LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
Anita Nurazizah Agussalim
NIM: 2311102017

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

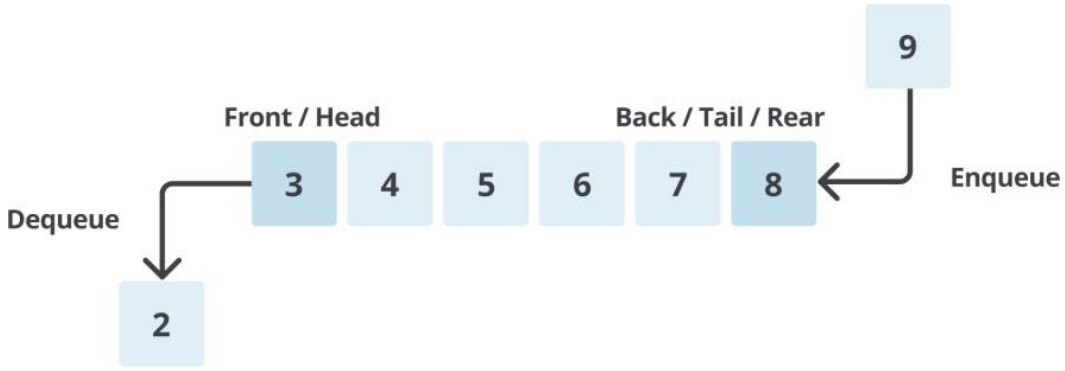
1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II

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

1. Guided

Source code

```
#include <iostream>

using namespace std;

const int maksimalQueue = 5;
int front = 0;
int back = 0;

string queueTeller[5];

bool isFull()
{
  if (back == maksimalQueue)
  {
    return true; // =1
  }
  else
  {
    return false;
  }
}

bool isEmpty()
{
  if (back == 0)
  {
    return true;
  }
  else
```

```
  {
    return false;
  }
}

void enqueueAntrian(string data)
{
  if (isFull())
  {
    cout << "Antrian penuh" << endl;
  }
  else
  {
    if (isEmpty())
    {
      queueTeller[0] = data;
      front++;
      back++;
    }
    else
    {
      queueTeller[back] = data;
      back++;
    }
  }
}

void dequeueAntrian()
{
  if (isEmpty())
  {
    cout << "Antrian kosong" << endl;
  }
  else
```

```
  {
    for (int i = 0; i < back; i++)
    {
      queueTeller[i] = queueTeller[i + 1];
    }
    back--;
  }
}

int countQueue()
{
  return back;
}

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

void viewQueue()
{
  cout << "\nData 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;
}
```

Screenshot Output

```
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

Program di atas adalah implementasi sederhana dari antrian (queue) menggunakan array dengan kapasitas maksimum lima elemen. Program ini menyediakan fungsi-fungsi untuk memeriksa apakah antrian penuh atau kosong, menambahkan elemen ke antrian (enqueue), menghapus elemen dari antrian (dequeue), menghitung jumlah elemen dalam antrian, mengosongkan seluruh antrian, dan menampilkan isi antrian. Antrian diimplementasikan dengan menjaga indeks depan (front) dan belakang (back) untuk melacak posisi elemen dalam array. Program utama menambahkan dua elemen ke antrian, menampilkan isi antrian, menghapus satu elemen, dan kemudian mengosongkan antrian, dengan setiap langkah menampilkan kondisi antrian dan jumlah elemen yang tersisa.

LATIHAN KELAS - 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* back;
public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

```
            }
            cout << "Antrian ditambahkan: " << data << endl;
        }

        void dequeueAntrian() {
            if (isEmpty()) {
                cout << "\nAntrian kosong" << endl;
                return;
            }
            Node* temp = front;
            cout << "\nAntrian dihapus: " << front->data <<
    endl;
            front = front->next;
            delete temp;
        }

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

        void clearQueue() {
            while (!isEmpty()) {
                dequeueAntrian();
            }
            cout << "\nAntrian dikosongkan" << endl;
        }

        void viewQueue() {
```

```
            if (isEmpty()) {
                cout << "Antrian kosong" << endl;
                return;
            }
            cout << "\nData antrian teller:" << endl;
            Node* current = front;
            int position = 1;
            while (current != nullptr) {
                cout << position << ". " << current->data <<
    endl;
                current = current->next;
                position++;
            }
        }

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

    int main() {
        Queue antrian;
        antrian.enqueueAntrian("Andi");
        antrian.enqueueAntrian("Maya");
        antrian.viewQueue();
        cout << "Jumlah antrian = " << antrian.countQueue()
    << endl;
        antrian.dequeueAntrian();
        antrian.viewQueue();
        cout << "Jumlah antrian = " << antrian.countQueue()
    << endl;
        antrian.clearQueue();
        antrian.viewQueue();
```

```
            cout << "Jumlah antrian = " << antrian.countQueue()
       << endl;
            return 0;
}
```

OUTPUT PROGRAM

