LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
Imelda Fajar Awalina Crisyanti
NIM: 2311102004

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I

TUJUAN PRAKTIKUM

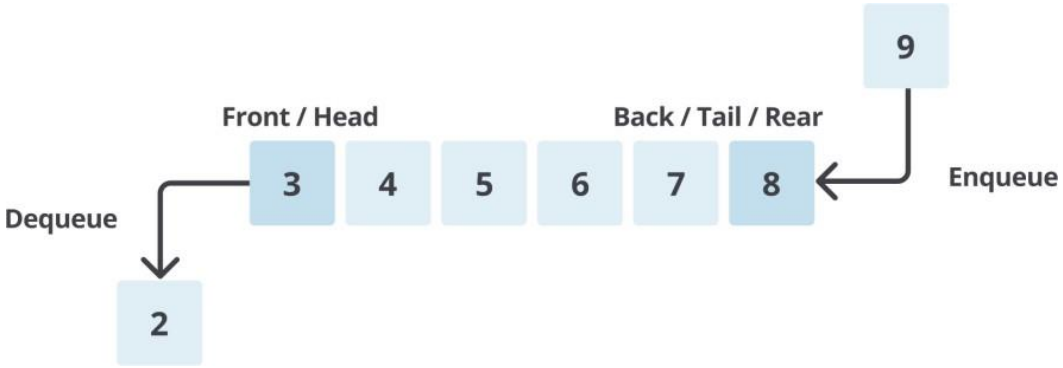
1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

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 menjadidata 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

```
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 antrian sederhana menggunakan array di dalam C++. Berikut adalah deskripsi singkatnya:

Konstanta dan Variabel:

- maksimalQueue: Batas maksimum elemen antrian (5).
- front dan back: Penanda posisi depan dan belakang antrian.
- queueTeller: Array yang menyimpan elemen antrian.

Fungsi Utama:

- isFull(): Mengecek apakah antrian penuh.
- isEmpty(): Mengecek apakah antrian kosong.
- enqueueAntrian(string data): Menambahkan elemen ke antrian.
- dequeueAntrian(): Menghapus elemen dari antrian.
- countQueue(): Menghitung jumlah elemen dalam antrian.
- clearQueue(): Menghapus semua elemen antrian.
- viewQueue(): Menampilkan elemen dalam antrian.

Fungsi main():

Menambahkan elemen ke antrian, menampilkan antrian, menghapus elemen, dan mengosongkan antrian, serta menampilkan jumlah elemen setelah setiap operasi.

Program ini menunjukkan bagaimana menambah, menghapus, dan melihat elemen dalam antrian dengan kapasitas tetap.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

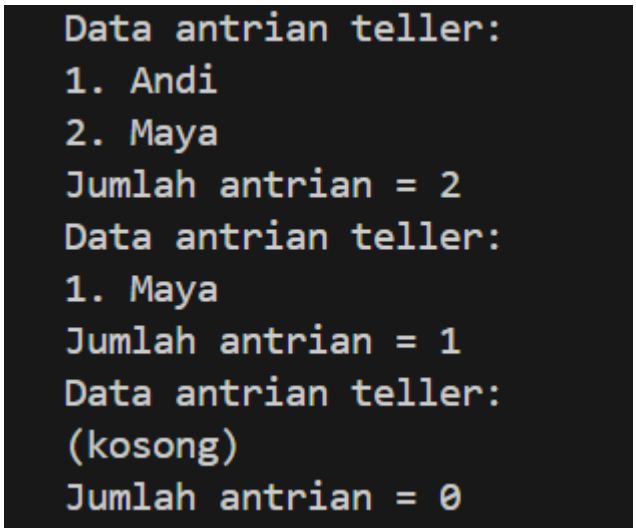
Source code

```
#include <iostream>

using namespace std;

const int maksimalQueue = 5;  // Maksimal tampilan antrian, hanya
untuk tampilan

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

Node* front = nullptr;
Node* back = nullptr;

bool isFull() {
    return false;  // Queue dengan linked list tidak pernah penuh
kecuali memori habis
}

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

void enqueueAntrian(string data) {
    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;
        if (front == nullptr) {
            back = nullptr;
        }
        delete temp;
    }
}

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;
    while (temp != nullptr) {
        cout << index << ". " << temp->data << endl;
        temp = temp->next;
        index++;
    }
    if (isEmpty()) {
        cout << "(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



Deskripsi program

Program di atas adalah implementasi antrian (queue) menggunakan linked list dalam bahasa C++. Antrian ini mendukung operasi dasar seperti menambahkan elemen ke antrian (enqueue), menghapus elemen dari antrian (dequeue), menghitung jumlah elemen dalam antrian, mengosongkan antrian, dan menampilkan isi antrian. Program ini juga menyertakan fungsi utama yang mendemonstrasikan operasi-operasi tersebut dengan beberapa contoh.

Berikut adalah deskripsi singkat setiap fungsi:

- isFull(): Mengembalikan false karena antrian linked list tidak memiliki batas selain memori yang tersedia.
- isEmpty(): Mengembalikan true jika antrian kosong (pointer front adalah nullptr).
- enqueueAntrian(string data): Menambahkan elemen baru ke akhir antrian.
- dequeueAntrian(): Menghapus elemen dari depan antrian.
- countQueue(): Menghitung dan mengembalikan jumlah elemen dalam antrian.
- clearQueue(): Mengosongkan antrian dengan menghapus semua elemen satu per satu.
- viewQueue(): Menampilkan semua elemen dalam antrian, atau "(kosong)" jika antrian kosong.
- main(): Fungsi utama yang mendemonstrasikan penggunaan fungsi-fungsi di atas dengan beberapa operasi pada antrian, seperti menambahkan elemen, menampilkan antrian, menghapus elemen, dan mengosongkan antrian.

