LAPORAN PRAKTIKUM

MODUL VII
“Queue”

Disusun oleh:
Shiva Indah Kurnia
NIM 2311102035

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

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

TUJUAN PRAKTIKUM

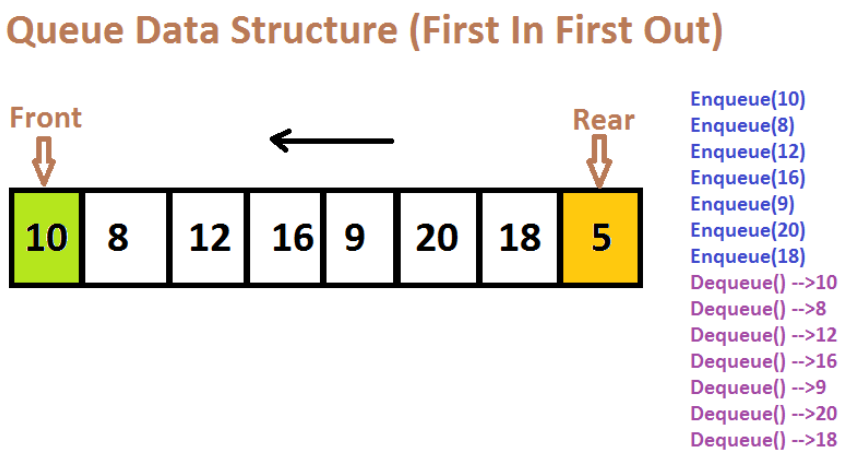
Setelah melakukan praktikum ini diharapkan mahasiswa dapat:

1. Mahasiswa diharapkan mampu menjelaskan dan memahami konsep dari Queue itu sendiri.
2. Mahasiswa diharapkan mampu mengaplikasikan Queue kedalam pemrograman C++.
3. Mahasiswa diharapkan mampu menyelesaikan studi kasus menggunakan penyelesaian program Queue yang dikombinasikan dengan array serta linked list.

DASAR TEORI

Queue bisa disebut juga antrian pada struktur data. Pengertian queue adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung yang disebut sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain.

Contoh paling sederhana dapat dilihat pada antrian. Prinsip kerja dari queue adalah prinsip "*First In First Out*" (FIFO) atau "masuk pertama keluar pertama". Sedangkan prinsip "masuk terakhir keluar pertama" atau "Last In First Out" (LIFO), digunakan pada tumpukan atau stack.



Gambar 1. *Queue Data Structure*

Pada antrian terdapat satu pintu masuk di salah satu ujung dan satu pintu keluar di ujung lainnya. Maka ada penunjuk yang menunjukkan awal dan akhir. Operasi penting dalam antrian:

1. Add yang berfungsi menambah sebuah elemen ke dalam antrian.
2. Delete yang berfungsi menghapus atau mengeluarkan elemen dari antrian.

Dalam ilmu komputer, antrian banyak digunakan terutama dalam sistem operasi yang memerlukan manajemen sumber daya dan penjadwalan. Contohnya time-sharing computer-system yang bisa dipakai oleh sejumlah orang secara serempak.

Sebuah antrian memiliki suatu operasi bernama add_priority. Dalam hal ini, antrian tidak lagi menerapkan konsep antrian yang murni, namun berubah menjadi antrian sesuai prioritas tertentu pada elemen, dan elemen yang baru ditambah tidak selalu berada di akhir.

