

LAPORAN PRAKTIKUM

MODUL VII

QUEUE



Disusun Oleh:

Muhammad Raafi Al Hafiidh

2311102070

Dosen:

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

- Mahasiswa mampu menjelaskan definisi dan konsep dari Double Queue.
- Mahasiswa mampu menerapkan operasi tambah dan menghapus pada Queue.
- Mahasiswa mampu menerapkan operasi tampil data pada Queue

BAB II

DASAR TEORI

Queue atau antrian dalam C++ adalah struktur data linier yang digunakan untuk menyimpan dan mengambil data dalam urutan tertentu. Queue mengikuti prinsip **FIFO** (First In, First Out), di mana data yang dimasukkan terlebih dahulu akan dikeluarkan terlebih dahulu.

Karakteristik Queue:

- **FIFO:** Data yang dimasukkan terlebih dahulu akan dikeluarkan terlebih dahulu.
- **Linear:** Data disusun secara linier, dengan satu elemen di depan dan satu elemen di belakang.
- **Dinamis:** Queue dapat bertambah dan berkurang panjangnya saat data ditambahkan atau dihapus.

Penggunaan Queue:

- Simulasi antrian, seperti antrian kasir di supermarket.
- Buffer data, seperti penyimpanan data sementara sebelum diproses.
- Algoritma pencarian dan pencocokan pola.
- Sistem multitasking, seperti penjadwalan proses dalam sistem operasi.

Operasi Dasar Queue:

- **push(data):** Menambahkan data ke bagian belakang queue.
- **pop():** Menghapus data dari bagian depan queue.
- **front():** Mengembalikan data di bagian depan queue tanpa menghapusnya.
- **size():** Mengembalikan jumlah elemen dalam queue.
- **empty():** Memeriksa apakah queue kosong.

Implementasi Queue:

C++ menyediakan kelas **std::queue** dalam Standard Template Library (STL) untuk implementasi queue. Kelas ini menggunakan container adaptor untuk mengelola data internal queue.

Contoh Penggunaan Queue:

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
    // Deklarasi queue integer
    queue<int> myQueue;

    // Menambahkan elemen ke queue
    myQueue.push(1);
    myQueue.push(2);
    myQueue.push(3);

    // Mengambil elemen dari queue
    while (!myQueue.empty()) {
        int value = myQueue.front();
        cout << value << " ";
        myQueue.pop();
    }

    return 0;
}
```

BAB III

LATIHAN DAN TUGAS

A. GUIDED

1. Guided

Source Code

```
#include <iostream>
using namespace std;
// queue array
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;
    }
}
// fungsi pengecekan
bool isEmpty()
{ // antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
// fungsi menambahkan antrian
void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "antrian penuh" << endl;
    }
    else
    { // nested if, nested for
        if (isEmpty())
        { // kondisi ketika queue kosong
            queueTeller[0] = data;
```

```

        front++; // front = front +1;
        back++;
    }
    else
    {
        // antrianya ada
isi
        queueTeller[back] = data; //
queueTeller[1]=data
        back++; // back=back+1; 2
    }
}
// fungsi mengurangi antrian
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        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");
    enqueueAntrian("Budi");
    viewQueue();
    cout << "jumlah antrian = " << countQueue() << endl;

    dequeueAntrian();
    viewQueue();

    enqueueAntrian("Ucup");
    enqueueAntrian("Umar");
    viewQueue();

