

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

**NAMA : WISNU RANANTA RADITYA
PUTRA
NIM : 2311102013**

Dosen :

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

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

BAB I

TUJUAN PRAKTIKUM

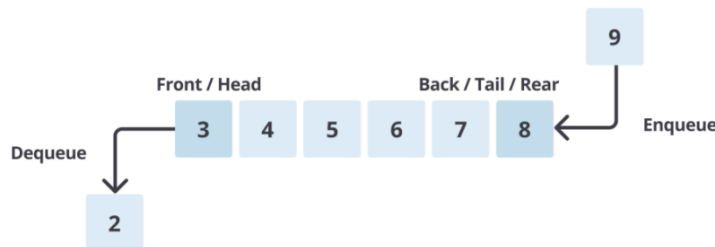
1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

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

Guided 1

```
#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 { // Antriannya 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;
}

```

Screenshots 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
PS C:\Semester 2\Praktikum Strutur Data\Modul_7>
```

Deskripsi:

Program diatas merupakan implementasi dari struktur data Queue. Program ini memiliki beberapa fungsi untuk mengelola antrian seperti menambah, menghapus, melihat, menghitung, dan mengosongkan antrian. Fungsi `isFull()`: Mengecek apakah antrian penuh. Fungsi `isEmpty()`: Mengecek apakah antrian kosong. Fungsi `enqueueAntrian(data)`: Menambahkan elemen baru jika antrian tidak penuh. Fungsi `dequeueAntrian()`: Menghapus elemen terdepan jika antrian tidak kosong. Fungsi `countQueue()`: Menghitung jumlah elemen dalam antrian. Fungsi `clearQueue()`: Menghapus semua elemen dalam antrian. Fungsi `viewQueue()`: Menampilkan isi antrian beserta posisi kosong. Pada fungsi main program menambahkan "Andi" dan "Maya" ke antrian. Selanjutnya program akan menampilkan antrian. Kemudian program akan menghapus elemen pertama dari antrian. Kemudian program akan menampilkan antrian lagi. Lalu program akan mengosongkan antrian. Kemudian program akan Menampilkan antrian setelah dikosongkan.

BAB IV

UNGUIDED

Unguided 1

```
#include <iostream>
#include <string>

using namespace std;

struct Node {
    string data_2311102013;
    Node* next;
};

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

bool isFull() {
    return false;
}

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

void enqueueAntrian(string data_2311102013) {
    Node* newNode = new Node{data_2311102013, 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_2311102013 << endl;
        temp = temp->next;
        index++;
    }
    if (index == 1) {
        cout << "Antrian 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;
}

```

Screenshots Output:

```

Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Antrian kosong
Jumlah antrian = 0
PS C:\Semester 2\Praktikum Strudur Data\Modul_7>

```


Deskripsi:

Program di atas merupakan implementasi dari struktur data Queue menggunakan Linked List. Program diatas memiliki fungsi isFull(): Mengecek apakah antrian penuh. Fungsi isEmpty(): Mengecek apakah antrian kosong. Fungsi enqueueAntrian(data): Menambahkan elemen baru jika antrian tidak penuh. Fungsi dequeueAntrian(): Menghapus elemen terdepan jika antrian tidak kosong. Fungsi countQueue(): Menghitung jumlah elemen dalam antrian. Fungsi clearQueue(): Menghapus semua elemen dalam antrian. Fungsi viewQueue(): Menampilkan isi antrian beserta posisi kosong. Pada fungsi main program akan menambahkan dua elemen ("Andi" dan "Maya") ke dalam antrian, menampilkan antrian dan jumlah elemennya, menghapus satu elemen, menampilkan antrian dan jumlah elemennya lagi, menghapus semua elemen, dan menampilkan antrian serta jumlah elemennya terakhir kali.

Unguided 2

```
#include <iostream>

using namespace std;

struct Node {
    string nama_2311102013;
    long long int nim_2311102013;
    Node* next;
};

Node* front = NULL;
Node* back = NULL;

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

void enqueueAntrian(string nama, long long int NIM) {
    Node* newNode = new Node;
    newNode->nama_2311102013 = nama;
    newNode->nim_2311102013 = NIM;
    newNode->next = NULL;

