LAPORAN PRAKTIKUM

MODUL 7
QUEUE

Disusun oleh:
Nofita Fitriyani
NIM: 2311102001

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

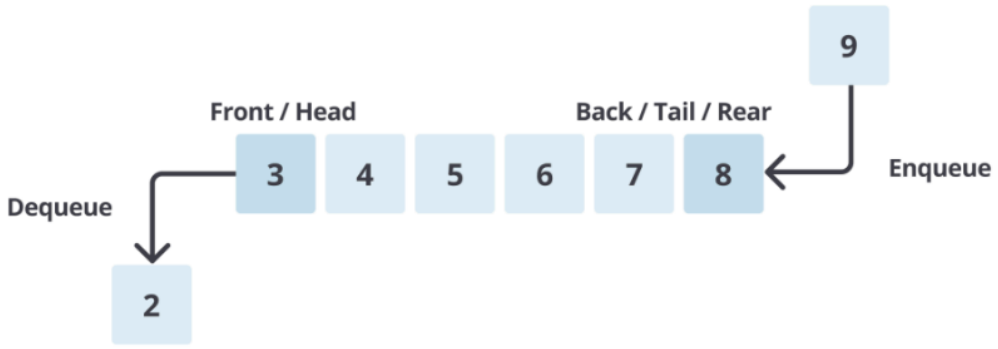
1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

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 1

Source code

```
#include <iostream>

using namespace std;

const 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;
  }
}

bool isEmpty() {  // Antriannya kosong atau tidak
  if (back == 0) {
    return true;
  } else {
    return false;
  }
}

void enqueueAntrian(string data) {  // Fungsi menambahkan antrian
  if (isFull()) {
    cout << "Antrian penuh" << endl;
```

```
  } else {
    if (isEmpty()) {  // Kondisi ketika queue kosong
      queueTeller[0] = data;
      front++;
      back++;
    } else {  // Antrianya ada isi
      queueTeller[back] = data;
      back++;
    }
  }
}

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

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

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

```
    back = 0;
    front = 0;
  }
}

void viewQueue() {  // Fungsi melihat antrian
  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

```
PS C:\Users\LENOVO> cd "c:\Praktikum
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 C:\Praktikum Struktur Data\Modul 7>
```

Deskripsi program

Program ini merupakan implementasi sederhana dari antrian (queue) menggunakan array. Antrian memiliki kapasitas maksimal 5 orang dan digunakan untuk mensimulasikan antrian pada teller bank. Program ini mencakup beberapa fungsi utama: isFull() untuk memeriksa apakah antrian penuh, isEmpty() untuk memeriksa apakah antrian kosong, enqueueAntrian() untuk menambahkan elemen ke dalam antrian, dequeueAntiran() untuk menghapus elemen dari antrian, countQueue() untuk menghitung jumlah elemen dalam antrian, clearQueue() untuk menghapus semua elemen dalam antrian, dan viewQueue() untuk menampilkan seluruh elemen dalam antrian. Fungsi main() menambahkan dua nama ke dalam antrian, menampilkan isi antrian dan jumlah elemen, menghapus satu elemen,

kemudian menampilkan kembali antrian dan jumlah elemen, lalu menghapus semua elemen dalam antrian dan menampilkan isi antrian yang sudah kosong. Program ini membantu memahami konsep dasar antrian dalam pemrograman, termasuk operasi penambahan, penghapusan, dan pengecekan elemen.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list.
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

Source code

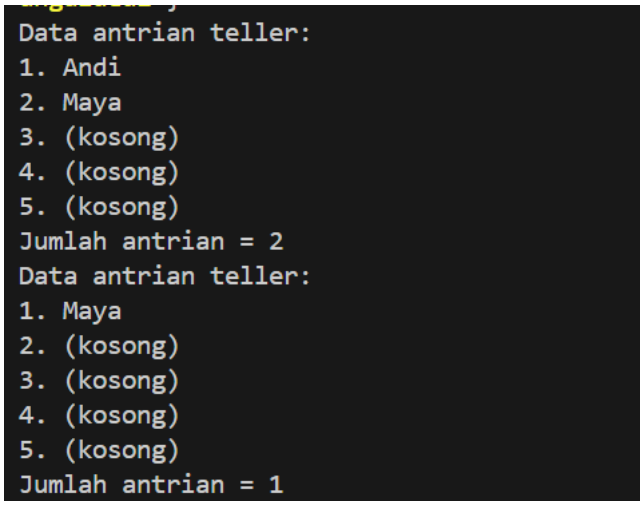
```
#include <iostream>

using namespace std;

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

Node* front = nullptr;
Node* back = nullptr;
const int maksimalQueue = 5; // Batas maksimal antrian

int countQueue();  // Deklarasi fungsi countQueue

bool isFull() {
    return countQueue() == maksimalQueue;
}

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

void enqueueAntrian(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
```

