

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



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

Disusun oleh:

HAIKAL SATRIATAMA

2311102066

IF-11-B

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

2024

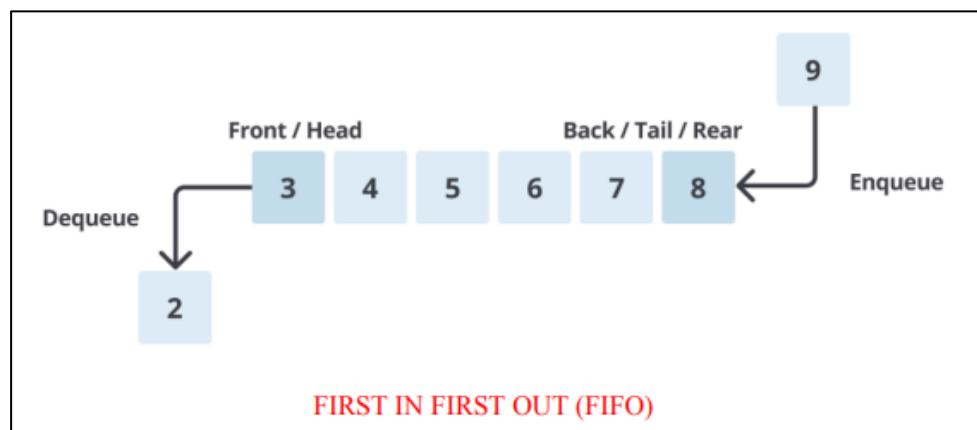
BAB I

DASAR TEORI

A. 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.



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 II

GUIDED

LATIHAN – GUIDED

1. Guided 1

Guided (berisi screenshot source code & output program disertai penjelasannya)

Source Code

```
#include <iostream>
using namespace std;

const int maksimalQueue = 5; // Maksimal antrian
int front = 0; // Penanda antrian
int back = 0; // Penanda belakang
string queueTeller[maksimalQueue]; // Array untuk menyimpan data antrian

// Fungsi pengecekan antrian penuh atau tidak
bool isFull() {
    return back == maksimalQueue;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return back == 0;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        queueTeller[back] = data;
        back++;
    }
}

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

```

        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("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

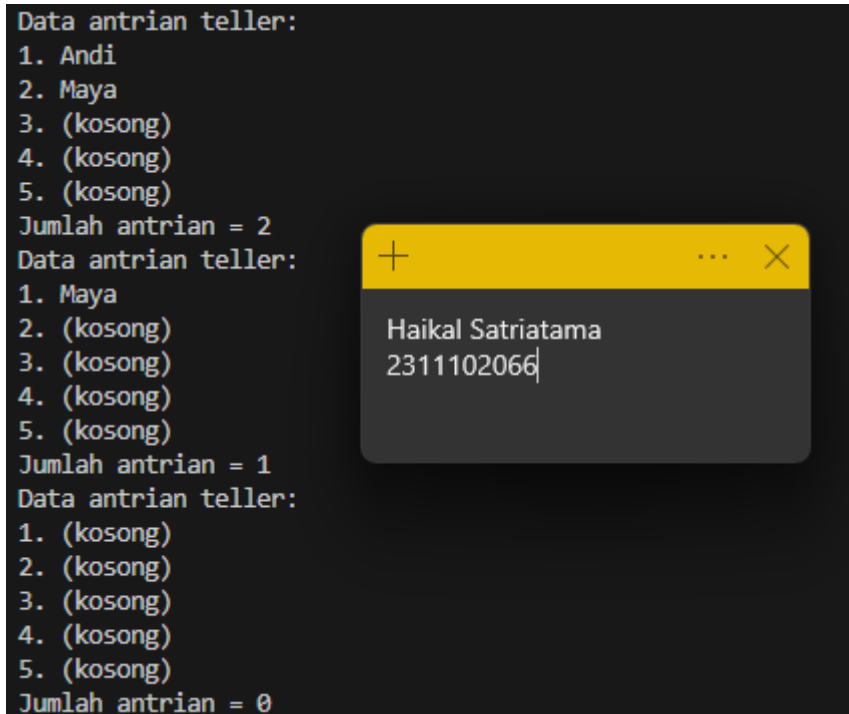
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    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 menyediakan beberapa fungsi untuk mengelola antrian, seperti `isFull()` untuk memeriksa apakah antrian sudah penuh, `isEmpty()` untuk memeriksa apakah antrian kosong, `enqueueAntrian()` untuk menambahkan data baru ke antrian, `dequeueAntrian()` untuk mengeluarkan data dari antrian, `countQueue()` untuk menghitung jumlah data dalam antrian, `clearQueue()` untuk mengosongkan seluruh antrian, dan `viewQueue()` untuk menampilkan isi antrian saat ini.