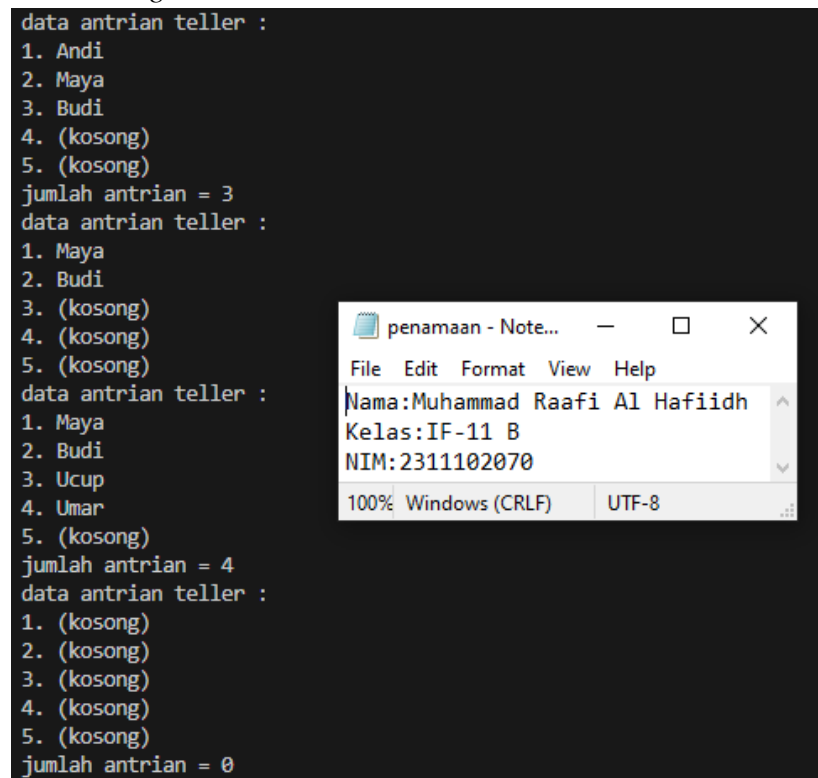
    cout << "jumlah antrian = " << countQueue() << endl;

    clearQueue();
    viewQueue();

    cout << "jumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Screenshoot Program



```
data antrian teller :
1. Andi
2. Maya
3. Budi
4. (kosong)
5. (kosong)
jumlah antrian = 3
data antrian teller :
1. Maya
2. Budi
3. (kosong)
4. (kosong)
5. (kosong)
data antrian teller :
1. Maya
2. Budi
3. Ucup
4. Umar
5. (kosong)
jumlah antrian = 4
data antrian teller :
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
jumlah antrian = 0
```

Deskripsi Program

Kode di atas menggunakan queue pada proses fungsinya, pertama program akan menentukan batas maksimal antrian menggunakan variabel `maksimalQueue`, lalu melakukan pengecekan apakah antrian tersebut kosong menggunakan `isEmpty()` atau penuh menggunakan `isFull()`, apabila pengecekan program mengeluarkan hasil output dari selain `isFull()` maka program bisa melakukan `enqueueAntrian()` untuk menambahkan data pada antrian yang ada, dan untuk mengurangi data yang ada pengguna bisa menggunakan `dequeueAntrian()`. Untuk menghitung jumlah banyaknya antrian bisa menggunakan `countQueue()`, lalu untuk menghapus semua antrian bisa menggunakan `clearQueue()`. Yang terakhir untuk melihat semua data yang ada bisa menggunakan `viewQueue()`.

B. 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 *rear;

public:
    Queue()
    {
        front = nullptr;
        rear = nullptr;
    }
    bool isEmpty()
    {
        return (front == nullptr);
    }
    void enqueue(string data)
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->next = nullptr;
        if (isEmpty())
        {
            front = newNode;
            rear = newNode;
        }
        else
        {
            rear->next = newNode;
            rear = newNode;
        }
    }
    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *temp = front;
            front = front->next;
            delete temp;
        }
    }
}

```

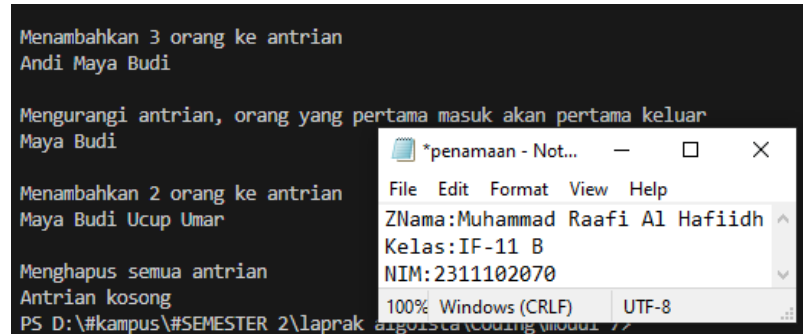
```