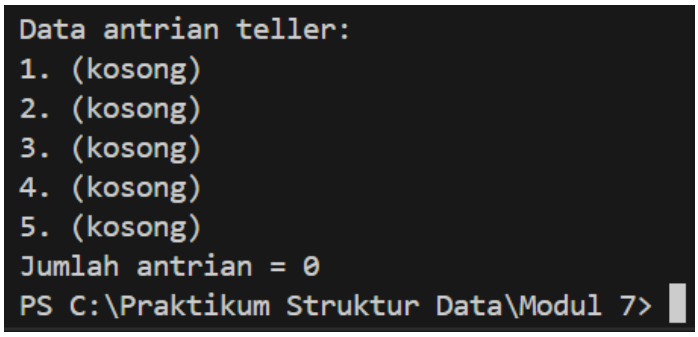
```
        return;
    }

    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;

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

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

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

```
    }
    return count;
}

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

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    for (int i = 0; i < maksimalQueue; i++) {
        if (temp != nullptr) {
            cout << index << ". " << temp->data << endl;
            temp = temp->next;
        } else {
            cout << index << ". (kosong)" << endl;
        }
        index++;
    }
}

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





Deskripsi program

Program ini merupakan implementasi antrian (queue) menggunakan struktur data linked list di C++ dengan batas maksimal lima elemen yang masing-masing berisi data berupa string. Program mencakup fungsi untuk menambahkan elemen ke belakang antrian (enqueueAntrian), menghapus elemen dari depan antrian (dequeueAntrian), memeriksa apakah antrian penuh (isFull) atau kosong (isEmpty), menghitung jumlah elemen dalam antrian (countQueue) menghapus semua elemen dalam antrian (clearQueue) dan menampilkan seluruh elemen dalam antrian, mengisi posisi kosong dengan "(kosong)" (viewQueue). Dalam

fungsi main(), beberapa operasi antrian dilakukan: menambahkan dua elemen ("Andi" dan "Maya"), menampilkan isi antrian, menghitung jumlah elemen, menghapus satu elemen, dan menghapus semua elemen, kemudian program menampilkan antrian pada setiap langkah untuk menunjukkan perubahan yang terjadi.

Source code

```
#include <iostream>

using namespace std;

struct Mahasiswa {
    string nama;
    string nim;
};

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

Node* front = nullptr;
Node* back = nullptr;
const int maksimalQueue = 5; // Batas maksimal antrian

int countQueue();  // Deklarasi fungsi countQueue

bool isFull() {
    return countQueue() == maksimalQueue;
}

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

```
}

void enqueueAntrian(Mahasiswa data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
        return;
    }

    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;

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

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

int countQueue() {
```

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

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

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    for (int i = 0; i < maksimalQueue; i++) {
        if (temp != nullptr) {
            cout << index << ". Nama: " << temp->data.nama << ",
NIM: " << temp->data.nim << endl;
            temp = temp->next;
        } else {
            cout << index << ". (kosong)" << endl;
        }
        index++;
    }
}

int main() {
    Mahasiswa mhs1 = {"Andi", "2311102001"};
    Mahasiswa mhs2 = {"Maya", "2311102002"};
```

```
    enqueueAntrian(mhs1);
    enqueueAntrian(mhs2);
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}
```

Screenshoot program

```
PS C:\Praktikum Struktur Data\Modul 7> cd "c:\Prakti
Unguided2 }
Data antrian teller:
1. Nama: Andi, NIM: 2311102001
2. Nama: Maya, NIM: 2311102002
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
```

```
Data antrian teller:
1. Nama: Maya, NIM: 2311102002
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 C:\Praktikum Struktur Data\Modul 7>
```

Deskripsi program

Program ini adalah implementasi antrian (queue) menggunakan linked list di C++ untuk menyimpan data mahasiswa yang terdiri dari nama dan NIM. Program mencakup fungsi-fungsi untuk menambahkan mahasiswa ke dalam antrian (enqueueAntrian), menghapus mahasiswa dari antrian (dequeueAntrian) memeriksa apakah antrian penuh (isFull) atau kosong (isEmpty), menghitung jumlah mahasiswa dalam antrian (countQueue), menghapus semua mahasiswa dari antrian (clearQueue) dan menampilkan daftar mahasiswa dalam antrian beserta posisi kosong (viewQueue). Dalam fungsi main(), beberapa operasi dilakukan: menambahkan dua mahasiswa ("Andi" dan "Maya") ke dalam antrian, menampilkan isi antrian, menghitung jumlah mahasiswa dalam antrian, menghapus satu mahasiswa dari antrian, dan menghapus semua mahasiswa, sambil menampilkan kondisi antrian pada setiap langkah untuk menunjukkan perubahan yang terjadi.

BAB IV

KESIMPULAN

Queue adalah struktur data yang menggunakan metode First-In First-Out (FIFO), dimana elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan. Queue mirip dengan antrian dalam kehidupan sehari-hari dan diimplementasikan menggunakan array atau linked list dengan dua pointer utama: front (mengacu pada elemen pertama) dan rear (mengacu pada elemen terakhir). Berbeda dengan stack yang mengikuti metode Last-In First-Out (LIFO), queue melakukan operasi penambahan (enqueue) dan penghapusan (dequeue) elemen di tempat yang berbeda. Operasi dasar pada queue termasuk enqueue (menambahkan elemen), dequeue (menghapus elemen), peek (mengambil elemen tanpa menghapusnya), isEmpty (memeriksa apakah queue kosong), isFull (memeriksa apakah queue penuh), dan size (menghitung jumlah elemen dalam queue).