Dalam fungsi `main()`, program mendemonstrasikan penggunaan fungsi-fungsi tersebut. Pertama, program menambahkan dua data ("Andi" dan "Maya") ke dalam antrian menggunakan `enqueueAntrian()`, kemudian menampilkan isi antrian dengan `viewQueue()`. Setelah itu, program menghitung jumlah data dalam antrian dengan `countQueue()`, mengeluarkan satu data dari antrian dengan `dequeueAntrian()`, menampilkan isi antrian yang baru, dan menghitung jumlah data lagi. Terakhir, program mengosongkan seluruh antrian dengan `clearQueue()`, menampilkan isi antrian kosong, dan menghitung jumlah data yang tersisa.

BAB III

UNGUIDED

TUGAS – UNGUIDED

1. Unguided 1

Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

Source Code

```
#include <iostream>
using namespace std;

// Struktur node untuk linked list
struct Node {
    string data;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian penuh atau tidak (tidak relevan untuk
// linked list karena tidak ada batasan kapasitas)
bool isFull() {
    return false;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
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;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
```

```

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

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

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". " << temp->data << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("bonek");
    enqueueAntrian("abengg");
}

```



```

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

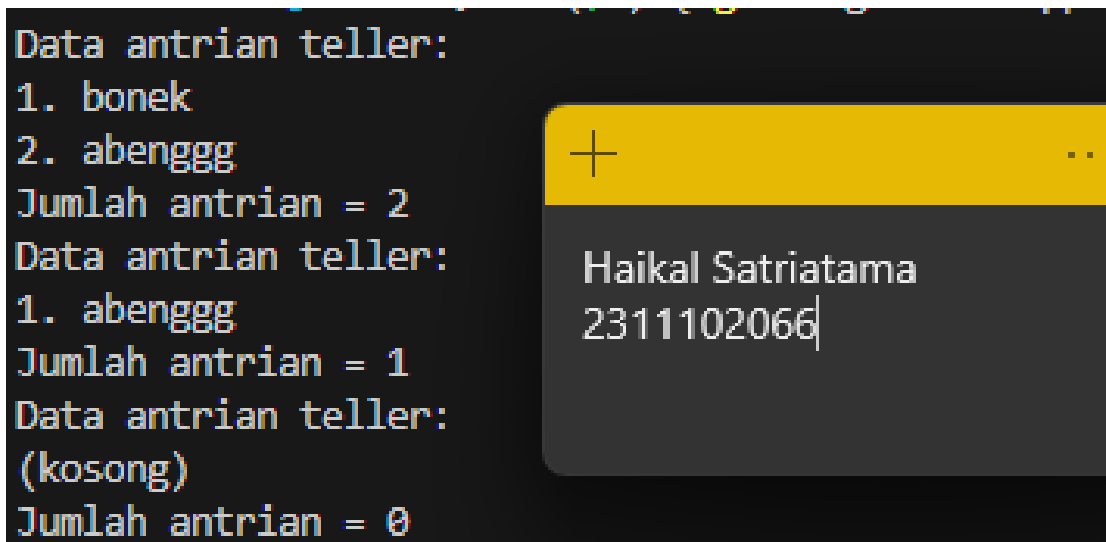
dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

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

return 0;
}

```

Screenshot Program



Deskripsi program

Selanjutnya, program mendeklarasikan dua pointer yaitu "front" dan "back", yang digunakan untuk melacak node pertama dan node terakhir dalam antrian. Beberapa fungsi juga didefinisikan untuk melakukan operasi pada antrian ini, seperti memeriksa apakah antrian kosong atau tidak, menambahkan data baru ke antrian, menghapus data dari antrian, menghitung jumlah data dalam antrian, menghapus semua data dari antrian, dan melihat isi antrian.

Di dalam fungsi main, program mendemonstrasikan penggunaan fungsi-fungsi tersebut. Pertama, dua string "bonek" dan "abenggg" ditambahkan ke dalam antrian. Kemudian, isi antrian ditampilkan dengan memanggil fungsi viewQueue(). Setelah itu,

jumlah data dalam antrian dihitung dan ditampilkan menggunakan fungsi `countQueue()`.

Selanjutnya, data pertama dalam antrian dihapus dengan memanggil fungsi `dequeueAntrian()`, dan isi antrian yang baru ditampilkan kembali. Jumlah data dalam antrian yang baru juga dihitung dan ditampilkan.

