

**LAPORAN PRAKTIKUM**  
**Queue**  
**Bagian Pertama**



**Disusun Oleh:**

**Ryan Gabriel Togar Simamora (2311104045)**

**Kelas : SE0702**

**Dosen :**

**Wahyu Andi Saputra**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024**

## I. Tujuan

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

## II. Landasan Teori

### A.Queue

Apa itu Queue?

Queue adalah struktur data yang bekerja seperti antrian di kehidupan nyata. Siapa yang datang duluan, dia yang dilayani duluan. Ini disebut metode FIFO (First-In, First-Out). Misalnya, saat kamu antri di kantin: orang pertama yang datang akan dilayani lebih dulu, dan yang terakhir masuk antrian harus menunggu.

### Cara Kerja Queue

Queue punya dua ujung utama:

- ❖ Front (depan): Tempat data dikeluarkan (dilayani).
- ❖ Rear (belakang): Tempat data baru dimasukkan.

### Perbedaan Queue dan Stack

- ❖ Stack: Data dimasukkan dan dikeluarkan di satu ujung (LIFO – Last-In, First-Out). Contohnya, tumpukan piring: piring yang ditaruh terakhir akan diambil duluan.
- ❖ Queue: Data masuk dari belakang (rear) dan keluar dari depan (front), sesuai urutan yang masuk pertama.

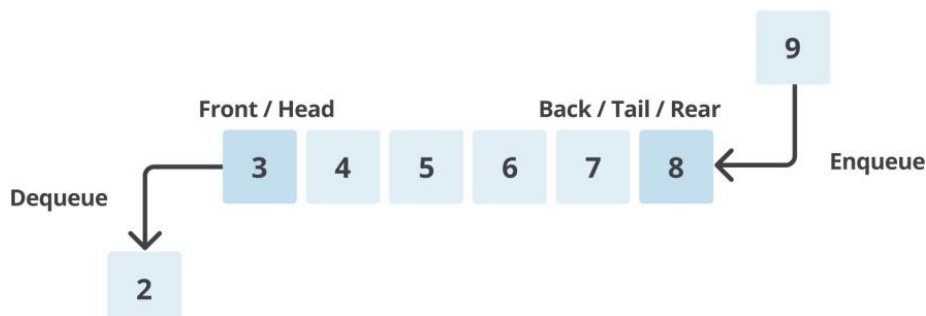
### Operasi pada Queue

- ❖ Enqueue: Menambahkan data ke belakang (rear).
- ❖ Dequeue: Menghapus data dari depan (front).
- ❖ Peek: Melihat data di depan tanpa menghapusnya.
- ❖ isEmpty: Mengecek apakah queue kosong.
- ❖ isFull: Mengecek apakah queue penuh (biasanya berlaku untuk array).
- ❖ Size: Menghitung berapa banyak data di queue.

### Implementasi Queue

Queue bisa dibuat dengan:

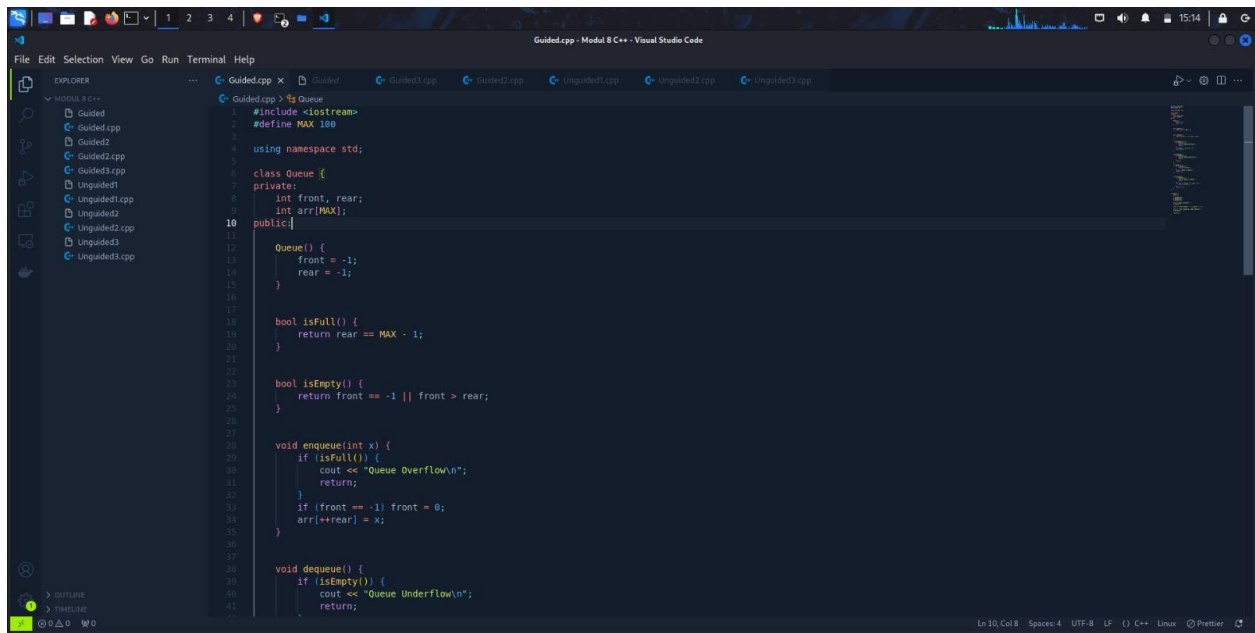
- ❖ Array: Kapasitas tetap, ada batas jumlah data yang bisa disimpan.
- ❖ Linked List: Lebih fleksibel, data bisa terus ditambah selama memori cukup.



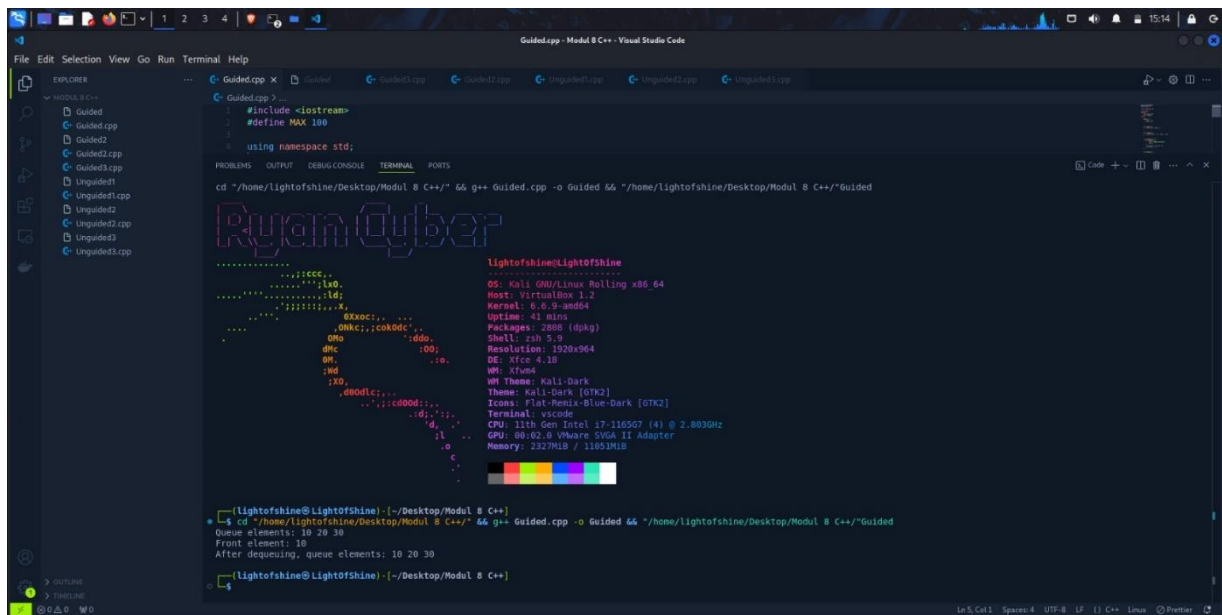
### Contoh dalam Kehidupan Nyata

Antrean kasir: Orang pertama yang masuk antrian akan dilayani lebih dulu, dan yang terakhir masuk harus menunggu giliran.

