

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



**Disusun Oleh :
WIDARI DWI HAYATI
2311102060**

**Dosen :
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.**

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

A. Dasar Teori

Queue adalah sebuah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Artinya, data yang pertama kali dimasukkan ke dalam queue akan menjadi data yang pertama kali dikeluarkan dari queue. Konsep ini mirip dengan antrian yang sering kita temui dalam kehidupan sehari-hari, di mana orang 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 atau head adalah pointer ke elemen pertama dalam queue, sedangkan rear atau tail atau back adalah pointer ke elemen terakhir dalam queue.

Perbedaan antara stack dan queue terletak pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan pada satu ujung. Elemen yang terakhir dimasukkan akan berada paling depan atau dianggap sebagai elemen paling atas, sehingga pada operasi penghapusan, elemen teratas yang dimasukkan terakhir akan dihapus terlebih dahulu. Hal ini dikenal dengan sifat LIFO (Last-In First-Out). Sedangkan pada queue, operasi tersebut dilakukan pada tempat berbeda, yaitu melalui salah satu ujung. Hal ini disebabkan karena perubahan data selalu mengacu pada head, sehingga hanya ada satu jenis insert maupun delete. Prosedur ini sering disebut Enqueue dan Dequeue pada kasus queue. Untuk Enqueue, cukup menambahkan 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.

B. Guided

Source Code

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

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

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

// Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty() {
    return back == 0;
}

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

//Fungsi untuk menghapus elemen dari 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] = ""; //Membersihkan data terakhir
        back--;
    }
}

//Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue(){
    return back;
}

//Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    for (int i = 0; i < back; i++) {
```

```

        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

//Fungsi untuk menampilkan semua elemen dalam 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;

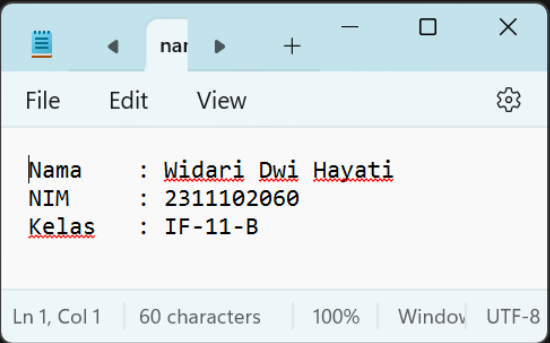
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    return 0;
}

```

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:\Users\ADVAN>
```



The screenshot shows a Notepad window titled 'nar' with a menu bar (File, Edit, View) and a status bar (Ln 1, Col 1 | 60 characters | 100% | Window | UTF-8). The text inside the window is:

```
Nama      : Widari Dwi Hayati
NIM       : 2311102060
Kelas    : IF-11-B
```

Deskripsi:

Program di atas merupakan implementasi dari struktur data queue menggunakan array. Di awal program, terdefinisi beberapa variabel global, yaitu `maksimalQueue`: maksimal kapasitas queue, yang diinisialisasi dengan nilai 5. `front`: penanda untuk elemen pertama di queue. `back`: penanda untuk elemen terakhir di queue. `queueTeller`: array untuk menyimpan elemen-elemen di queue.

Selain itu, program ini juga memiliki beberapa fungsi, yaitu `isFull()`: fungsi untuk memeriksa apakah queue sudah penuh. `isEmpty()`: fungsi untuk memeriksa apakah queue kosong. `enqueueAntrian(string data)`: fungsi untuk menambahkan elemen ke queue. `dequeueAntrian()`: fungsi untuk menghapus elemen dari queue. `countQueue()`: fungsi untuk menghitung jumlah elemen di queue. `clearQueue()`: fungsi untuk mengosongkan semua elemen di queue. `viewQueue()`: fungsi untuk menampilkan semua elemen di queue.