Operasi-Operasi Queue:

1. **Create**
   Untuk menciptakan dan menginisialisasi antrian dengan cara membuat Head dan Tail = -1
2. **IsEmpty**
   Dipakai untuk memeriksa penuh tidaknya sebuah antrian dengan cara memeriksa nilai tail, jika tail = -1 maka empty. Kita tidak memeriksa head, karena head adalah tanda kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah. Pergerakan pada antrian terjadi dengan penambahan elemen antrian di bagian belakang, yaitu menggunakan nilai tail.
3. **IsFull**
   Dipakai untuk mengecek penuh tidaknya antrian dengan cara mengecek nilai tail, jika tail >= MAX-1 (karena MAX-1 adalah batas elemen array pada C) maka sudah penuh.
4. **EnQueue**
   Digunakan untuk penambahan elemen ke dalam antrian, penambahan elemen selalu ditambahkan di elemen paling belakang. Penambahan elemen selalu menggerakan variabel tail dengan cara increment counter tail terlebih dahulu
5. **DeQueue**
   Dipakai untuk menghapus elemen terdepan (head) dari antrian dengan cara menggeser semua elemen antrian ke bagian depan dan mengurangi tail dengan 1 penggeseran yang dilakukan dengan menggunakan pengulangan.
6. **Clear**
   Digunakan untuk menghapus elemen-elemen antrian dengan cara membuat tail dan head bernilai -1. Penghapusan elemen-elemen antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesannya menjadi -1 sehingga elemen-elemen antrian tidak lagi terbaca.
7. **Tampil**
   Dipakai untuk menampilkan nilai-nilai elemen antrian menggunakan pengulangan dari head hingga tail.

GUIDED

1. Guided 1

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
        {                               // antriannya 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("Dila");
    enqueueAntrian("Isna");
    enqueueAntrian("Adel");
    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

Program dimulai dengan adanya sebuah antrian teller dengan maksimal kapasitas 5 orang. Variabel `maksimalQueue` digunakan untuk menyimpan nilai maksimal dari antrian. Variabel `front` dan `back` digunakan sebagai penanda untuk mengakses elemen pertama dan terakhir dalam antrian. Array `queueTeller` berfungsi untuk menyimpan nama-nama pelanggan dalam antrian.

Pada awalnya, antrian masih kosong. **Fungsi `isEmpty()`** digunakan untuk memeriksa apakah antrian kosong atau tidak. Jika `back` memiliki nilai 0, artinya antrian kosong, maka fungsi `isEmpty()` akan mengembalikan nilai true. **Fungsi `enqueueAntrian`**(string data)` digunakan untuk menambahkan nama pelanggan ke dalam antrian. Jika antrian sudah penuh (nilai `back` sama dengan `maksimalQueue`), maka **fungsi `isFull()`** akan mengembalikan nilai true dan akan ditampilkan pesan "antrian penuh". Namun, jika antrian masih ada tempat kosong,

maka nama pelanggan akan ditambahkan ke dalam array `queueTeller` pada posisi `back`. Setelah itu, nilai `back` akan ditambahkan 1.

Fungsi `dequeueAntrian()` digunakan untuk mengurangi jumlah antrian dengan menghapus pelanggan pertama dari antrian. Jika antrian kosong (nilai `back` sama dengan 0), maka akan ditampilkan pesan "antrian kosong". Jika tidak, maka elemen-elemen dalam array `queueTeller` akan bergeser satu posisi ke depan. Setelah itu, nilai `back` akan dikurangi 1. **Fungsi `countQueue()`** digunakan untuk menghitung jumlah pelanggan dalam antrian. Fungsi ini mengembalikan nilai `back`.

Fungsi `clearQueue()` digunakan untuk menghapus semua elemen dalam antrian. Jika antrian kosong, maka akan ditampilkan pesan "antrian kosong". Jika tidak, semua elemen dalam array `queueTeller` akan dihapus dengan mengosongkan nilainya. Selain itu, nilai `back` dan `front` akan dikembalikan ke nilai awal yaitu 0. **Fungsi `viewQueue()`** digunakan untuk menampilkan semua elemen dalam antrian. Fungsi ini akan menampilkan nomor urut antrian dan nama pelanggan yang ada di dalamnya. Jika ada elemen kosong, maka akan ditampilkan "(kosong)".

