LAPORAN PRAKTIKUM
STURKTUR DATA DAN ALGORITMA

MODUL VII
QUEUE

Disusun Oleh :
Muhammad Dani Ayubi
2311102003

Dosen Pengmapu:
Wahyu Andi Syahputra, S.pd.,M.Eng

LABORATORIUM MULTIMEDIA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO
2024

BAB I
TUJUAN PRAKTIKUM

a. Mahasiswa mampu menjelaskan definisi dan konsep dari Double Queque
b. Mahasiswa mampu menerapkan operasi tamba, menghapus pada Queque
c. Mahasiswa mampu menerapkan operasi tampil data pada queque

BAB II
DASAR TEORI

Queue adalah struktur data linear yang mengikuti prinsip "First-In First-Out" (FIFO), artinya elemen yang pertama kali masuk akan menjadi elemen pertama yang keluar. Mirip seperti antrian dalam kehidupan sehari-hari, elemen baru ditambahkan di bagian belakang (rear) dan elemen yang ada di bagian depan (front) akan dikeluarkan terlebih
dahulu.

Operasi pada Queue:

- Enqueue: Menambahkan elemen baru ke dalam antrian pada posisi rear.
- Dequeue: Menghapus elemen dari antrian pada posisi front.
- Front: Mendapatkan elemen yang berada di posisi front tanpa menghapusnya.
- Rear: Mendapatkan elemen yang berada di posisi rear tanpa menghapusnya.
- IsEmpty: Memeriksa apakah antrian kosong atau tidak.
- Size: Mengembalikan jumlah elemen yang ada dalam antrian.

Representasi Queue:

- Array: Queue dapat diimplementasikan menggunakan array dengan dua variabel penunjuk, yaitu front dan rear. Front menunjukkan posisi elemen terdepan dalam antrian, sedangkan rear menunjukkan posisi elemen terbelakang dalam antrian. Ketika elemen ditambahkan atau dihapus, penunjuk front atau rear akan bergerak sesuai dengan operasi yang dilakukan.
- Linked List: Queue juga dapat diimplementasikan menggunakan linked list, di mana setiap elemen dalam antrian memiliki referensi ke elemen berikutnya.

Keuntungan dan Penggunaan Queue:

- Antrian sangat berguna dalam situasi-situasi di mana elemen harus diproses berdasarkan urutan kedatangan mereka.
- Antrian sering digunakan dalam masalah penjadwalan, pemrosesan paralel, simulasi, pengolahan sinyal, dan berbagai aplikasi lainnya yang melibatkan pengolahan data secara berurutan.
- Struktur data antrian memungkinkan akses cepat ke elemen pertama dan terakhir, serta operasi enkripsi dan dekripsi yang efisien pada data.