    }
    void viewQueue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *current = front;
            while (current != nullptr)
            {
                cout << current->data << " ";
                current = current->next;
            }
            cout << endl;
        }
    }
};

int main()
{
    Queue queue;
    cout << "\nMenambahkan 3 orang ke antrian" << endl;
    queue.enqueue("Andi");
    queue.enqueue("Maya");
    queue.enqueue("Budi");
    queue.viewQueue();
    cout << "\nMengurangi antrian, orang yang pertama
masuk akan pertama keluar " << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "\nMenambahkan 2 orang ke antrian" << endl;
    queue.enqueue("Ucup");
    queue.enqueue("Umar");
    queue.viewQueue();
    cout << "\nMenghapus semua antrian" << endl;
    while (!queue.isEmpty())
    {
        queue.dequeue();
    }
    queue.viewQueue();
    return 0;
}

```

Screenshoot Program



```
Menambahkan 3 orang ke antrian
Andi Maya Budi

Mengurangi antrian, orang yang pertama masuk akan pertama keluar
Maya Budi

Menambahkan 2 orang ke antrian
Maya Budi Ucup Umar

Menghapus semua antrian
Antrian kosong
PS D:\#kampus\#SEMESTER 2\laprak argoista\coding\modul 7 >
```

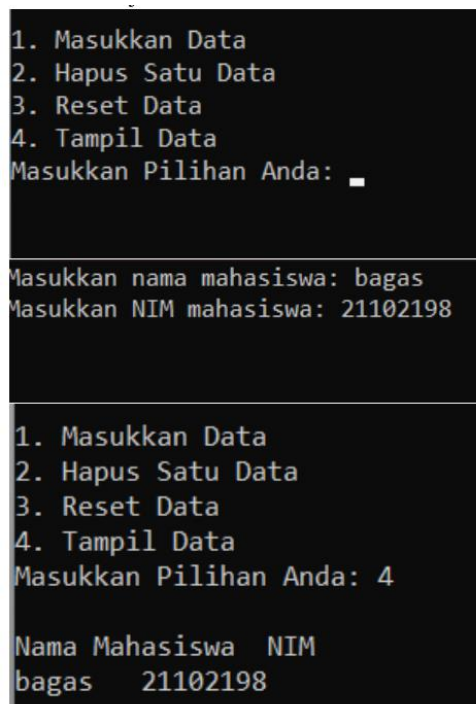
Deskripsi Program

Kode diatas merupakan perubahan dari program guided diatas yang sebelumnya berbentuk array diubah menjadi linked list dengan cara setiap elemen dalam antrian diubah menjadi node dengan 2 bagian, data untuk menyimpan nilai elemen dan next untuk menunjukkan ke elemen berikutnya dalam antrian. Untuk metode operasinya masih sama menggunakan fungsi isEmpty(), enqueue(), dequeue(), dan viewQueue().

2. Unguided 2

Dari soal no. 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM mahasiswa.

Contoh hasil jadi :



```
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
Masukkan Pilihan Anda: 1

Masukkan nama mahasiswa: bagas
Masukkan NIM mahasiswa: 21102198

1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
Masukkan Pilihan Anda: 4

Nama Mahasiswa NIM
bagas 21102198
```

Source Code

```
#include <iostream>
#include <conio.h>
using namespace std;
struct Node
{
    string nama;
    int nim;
    Node *next;
};
class Queue
{
private:
    Node *front;
    Node *rear;
public:
    Queue()
    {
        front = nullptr;
        rear = nullptr;
    }
    bool isEmpty()
    {
        return (front == nullptr);
    }
    void enqueue(string nama, int nim)
    {
        Node *newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (isEmpty())
        {
            front = newNode;
            rear = newNode;
        }
        else
        {
            rear->next = newNode;
            rear = newNode;
        }
    }
    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
```