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

void dequeueAntrian() {
    if (isEmpty()) {
```

```

        cout << " Antrian kosong!" << endl;
        return;
    }

    Node* temp = front;
    front = front->next;
    delete temp;

    if (front == NULL) {
        back = NULL;
    }
}

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

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

void viewQueue() {
    if (isEmpty()) {
        cout << "Antrian kosong!" << endl;
        return;
    }

    cout << "Data antrian mahasiswa:" << endl;
    Node* current = front;
    int i = 1;
    while (current != NULL) {
        cout << i << ". " << current->nama_2311102013 << " (" <<
current->nim_2311102013 << ")" << endl;
        i++;
        current = current->next;
    }
}

int main() {

    enqueueAntrian("Raditya", 2311102013);
    enqueueAntrian("Wisnu", 231112913);
    enqueueAntrian("Rananta", 2311102813);
    enqueueAntrian("Putra", 2311102513);
    viewQueue();
    cout << "Jumlah antrian: " << countQueue() << endl;

    dequeueAntrian();
}

```

```

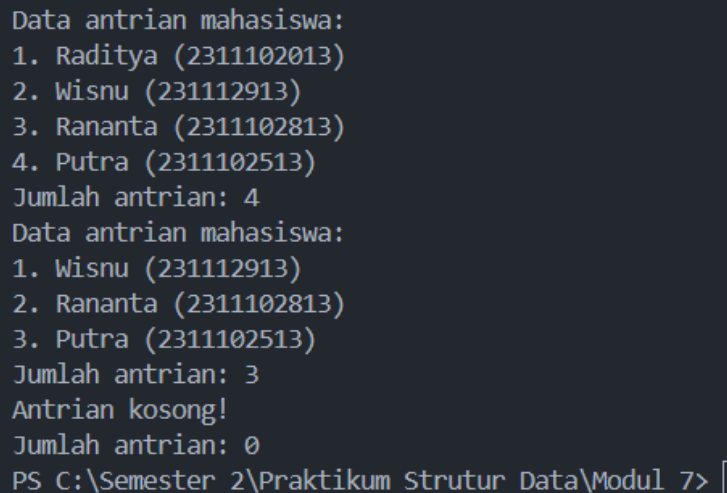
viewQueue();
cout << "Jumlah antrian: " << countQueue() << endl;

clearQueue();
viewQueue();
cout << "Jumlah antrian: " << countQueue() << endl;

return 0;
}

```

Screenshots Output:



```

Data antrian mahasiswa:
1. Raditya (2311102013)
2. Wisnu (231112913)
3. Rananta (2311102813)
4. Putra (2311102513)
Jumlah antrian: 4
Data antrian mahasiswa:
1. Wisnu (231112913)
2. Rananta (2311102813)
3. Putra (2311102513)
Jumlah antrian: 3
Antrian kosong!
Jumlah antrian: 0
PS C:\Semester 2\Praktikum Strukur Data\Modul 7>

```

Deskripsi:

Program di atas merupakan implementasi dari struktur data queue. Antrian ini menggunakan linked list untuk menyimpan data mahasiswa beserta nomor induk mahasiswa (NIM). Setiap elemen dalam antrian direpresentasikan oleh struktur Node yang memiliki data nama, NIM, dan pointer ke node berikutnya. Fungsi isEmpty: Memeriksa apakah antrian kosong. Fungsi enqueueAntrian: Menambahkan elemen baru ke dalam antrian. Fungsi dequeueAntrian: Menghapus elemen pertama dari antrian. Fungsi countQueue: Menghitung jumlah elemen dalam antrian. Fungsi clearQueue: Menghapus semua elemen dari antrian. Fungsi viewQueue: Menampilkan data mahasiswa yang sedang mengantri. Pada fungsi main, mahasiswa ditambahkan ke antrian menggunakan enqueueAntrian, kemudian antrian ditampilkan menggunakan viewQueue dan jumlahnya ditampilkan menggunakan countQueue. Setelah itu, satu elemen dihapus dari antrian menggunakan dequeueAntrian, lalu antrian dihapus seluruhnya dengan clearQueue.

BAB V

KESIMPULAN

Kesimpulan yang dapat saya ambil, Queue adalah struktur data yang menggunakan metode FIFO (First-In First-Out), di mana data yang pertama dimasukkan akan menjadi data yang pertama pula untuk dikeluarkan, mirip dengan konsep antrian dalam kehidupan sehari-hari. Implementasi dari queue dapat dilakukan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan back, di mana front adalah pointer ke elemen pertama dalam queue dan back adalah pointer ke elemen terakhir dalam queue. Perbedaan utama antara stack dan queue terletak pada aturan penambahan dan penghapusan elemen. Pada stack, penambahan dan penghapusan dilakukan di satu ujung (LIFO), sedangkan pada queue, operasi tersebut dilakukan di ujung yang berbeda (FIFO).

Operasi pada queue meliputi:

- enqueue(): menambahkan data ke dalam queue.
- dequeue(): mengeluarkan data dari queue.
- peek(): mengambil data dari queue tanpa menghapusnya.
- isEmpty(): mengecek apakah queue kosong.
- isFull(): mengecek apakah queue penuh.
- size(): menghitung jumlah elemen dalam queue.

BAB VI

DAFTAR PUSTAKA

[1] Asisten Praktikum. 2024. Modul 7 “Queue”. Diakses 27 Mei 2024, 19:00 WIB.
<https://lms.ittelkom-pwt.ac.id/>

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