BAB III

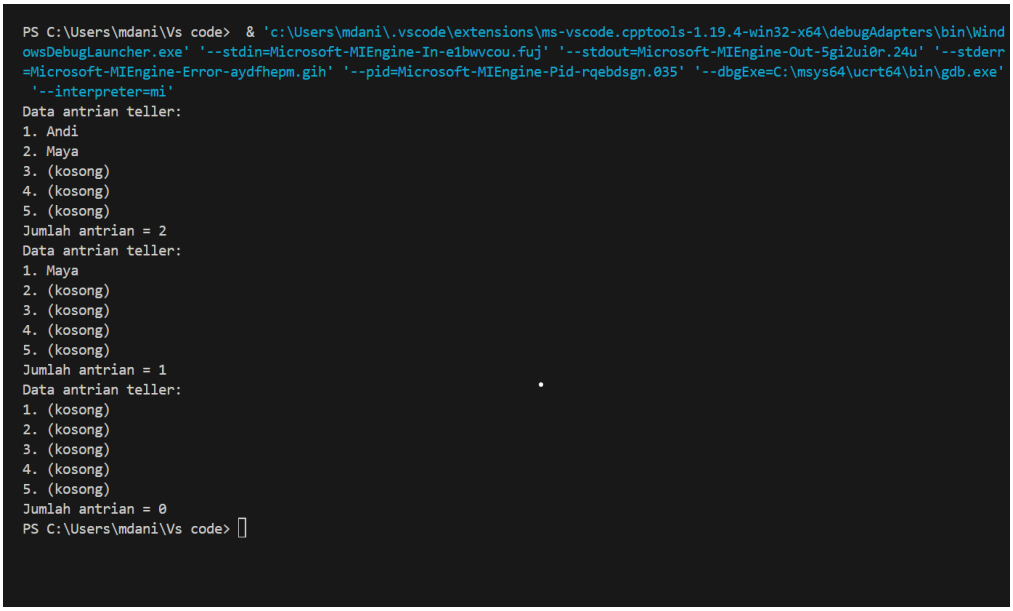
GUIDED

1. Guided 1

Source code

```
```

Screenshoot program



Deskripsi program

Kode di atas adalah implementasi konsep antrian (queue) menggunakan array pada C++. Kode tersebut memiliki beberapa fungsi untuk melakukan operasi pada antrian, seperti menambahkan elemen (enqueue), menghapus elemen (dequeue), menghitung jumlah elemen dalam antrian (countQueue) menghapus semua elemen dalam antrian (clearQueue), dan melihat elemen elemen dalam antrian (viewQueue).

LATIHAN KELAS - UNGUIDED

1. Unguided 1

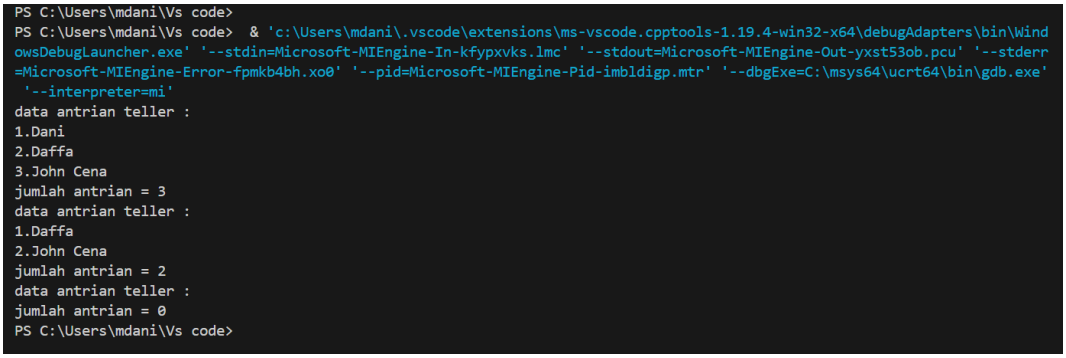
Source code

```
#include <iostream>
using namespace std;
struct Node
{
    string data;
    Node *next;
};
Node *front = NULL;
Node *back = NULL;
bool isEmpty()
{
    return (front == NULL);
}
void enqueueAntrian(string data)
{
    Node *temp = new Node();
    temp->data = data;
    temp->next = NULL;
    if (front == NULL && back == NULL)
    {
        front = back = temp;
        return;
    }
    back->next = temp;
    back = temp;
}
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
        return;
    }
    Node *temp = front;
    if (front == back)
    {
        front = back = NULL;
    }
    else
    {
```

```
        front = front->next;
    }
    delete temp;
}
int countQueue()
{
    int count = 0;
    Node *temp = front;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    return count;
}
void clearQueue()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
        return;
    }
    Node *temp = front;
    while (temp != NULL)
    {
        front = front->next;
        delete temp;
        temp = front;
    }
}
void viewQueue()
{
    cout << "data antrian teller : " << endl;
    Node *temp = front;
    int i = 1;
    while (temp != NULL)
    {
        cout << i << "." << temp->data << endl;
        i++;
        temp = temp->next;
    }
}
int main()
{
    enqueueAntrian("Dani");
    enqueueAntrian("Daffa");
enqueueAntrian("John Cena");
```

```
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 C++ di atas adalah implementasi struktur data queue menggunakan linked list. Program ini juga mensimulasikan antrian teller dengan menggunakan linked list sebagai representasi queue. Di setiap elemen antrian akan direpresentasikan sebagai node dalam linked list, dengan memiliki data dan pointer ke node berikutnya. Fungsi “enqueue” digunakan untuk membuat node baru dan mengatur pointer rear untuk menunjuk ke node baru tersebut. Fungsi “dequeue” ini akan menghapus node pertama dalam linked list dengan mengatur pointer front ke node berikutnya. Kemudian untuk “Count” sendiri digunakan untuk melacak jumlah elemen dalam antrian

2. Unguided 2

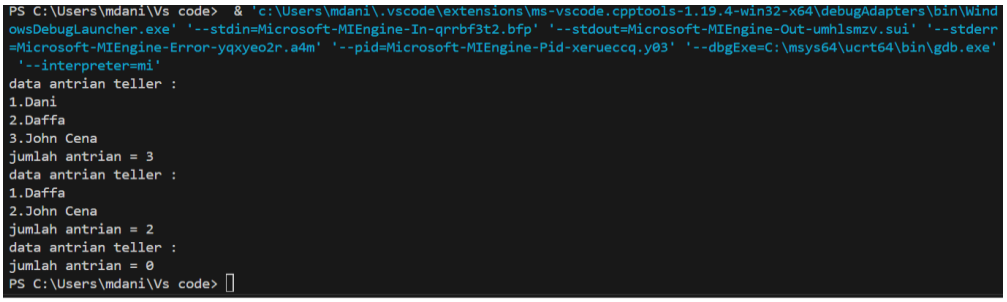
Source code