```

        {
            Node *temp = front;
            front = front->next;
            delete temp;
        }
    }
    void viewQueue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *current = front;
            while (current != nullptr)
            {
                cout << "\t\t" << current->nama << " NIM
: "
                << current->nim << endl;
                current = current->next;
            }
            cout << endl;
        }
    }
};
int main()
{
    Queue queue;
    string opsi, nama;
    int nim;
    do
    {
        cout <<
"\n\n\t\t===== " <<
endl;
        cout << "\t\tManajemen Antrian" << endl;
        cout <<
"\t\t===== "
        << endl;
        cout << "\t\t1. Masukkan Data" << endl;
        cout << "\t\t2. Hapus Satu Data" << endl;
        cout << "\t\t3. Reset Data" << endl;
        cout << "\t\t4. Tampil Data" << endl;
        cout <<
"\t\t===== "
        << endl;
        int choice;
        cout << "\t\tMasukkan Pilihan: ";

```

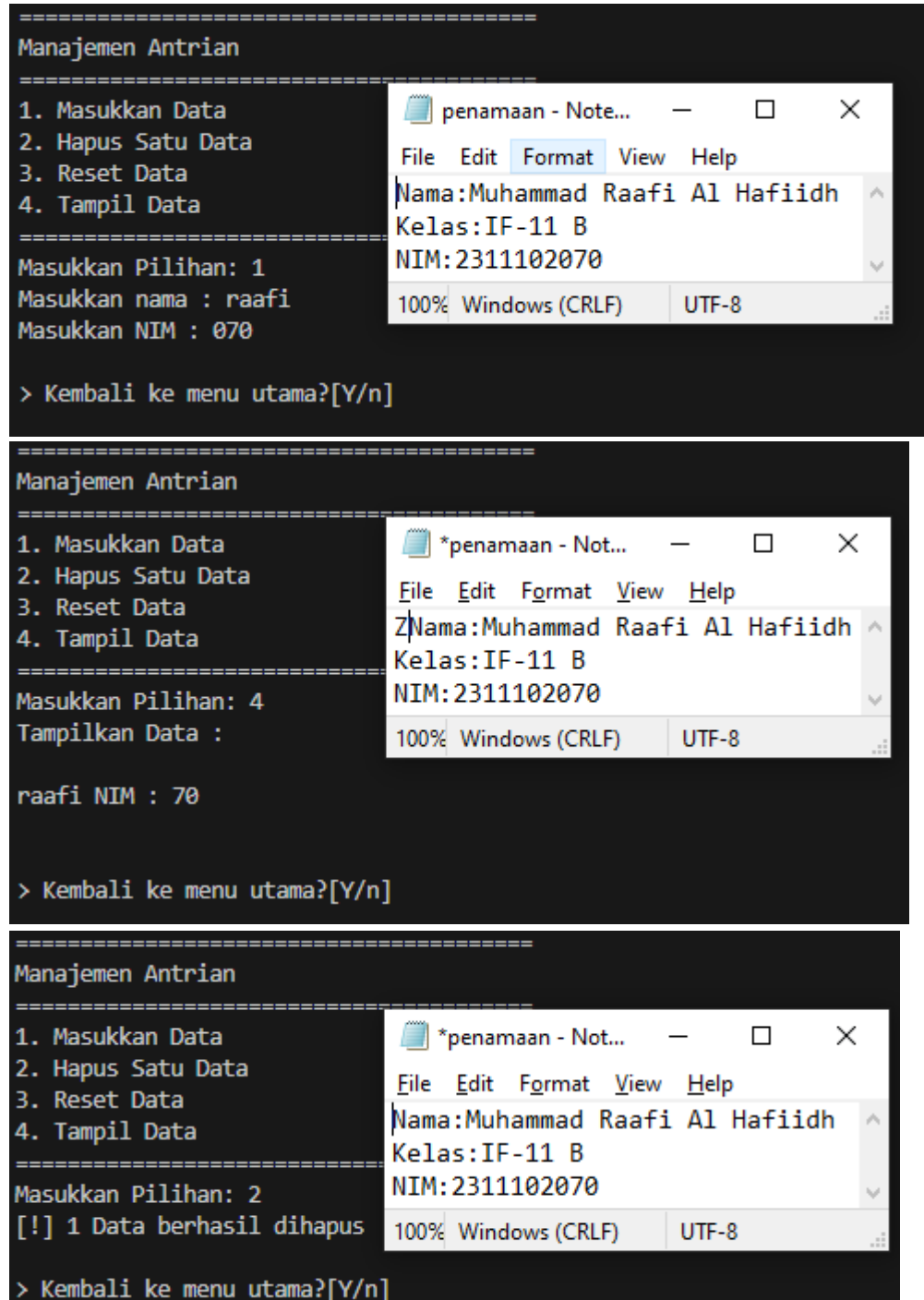
```