2. Guided 2

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

Source code

```
#include <iostream>
using namespace std;

// Struktur node untuk linked list dengan Nama dan NIM
struct Node {
    string nama;
    string NIM;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
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;
    }
}
```

```

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

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

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". Nama: " << temp->nama << ", NIM: " << temp->NIM << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

```

```

int main() {
    enqueueAntrian("bonek", "2311102066");
    enqueueAntrian("abengg", "12345");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

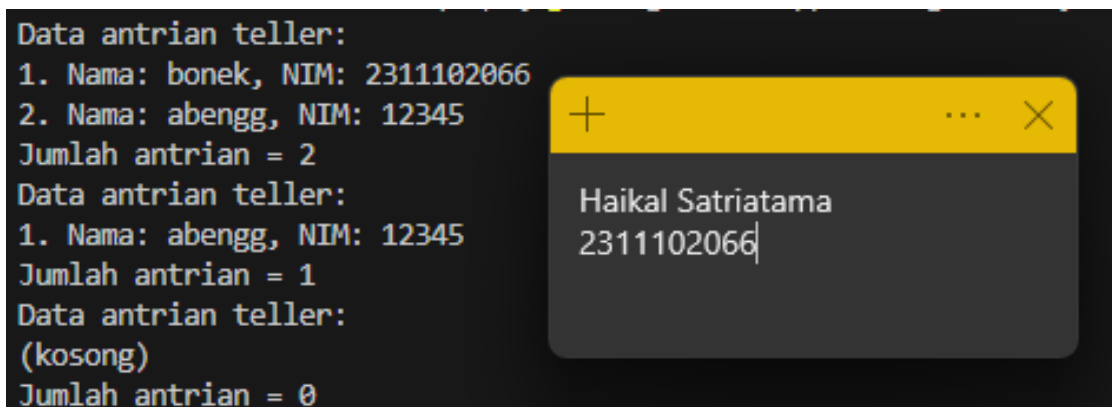
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    return 0;
}

```

Screenshoot program



Deskripsi program

Program ini menggunakan konsep struktur data Linked List untuk membuat sebuah sistem antrian. Struktur Node digunakan untuk menyimpan data nama dan NIM dari setiap elemen dalam antrian. Terdapat dua penanda yaitu front untuk menunjuk pada elemen paling depan dan back untuk menunjuk pada elemen paling belakang dalam antrian.

Fungsi isEmpty() digunakan untuk memeriksa apakah antrian kosong atau tidak dengan cara memeriksa apakah penanda front bernilai nullptr atau tidak. Fungsi enqueueAntrian() digunakan untuk menambahkan elemen baru ke dalam antrian dengan cara membuat Node baru dan menambahkannya ke akhir antrian. Jika antrian masih kosong, Node baru akan menjadi elemen pertama dan penanda front dan back akan menunjuk ke Node tersebut.

Fungsi `dequeueAntrian()` digunakan untuk menghapus elemen paling depan dari antrian dengan cara memindahkan penanda `front` ke elemen berikutnya dan menghapus elemen yang ditunjuk oleh `front` sebelumnya. Jika antrian hanya memiliki satu elemen, maka penanda `back` juga akan diatur menjadi `nullptr`. Fungsi `countQueue()` digunakan untuk menghitung banyaknya elemen dalam antrian dengan cara melakukan iterasi dari elemen paling depan hingga elemen paling belakang.

Fungsi `clearQueue()` digunakan untuk menghapus seluruh elemen dalam antrian dengan cara terus memanggil fungsi `dequeueAntrian()` hingga antrian menjadi kosong. Fungsi `viewQueue()` digunakan untuk mencetak seluruh elemen dalam antrian beserta nomor urutannya.

BAB IV

KESIMPULAN

Queue adalah suatu susunan data linear yang mengikuti aturan antrian First In, First Out (FIFO), di mana data yang lebih dulu dimasukkan akan lebih dulu dikeluarkan. Operasi-operasi utama pada queue mencakup menambahkan data di akhir (enqueue), menghapus data dari awal (dequeue), memeriksa apakah queue kosong, memeriksa apakah queue sudah penuh (untuk array), menghitung banyaknya data, membersihkan semua data, dan menampilkan seluruh data.

Queue dapat diimplementasikan dengan menggunakan array atau linked list. Implementasi menggunakan array memiliki kapasitas tetap dan membutuhkan pengelolaan posisi data, sedangkan linked list memungkinkan penambahan dan penghapusan data secara dinamis tanpa batasan kapasitas. Masing-masing metode implementasi memiliki kelebihan dan kekurangan sendiri. Array lebih sederhana dan cepat dalam mengakses data tetapi terbatas kapasitasnya dan memerlukan pergeseran data saat dequeue, sementara linked list lebih fleksibel dalam kapasitas dan tidak membutuhkan pergeseran data, namun memerlukan lebih banyak memori dan sedikit lebih rumit.

DAFTAR PUSTAKA

Modul 7 Queue, Learning Management System, Asisten Praktikum. 2024