```
Antrian ditambahkan: Andi
Antrian ditambahkan: Maya

Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2

Antrian dihapus: Andi

Data antrian teller:
1. Maya
Jumlah antrian = 1

Antrian dihapus: Maya

Antrian dikosongkan
Antrian kosong
Jumlah antrian = 0
```

DESKRIPSI PROGRAM

Program di atas mengimplementasikan antrian (queue) menggunakan struktur data linked list. Kelas ‘Queue’ memiliki dua pointer, ‘front’ dan ‘back’, yang menunjuk ke elemen pertama dan terakhir dalam antrian. Metode ‘enqueueAntrian’ menambahkan elemen baru ke belakang antrian, sementara ‘dequeueAntrian’ menghapus elemen dari depan antrian. Metode ‘countQueue’ menghitung jumlah elemen dalam antrian, ‘clearQueue’ mengosongkan antrian dengan menghapus semua elemen, dan ‘viewQueue’ menampilkan semua elemen dalam antrian dari depan ke belakang. Metode ‘isEmpty’ memeriksa apakah antrian kosong. Dalam fungsi ‘main’, berbagai operasi antrian dilakukan untuk mendemonstrasikan fungsionalitasnya, termasuk menambahkan, menghapus, menampilkan, menghitung, dan mengosongkan antrian.

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

SOURCE CODE

```
#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;
    }

    void enqueueAntrian(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;
```

```
        }
        cout << "\nAntrian ditambahkan: Nama: " << nama << ",
NIM: " << nim << endl;
    }

    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "\nAntrian dihapus: Nama: " << front->nama <<
", NIM: " << front->nim << endl;
        front = front->next;
        delete temp;
    }

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

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "\nAntrian dikosongkan" << endl;
    }
```

```
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        cout << "\nData antrian mahasiswa:" << endl;
        Node* current = front;
        int position = 1;
        while (current != nullptr) {
            cout << position << ". Nama: " << current->nama <<
", NIM: " << current->nim << endl;
            current = current->next;
            position++;
        }
    }

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

int main() {
    Queue antrian;
    antrian.enqueueAntrian("Anita Nurazizah Agussalim",
"2311102017");
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() <<
endl;
    antrian.dequeueAntrian();
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() <<
endl;
    antrian.clearQueue();
    antrian.viewQueue();
```

```
    cout << "Jumlah antrian = " << antrian.countQueue() <<
endl;
    return 0;
}
```

OUTPUT PROGRAM

```
Antrian ditambahkan: Nama: Anita Nurazizah Agussalim, NIM: 2311102017

Data antrian mahasiswa:
1. Nama: Anita Nurazizah Agussalim, NIM: 2311102017
Jumlah antrian = 1

Antrian dihapus: Nama: Anita Nurazizah Agussalim, NIM: 2311102017
Antrian kosong
Jumlah antrian = 0

Antrian dikosongkan
Antrian kosong
Jumlah antrian = 0
```

DESKRIPSI PROGRAM

Program di atas mengimplementasikan antrian (queue) menggunakan struktur data linked list untuk menyimpan data mahasiswa, dengan setiap elemen menyimpan nama dan NIM. Kelas ‘Queue’ memiliki metode ‘enqueueAntrian’ untuk menambahkan mahasiswa ke belakang antrian, ‘dequeueAntrian’ untuk menghapus mahasiswa dari depan antrian, ‘countQueue’ untuk menghitung jumlah mahasiswa dalam antrian, ‘clearQueue’ untuk mengosongkan antrian, dan ‘viewQueue’ untuk menampilkan semua mahasiswa dalam antrian. Metode ‘isEmpty’ memeriksa apakah antrian kosong. Dalam fungsi ‘main’, berbagai operasi antrian dijalankan untuk menambah, menghapus, menampilkan, menghitung, dan mengosongkan antrian, mendemonstrasikan fungsionalitas antrian tersebut.

BAB IV

KESIMPULAN

Queue adalah struktur data yang menerapkan prinsip FIFO (First-In First-Out), di mana elemen yang pertama dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan antrian dalam kehidupan sehari-hari, di mana orang yang datang lebih dulu akan dilayani lebih dulu. Implementasi queue dapat dilakukan menggunakan array atau linked list dan terdiri dari dua pointer utama: front (head) untuk menunjuk elemen pertama, dan rear (tail/back) untuk menunjuk elemen terakhir. Perbedaan utama antara stack dan queue adalah pada aturan penambahan dan penghapusan elemen; stack menggunakan prinsip LIFO (Last-In First-Out), sedangkan queue menggunakan FIFO. Operasi utama pada queue meliputi ‘enqueue’ (menambahkan elemen), ‘dequeue’ (menghapus elemen), ‘peek’ (mengambil elemen tanpa menghapusnya), ‘isEmpty’ (mengecek apakah queue kosong), ‘isFull’ (mengecek apakah queue penuh), dan ‘size’ (menghitung jumlah elemen dalam queue).

DAFTAR PUSTAKA

Asisten Praktikum. (2024). *MODUL 7: QUEUE.*

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.