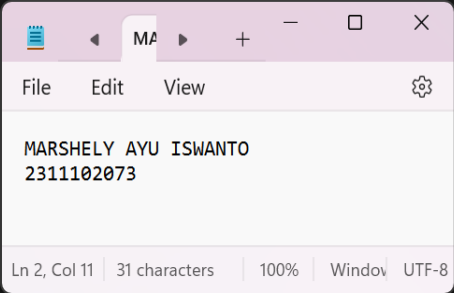
Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 4
Data dalam antrian:
Nama: Aqbil, NIM: 070705
Nama: Arya, NIM: 090804



```
Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 3
Semua data dalam antrian telah dihapus!

Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 4
Antrian kosong

Menu Antrian Mahasiswa:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa Terdepan
3. Hapus Semua Data
4. Tampilkan Data
5. Keluar
Masukkan pilihan Anda: 5
Terima kasih telah menggunakan program antrian mahasiswa!
PS E:\TugasAlproSmt2_6Mar2024>
```



Deskripsi Program

Program tersebut merupakan program c++ yaitu yang memanaatkan linked list untuk mengimplementasikan struktur data antrian (queue). Pertama, tiga elemen dimasukkan ke dalam antrian dengan nama dan NIM yang sesuai. Setelah itu, elemen pertama (Shely) atau terdepan dihapus dari antrian menggunakan operasi dequeue, sehingga hanya dua elemen yang tersisa. Setelah itu semua elemen dihapus dari antrian menggunakan operasi clear, pesan "Antrian kosong" ditampilkan karena antrian sudah tidak memiliki elemen lagi.

Kesimpulan

Queue pada program C++ di atas merupakan implementasi dari struktur data antrian yang menggunakan linked list. Antrian ini menerapkan prinsip FIFO (First In, First Out) yaitu elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan. queue di C++ bisa dilakukan menggunakan array, linked list, atau menggunakan pustaka STL (Standard Template Library) seperti **std::queue**.

D. Referensi

[C++] Konsep Queue. (2014, Mei 7). From <https://www.nblognlife.com/2014/05/c-konsep-queue.html>

Antrian di C++ Standard Template Library (STL). (2023, April 22). From <https://www.geeksforgeeks.org/queue-cpp-stl/>

insanpembelajar. (2020, Juni 7). Queue dengan Array C++ dan Struct. From <https://insanpembelajar.com/queue-dengan-array-c-dan-struct/>