UNGUIDED

1. Unguided 1

```cpp
#include <iostream>

using namespace std;

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

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

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

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

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

        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
```

```
            back->next = newNode;
            back = newNode;
        }
    }

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

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

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

    void viewQueue()
    {
        cout << "Data antrian teller: " << endl;
        Node *temp = front;
        int pos = 1;
        while (temp != nullptr)
```

```
        {
            cout << pos << ". " << temp->data << endl;
            temp = temp->next;
            pos++;
        }
    }
};

int main()
{
    Queue queue;
    queue.enqueue("Dila");
    queue.enqueue("Hijry");
    queue.enqueue("Ashofia");
    queue.enqueue("Ayu");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;

    return 0;
}
```

Screenshot program

Deskripsi program

Kelas Queue memiliki dua anggota data:

"Front" artinya bagian depan antrian dan "rear" artinya bagian belakang antrian. Di konstruktor, kedua anggota ini diinisialisasi ke nullptr, menunjukkan antrean kosong.

1. Metode isEmpty() digunakan untuk memeriksa apakah antrian kosong. Jika depan nullptr, antrian dianggap kosong.
2. Metode enqueue() digunakan untuk menambahkan item baru ke antrian. Node baru dibuat dan nilai data ditentukan sesuai dengan parameter data. Jika antrian kosong, titik maju dan mundur mengarah ke simpul baru ini. Jika antrean tidak kosong, node baru ditambahkan ke akhir antrean dan backend diperbarui.
3. Metode dequeue() menghapus item pertama dari antrian. Jika antrian kosong, pesan "Antrian kosong" ditampilkan. Jika tidak, simpul sebelumnya akan dihapus dan bagian depan diperbarui ke simpul berikutnya.
4. Metode countQueue() digunakan untuk menghitung jumlah item dalam antrian. Metode ini mengiterasi semua node dalam daftar tertaut dan setiap node dihitung.
5. Metode ClearQueue() digunakan untuk menghapus semua antrian. Metode ini memanggil dequeue() hingga antrian kosong. Metode viewQueue() digunakan untuk melihat semua item dalam antrian. Metode ini mengulangi semua node dalam daftar tertaut dan menampilkan nilai datanya.

Dalam fungsi main(), operasi antrian diuji dengan menambahkan elemen, menampilkan antrian, menghitung jumlah elemen, menghapus elemen pertama, menghapus semua elemen, dan menampilkan antrian setelah setiap operasi.

2. Unguided 2

```cpp
#include <iostream>

using namespace std;

struct Node
{
    string nama;
    string nim;
    Node *next;
};

class Queue
{
private:
```

```
    Node *front;
    Node *back;

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

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

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

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

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

```
        }
    }

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

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

    void viewQueue()
    {
        cout << "Data antrian Mahasiswa: " << endl;
        Node *temp = front;
        int pos = 1;
        while (temp != nullptr)
        {
            cout << pos << ". Nama: " << temp->nama << "|| NIM:
" << temp->nim << endl;
            temp = temp->next;
            pos++;
        }
    }
};

int main()
{
    Queue queue;
    int choice;

    do
    {
```

```
        cout << "=== Menu Antrian Mahasiswa Telkom ===" << endl;
        cout << "1. Masukkan data" << endl;
        cout << "2. Hapus satu data" << endl;
        cout << "3. Reset data" << endl;
        cout << "4. Tampil data" << endl;
        cout << "5. Keluar" << endl;
        cout << "====================================\n"
             << endl;
        cout << "Masukkan Pilihan Anda: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
        {
            string nama, nim;
            cout << "Masukkan Nama Mahasiswa : ";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan NIM Mahasiswa  : ";
            cin >> nim;
            queue.enqueue(nama, nim);
            cout << "Data berhasil dimasukkan ke dalam antrian"
<< endl;
            break;
        }
        case 2:
        {
            if (queue.isEmpty())
            {
                cout << "Antrian kosong" << endl;
            }
            else
            {
                queue.dequeue();
                cout << "Data berhasil dihapus dari antrian" <<
endl;
            }
            break;
        }
        case 3:
        {
            if (queue.isEmpty())
            {
                cout << "Antrian kosong" << endl;
```