Di bagian `main()`, program menggunakan beberapa fungsi tersebut untuk mengoperasikan queue. Pertama, program menambahkan dua elemen ke queue menggunakan `enqueueAntrian()`. Kemudian, program menampilkan isi queue menggunakan `viewQueue()` dan menghitung jumlah elemen di queue menggunakan `countQueue()`. Setelah itu, program menghapus salah satu elemen di queue menggunakan `dequeueAntrian()`. Selanjutnya, program mengosongkan semua elemen di queue menggunakan `clearQueue()` dan kembali menampilkan isi queue menggunakan `viewQueue()`.

C. Unguided/Tugas

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;
};

Node* front = nullptr; // Penanda antrian
Node* back = nullptr; // Penanda
int maksimalQueue = 5; // Maksimal antrian

bool isFull() { // Pengecekan antrian penuh atau tidak
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count == maksimalQueue;
}

// Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string data) {
    if (isFull()) {
        cout << " Antrian penuh" << endl;
    } else {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;
        if (front == nullptr) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }
}
```

```

}

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

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

//Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    while (front != nullptr) {
        Node* temp = front;
        front = front->next;
        delete temp;
    }
    back = nullptr;
}

//Fungsi untuk menampilkan semua elemen dalam 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++;
    }
}

```

```

}

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

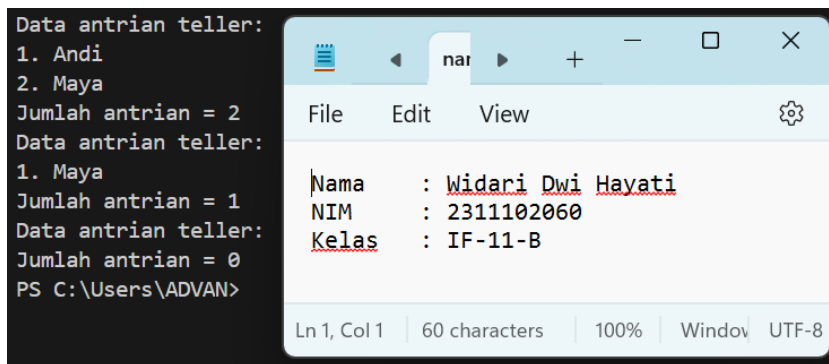
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    return 0;
}

```

Screenshots Output



The screenshot shows two windows. On the left is a terminal window with the following output:

```

Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Jumlah antrian = 0
PS C:\Users\ADVAN>

```

On the right is a Notepad++ window titled 'nar' containing the following text:

```

Nama      : Widari Dwi Hayati
NIM       : 2311102060
Kelas    : IF-11-B

```

The Notepad++ window also shows a status bar at the bottom indicating 'Ln 1, Col 1', '60 characters', '100%', 'Window', and 'UTF-8'.

Deskripsi:

Program di atas merupakan implementasi dari struktur data queue menggunakan linked list. Di awal program, terdefinisi beberapa variabel global, yaitu Node: struktur data yang digunakan untuk menyimpan elemen di queue. Setiap node terdiri dari dua bagian, yaitu data dan pointer ke node berikutnya. front: penanda untuk elemen pertama di queue. back: penanda untuk elemen terakhir di queue. maksimalQueue: maksimal kapasitas queue, yang diinisialisasi dengan nilai 5.

Selain itu, program ini juga memiliki beberapa fungsi, yaitu `isFull()`: fungsi untuk memeriksa apakah queue sudah penuh. `isEmpty()`: fungsi untuk memeriksa apakah queue kosong. `enqueueAntrian(string data)`: fungsi untuk menambahkan elemen ke queue. `dequeueAntrian()`: fungsi untuk menghapus elemen dari queue. `countQueue()`: fungsi untuk menghitung jumlah elemen di queue. `clearQueue()`: fungsi untuk mengosongkan semua elemen di queue. `viewQueue()`: fungsi untuk menampilkan semua elemen di queue.

Di bagian `main()`, program menggunakan beberapa fungsi tersebut untuk mengoperasikan queue. Pertama, program menambahkan dua elemen ke queue menggunakan `enqueueAntrian()`. Kemudian, program menampilkan isi queue menggunakan `viewQueue()` dan menghitung jumlah elemen di queue menggunakan `countQueue()`. Setelah itu, program menghapus salah satu elemen di queue menggunakan `dequeueAntrian()`. Selanjutnya, program mengosongkan semua elemen di queue menggunakan `clearQueue()` dan kembali menampilkan isi queue menggunakan `viewQueue()`.

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;
};

Node* front = nullptr; // Penanda antrian
Node* back = nullptr; // Penanda
int maksimalQueue = 5; // Maksimal antrian

bool isFull() { // Pengecekan antrian penuh atau tidak
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count == maksimalQueue;
}

// Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string nama, string nim) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (front == nullptr) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }
}
```

```

    }
}

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

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

//Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    while (front != nullptr) {
        Node* temp = front;
        front = front->next;
        delete temp;
    }
    back = nullptr;
}

//Fungsi untuk menampilkan semua elemen dalam antrian
void viewQueue() {
    cout << "_____ " << endl;
    cout << "| No. | Nama Mahasiswa \t\t | NIM Mahasiswa |" << endl;
    cout << "_____ " << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {

```

```

        cout << "|" << i << ". | " << temp->nama << "\t | " << temp->nim << "\t |" << endl;
        cout << " _____ " << endl;
        temp = temp->next;
        i++;
    }
}

int main() {
    string nama, nim;
    while(true) {
        int pilih;
        cout << "=====ANTRIAN MAHASISWA===== " <<
endl;
        cout << "\n1. Menambahkan data antrian mahasiswa.";
        cout << "\n2. Menghapus data antrian mahasiswa.";
        cout << "\n3. Melihat jumlah data antrian mahasiswa.";
        cout << "\n4. Menghapus seluruh data antrian mahasiswa.";
        cout << "\n5. Menampilkan data antrian mahasiswa.";
        cout << "\n6. Exit.";
        cout << "\n\nMasukkan pilihan : ";
        cin >> pilih;
        cin.ignore();
        switch (pilih)
        {
            case 1:
                cout << "Masukkan nama mahasiswa : ";
                getline(cin, nama);
                cout << "Masukkan NIM mahasiswa : ";
                getline(cin, nim);
                enqueueAntrian(nama, nim);
                cout << endl;
                break;
            case 2:
                dequeueAntrian();
                cout << endl;
                break;
            case 3:
                cout << "Jumlah antrian = " << countQueue() << endl;
                break;
            case 4:
                clearQueue();
                cout << endl;
                break;
            case 5:
                viewQueue();
                cout << endl;
                break;

```

```

        case 6:
            return 0;
            break;
        default:
            cout << "Pilihan Anda tidak valid";
            break;
        }
    }

    return 0;
}

```

Screenshots Output

```

=====ANTRIAN MAHASISWA=====
1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 1
Masukkan nama mahasiswa : Marshely Ayu Iswanto
Masukkan NIM mahasiswa : 2311102071

=====ANTRIAN MAHASISWA=====
1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 1
Masukkan nama mahasiswa : Widari Dwi Hayati
Masukkan NIM mahasiswa : 2311102060

```

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 1

Masukkan nama mahasiswa : D'Sharlendita F. A.