2. Unguided 2

Source code

```
#include <iostream>

using namespace std;

const int maksimalQueue = 5;  // Maksimal tampilan antrian, hanya
untuk tampilan

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

Node* front = nullptr;
Node* back = nullptr;

bool isFull() {
    return false;  // Queue dengan linked list tidak pernah penuh
kecuali memori habis
```

```
}

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

void enqueueAntrian(string nama, string nim) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    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;
        if (front == nullptr) {
            back = nullptr;
        }
        delete temp;
    }
}

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 mahasiswa:" << endl;
    Node* temp = front;
    int index = 1;
    while (temp != nullptr) {
        cout << index << ". Nama: " << temp->nama << ", NIM: " <<
temp->nim << endl;
        temp = temp->next;
        index++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("Andi", "2311102001");
    enqueueAntrian("Maya", "2311102002");
    enqueueAntrian("Imelda", "2311102004");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}
```

Screenshoot program

```
Data antrian mahasiswa:
1. Nama: Andi, NIM: 2311102001
2. Nama: Maya, NIM: 2311102002
3. Nama: Imelda, NIM: 2311102004
Jumlah antrian = 3
Data antrian mahasiswa:
1. Nama: Maya, NIM: 2311102002
2. Nama: Imelda, NIM: 2311102004
Jumlah antrian = 2
Data antrian mahasiswa:
(kosong)
Jumlah antrian = 0
```

Deskripsi program

Program di atas adalah implementasi antrian (queue) menggunakan linked list dalam bahasa C++ yang menyimpan data mahasiswa berupa Nama dan NIM. Program ini mendukung operasi dasar antrian seperti menambah elemen (enqueue), menghapus elemen (dequeue), menghitung jumlah elemen (countQueue), mengosongkan antrian (clearQueue), dan menampilkan isi antrian (viewQueue). Berikut adalah penjelasan singkat dari setiap fungsi dalam program:

- Struktur Node: Program ini menggunakan struktur Node untuk menyimpan data nama dan nim mahasiswa, serta pointer next yang menunjuk ke node berikutnya dalam antrian.
- Pointer Global: Terdapat dua pointer global, front dan back, yang digunakan untuk melacak elemen pertama dan terakhir dalam antrian. front menunjuk ke elemen pertama, sementara back menunjuk ke elemen terakhir.
- Fungsi isFull: Fungsi ini mengembalikan nilai false karena antrian berbasis linked list tidak memiliki batas kapasitas selain memori yang tersedia.
- Fungsi isEmpty: Fungsi ini mengembalikan true jika antrian kosong dengan memeriksa apakah pointer front adalah nullptr.
- Fungsi enqueueAntrian: Fungsi ini menambahkan node baru yang berisi nama dan nim mahasiswa ke akhir antrian. Jika antrian kosong, front dan back akan menunjuk ke node baru tersebut.
- Fungsi dequeueAntrian: Fungsi ini menghapus elemen dari depan antrian. Jika antrian kosong, fungsi menampilkan pesan "Antrian kosong". Setelah menghapus elemen depan, fungsi memperbarui pointer front ke elemen berikutnya dan menghapus node yang lama.
- Fungsi countQueue: Fungsi ini menghitung dan mengembalikan jumlah elemen dalam antrian dengan traversing dari front hingga akhir antrian.

- Fungsi clearQueue: Fungsi ini mengosongkan antrian dengan menghapus semua elemen satu per satu hingga antrian benar-benar kosong.
- Fungsi viewQueue: Fungsi ini menampilkan semua elemen dalam antrian. Jika antrian kosong, fungsi menampilkan pesan "(kosong)". Setiap elemen ditampilkan dengan format Nama dan NIM.
- Fungsi main: Fungsi utama ini mendemonstrasikan penggunaan fungsi-fungsi di atas. Program menambahkan beberapa mahasiswa ke antrian, menampilkan isi antrian, menghitung jumlah elemen, menghapus elemen dari depan antrian, dan mengosongkan antrian. Langkah-langkah tersebut memberikan gambaran bagaimana setiap operasi pada antrian bekerja dalam program ini.

BAB IV
KESIMPULAN

Double-ended queue (deque), atau yang sering disebut double queue, adalah struktur data linier yang mirip dengan queue, tetapi dengan kemampuan tambahan untuk menambah dan menghapus elemen di kedua ujung antrian. Konsep utama dari double queue adalah bahwa elemen-elemen dapat dimasukkan atau dihapus baik dari depan (front) maupun belakang (rear) antrian.

Operasi tambah dan menghapus pada queue (antrian) mengikuti prinsip FIFO (First-In-First-Out), yang berarti elemen pertama yang dimasukkan ke dalam antrian akan menjadi elemen pertama yang dihapus.

Operasi tampil data pada queue memungkinkan kita untuk melihat isi dari antrian tanpa mengubah urutan atau menghapus elemen-elemen dalam antrian tersebut.