```
#include <iostream>
using namespace std;
struct Node
{
    string data;
    Node *next;
};
Node *front = NULL;
Node *back = NULL;
bool isEmpty()
{
    return (front == NULL);
}
void enqueueAntrian(string data)
{
    Node *temp = new Node();
    temp->data = data;
    temp->next = NULL;
    if (front == NULL && back == NULL)
    {
        front = back = temp;
        return;
    }
    back->next = temp;
    back = temp;
}
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
        return;
    }
    Node *temp = front;
    if (front == back)
    {
        front = back = NULL;
    }
    else
    {
        front = front->next;
    }
    delete temp;
}
```

```
int countQueue()
{
    int count = 0;
    Node *temp = front;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    return count;
}
void clearQueue()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
        return;
    }
    Node *temp = front;
    while (temp != NULL)
    {
        front = front->next;
        delete temp;
        temp = front;
    }
}
void viewQueue()
{
    cout << "data antrian teller : " << endl;
    Node *temp = front;
    int i = 1;
    while (temp != NULL)
    {
        cout << i << "." << temp->data << endl;
        i++;
        temp = temp->next;
    }
}
int main()
{
    enqueueAntrian("Muhammad Dani Ayubi, NIM : 2311102003");
    enqueueAntrian("Daffa Maulana, NIM : 2311102004");
enqueueAntrian("John Cena, NIM :2311102005");
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 hampir sama persis dengan program unguided 1, yang berbeda hanya ditambahkannya atribut NIM pada linkedlist.

BAB IV

KESIMPULAN

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. Queue adalah struktur data yang mengikuti prinsip FIFO (First-In First-Out), artinya elemen yang pertama kali dimasukkan ke dalam queue akan menjadi yang pertama kali dihapus.Stack dapat diimplementasikan menggunakan array atau linked list. Dalam praktikum ini, implementasi stack menggunakan array dan linked list.
2. Dalam praktikum ini, kita mempelajari dua cara implementasi queue, yaitu menggunakan array dan linked list. Pada implementasi menggunakan array, kita menggunakan dua pointer, yaitu front dan back, untuk menunjukkan posisi awal dan akhir antrian. Kami mengimplementasikan fungsi-fungsi dasar seperti enqueue (menambahkan elemen ke antrian), dequeue (menghapus elemen dari antrian), isEmpty (memeriksa apakah antrian kosong), isFull (memeriksa apakah antrian penuh), dan beberapa fungsi lainnya untuk mengelola dan mengoperasikan antrian.
3. Pada implementasi menggunakan linked list, kita menggunakan node dan pointer front dan back untuk menunjukkan posisi awal dan akhir antrian. Fungsi-fungsi dasar seperti enqueue, dequeue, isEmpty, clearQueue, dan viewQueue diimplementasikan menggunakan operasi-operasi linked list seperti penambahan dan penghapusan node.
4. Dalam kedua implementasi, kita juga mempelajari fungsi-fungsi tambahan seperti countQueue untuk menghitung jumlah elemen dalam antrian.
5. Praktikum queue sangat berguna dalam situasi di mana data harus diproses secara berurutan dan dalam urutan mereka tiba.