cin >> choice;
switch (choice)
{
case 1:
{
    cin.ignore();
    cout << "\t\tMasukkan nama : ";
    getline(cin, nama);
    cout << "\t\tMasukkan NIM : ";
    cin >> nim;
    queue.enqueue(nama, nim);
    break;
}
case 2:
{
    cout << "\t\t[!] 1 Data berhasil dihapus\n";
    queue.dequeue();
    break;
}
case 3:
{
    cout << "\t\t[!] Data berhasil di reset\n";
    while (!queue.isEmpty())
    {
        queue.dequeue();
    }
    break;
}
case 4:
{
    cout << "\t\tTampilkan Data : \n\n";
    queue.viewQueue();
    break;
}
}
cout << endl
    << "\t\t> Kembali ke menu utama?[Y/n]\n";
opsi = getch();
cout << endl;
system("cls");
} while (opsi == "y" || opsi == "Y");

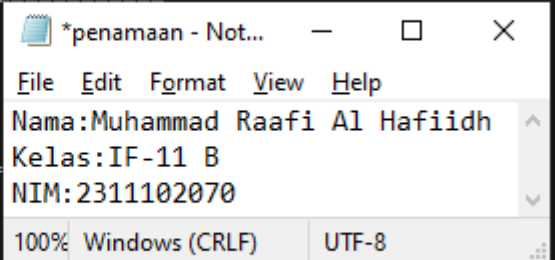
if (opsi == "n" || opsi == "N")
{
    cout << endl;
    cout << "\t\tSee you :) \n";
}
return 0;
}

```

Screenshoot Program



```
=====
Manajemen Antrian
=====
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
=====
Masukkan Pilihan: 3
[!] Data berhasil di reset
> Kembali ke menu utama?[Y/n]
```



Deskripsi Program

Untuk program diatas sama dengan unguided sebelumnya tetapi yang membedakan adalah diprogram tersebut pengguna dapat menginputkan langsung data sesuai keinginan pengguna dan dapat mengoperasikan data tersebut juga.

BAB IV

KESIMPULAN

Kesimpulan yang dapat diambil dari laporan praktikum dengan “*Modul 7 : QUEUE*” adalah bahwa praktikum ini bertujuan untuk mempelajari fungsi dari penggunaan Queue dalam pemrograman. Praktisi jadi bisa memahami penggunaan dari konsep tersebut dan bagaimana kegunaan dari konsep tersebut dalam program yang dibuat, serta setelah melakukan pratikum, mahasiswa mampu menerapkan konsep tersebut didalam program yang mereka buat. Queue pada linked list tidak memiliki batasan ukuran maksimal dan dapat secara dinamis menyesuaikan ukuran antrian nantinya.

Referensi

"**Application of Queues in Data Compression Algorithms**" (2021) oleh: D. Sharma, S. Kumar, dan M. Singh.