```
            }
            else
            {
                queue.clearQueue();
                cout << "Data berhasil di-reset" << endl;
            }
            break;
        }
        case 4:
        {
            if (queue.isEmpty())
            {
                cout << "Antrian kosong" << endl;
            }
            else
            {
                queue.viewQueue();
            }
            break;
        }
        case 5:
        {
            cout << "Terima kasih telah menggunakan layanan
kami;)" << endl;
            break;
        }
        default:
        {
            cout << "Pilihan yang anda masukkan tidak valid" <<
endl;
            break;
        }
        }

        cout << endl;

    } while (choice != 5);

    return 0;
}
```

Screenshot program

Deskripsi program

Program diatas memiliki bentuk yang sama dengan bentuk nomor sebelumnya. Hanya saja program kali ini dibuat untuk membuat inputan terkait dengan Nama dan NIM mahasiswa.

Dalam program ini, terdapat menu yang akan ditampilkan kepada pengguna. Pengguna dapat memilih salah satu opsi sesuai dengan kebutuhan:

1. Masukkan data: Pengguna diminta untuk memasukkan nama mahasiswa dan NIM mahasiswa. Data ini akan ditambahkan ke dalam antrian.
2. Hapus satu data: Jika antrian tidak kosong, data pertama dalam antrian akan dihapus.
3. Reset data: Jika antrian tidak kosong, semua data dalam antrian akan dihapus.
4. Tampil data: Jika antrian tidak kosong, semua data dalam antrian akan ditampilkan.
5. Keluar: Program berakhir.

Pengguna dapat memilih opsi menu sesuai keinginan, dan program akan menanggapi sesuai dengan opsi yang dipilih. Program akan terus berjalan dan menampilkan menu hingga pengguna memilih untuk keluar dengan memilih opsi 5.

KESIMPULAN

Setelah mempelajari materi tentang Queue dan penerapannya pada Linked List dan Array kita dapat mengetahui bahwa Queue adalah struktur data yang mengikuti prinsip FIFO (First-In-First-Out), artinya elemen yang pertama kali masuk akan menjadi yang pertama kali keluar. Queue dapat diimplementasikan menggunakan Linked List atau Array. Linked List memberikan fleksibilitas dalam penambahan dan penghapusan elemen, sementara Array menyediakan akses langsung ke elemen-elemen dalam antrian. Implementasi Queue menggunakan Linked List membutuhkan alokasi memori dinamis untuk setiap elemen yang ditambahkan atau dihapus. Ini memungkinkan antrian tumbuh atau menyusut sesuai kebutuhan. Implementasi Queue menggunakan Array memiliki batasan ukuran tetap, yang berarti antrian memiliki kapasitas maksimum yang sudah ditentukan sebelumnya. Jika antrian penuh, tidak mungkin menambahkan elemen baru ke dalamnya. Operasi dasar pada Queue meliputi enqueue (menambahkan elemen ke dalam antrian), dequeue (menghapus elemen dari antrian), isEmpty (memeriksa apakah antrian kosong), countQueue (menghitung jumlah elemen dalam antrian), dan viewQueue (menampilkan elemen-elemen dalam antrian). Penggunaan Queue sangat berguna dalam situasi di mana ada kebutuhan untuk memproses data secara berurutan sesuai dengan waktu kedatangannya, seperti simulasi antrian pelanggan atau penjadwalan tugas. Pemilihan implementasi Queue (Linked List atau Array) tergantung pada kebutuhan dan karakteristik spesifik masalah yang akan diselesaikan. Linked List lebih fleksibel, sementara Array lebih efisien dalam akses elemen.

Dengan memahami konsep dasar Queue dan penerapannya menggunakan Linked List dan Array, kita dapat memanfaatkannya untuk memecahkan berbagai masalah yang melibatkan pengolahan data berdasarkan urutan waktu kedatangan.