Masukkan NIM mahasiswa : 2311102069

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 5

No.	Nama Mahasiswa	NIM Mahasiswa
1.	Marshely Ayu Iswanto	2311102071
2.	Widari Dwi Hayati	2311102060
3.	D'Sharlendita F. A.	2311102069

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 1, Col 160 characters100%WindowUTF-8

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 3

Jumlah antrian = 3

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 2

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 1, Col 160 characters100%WindowUTF-8

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 5

No.	Nama Mahasiswa	NIM Mahasiswa
1.	Widari Dwi Hayati	2311102060
2.	D'Sharlendita F. A.	2311102069

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 3

Jumlah antrian = 2

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 1, Col 1 | 60 characters | 100% | Window | UTF-8

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 4

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 5

No.	Nama Mahasiswa	NIM Mahasiswa
-----	----------------	---------------

=====ANTRIAN MAHASISWA=====

1. Menambahkan data antrian mahasiswa.
2. Menghapus data antrian mahasiswa.
3. Melihat jumlah data antrian mahasiswa.
4. Menghapus seluruh data antrian mahasiswa.
5. Menampilkan data antrian mahasiswa.
6. Exit.

Masukkan pilihan : 6

PS C:\Users\ADVAN>

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 1, Col 1 | 60 characters | 100% | Window | UTF-8

Deskripsi:

Program di atas merupakan implementasi dari struktur data queue menggunakan linked list. Program ini menyediakan beberapa fitur, yaitu menambahkan data mahasiswa ke antrian, menghapus data mahasiswa dari antrian, melihat jumlah data mahasiswa di antrian, menghapus semua data mahasiswa di antrian, dan menampilkan data mahasiswa di antrian.

Di awal program, terdefinisi beberapa variabel global, yaitu Node: struktur data yang digunakan untuk menyimpan elemen di queue. Setiap node terdiri dari tiga bagian, yaitu nama, NIM, dan pointer ke node berikutnya. front: penanda untuk elemen pertama di queue. back: penanda untuk elemen terakhir di queue. maksimalQueue: maksimal kapasitas queue, yang diinisialisasi dengan nilai 5.

Selain itu, program ini juga memiliki beberapa fungsi, yaitu isFull(): fungsi untuk memeriksa apakah queue sudah penuh. isEmpty(): fungsi untuk memeriksa apakah queue kosong. enqueueAntrian(string nama, string nim): fungsi untuk menambahkan elemen ke queue. dequeueAntrian(): fungsi untuk menghapus elemen dari queue. countQueue(): fungsi untuk menghitung jumlah elemen di queue. clearQueue(): fungsi untuk mengosongkan semua elemen di queue. viewQueue(): fungsi untuk menampilkan semua elemen di queue.

Di bagian main(), program menampilkan menu pilihan dan menggunakan beberapa fungsi tersebut untuk mengoperasikan queue. Program akan terus berjalan hingga pengguna memilih untuk keluar dengan memasukkan pilihan 6.

D. Kesimpulan

Queue adalah struktur data yang mengorganisir data dengan metode FIFO (First-In First-Out), yang berarti data yang pertama dimasukkan akan pertama kali dikeluarkan. Queue dapat diimplementasikan menggunakan array atau linked list, dan terdiri dari dua pointer, yaitu front dan rear.

Perbedaan antara stack dan queue terletak pada aturan penambahan dan penghapusan elemen. Pada stack, operasi terjadi pada satu ujung, sedangkan pada queue, operasi tersebut dilakukan pada ujung yang berbeda. Prosedur pada queue disebut Enqueue dan Dequeue, yang menambahkan dan mengeluarkan elemen dari queue.

Operasi lain yang dapat dilakukan pada queue adalah peek(), isEmpty(), isFull(), dan size(). Peek() digunakan untuk mengambil data dari queue tanpa menghapusnya, isEmpty() untuk mengecek apakah queue kosong atau tidak, isFull() untuk mengecek apakah queue penuh atau tidak, dan size() untuk menghitung jumlah elemen dalam queue.

E. Referensi (APA)

Stroustrup, B. (2021). A Tour of C++ (2nd ed.). Addison-Wesley Professional.

Lippman, S. L., Lajoie, J., & Moo, B. (2020). C++ Primer (6th ed.). Addison-Wesley Professional.

Malik, D. S. (2014). C++ Programming: From Problem Analysis to Program Design (7th ed.). Cengage Learning.