## File Guided1.cpp



## Outputnya



Untuk Source Codenya Lebih Lengkap dibawah ini :

### **Guided1.cpp**

```
#include <iostream>
#define MAX 100

using namespace std;

class Queue {
private:
    int front, rear;
    int arr[MAX];
public:

    Queue() {
        front = -1;
        rear = -1;
    }

    bool isFull() {
        return rear == MAX - 1;
    }

    bool isEmpty() {
        return front == -1 || front > rear;
    }

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue Overflow\n";
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow\n";
            return;
        }
        front++;
    }
}
```

```

int peek() {
    if (!isEmpty()) {
        return arr[front];
    }
    cout << "Queue is empty\n";
    return -1;
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty\n";
        return;
    }
    for (int i = front; i <= rear; i++) {
        cout << arr[i] << " ";
    }
    cout << "\n";
}
};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Queue elements: ";
    q.display();

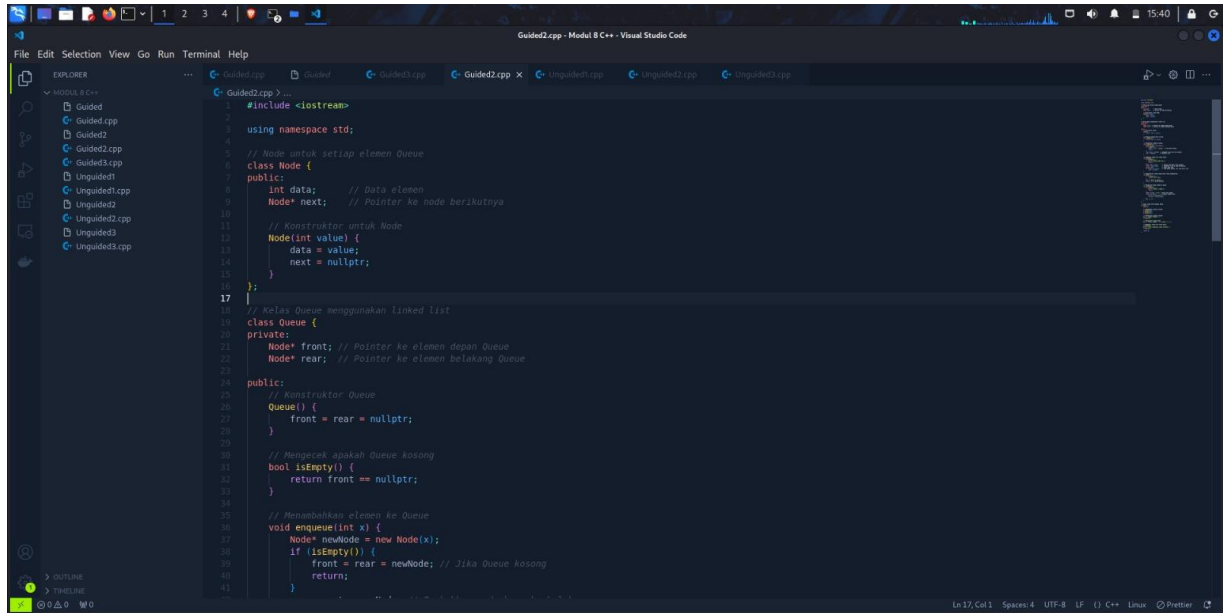
    cout << "Front element: " << q.peek() << "\n";

    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}

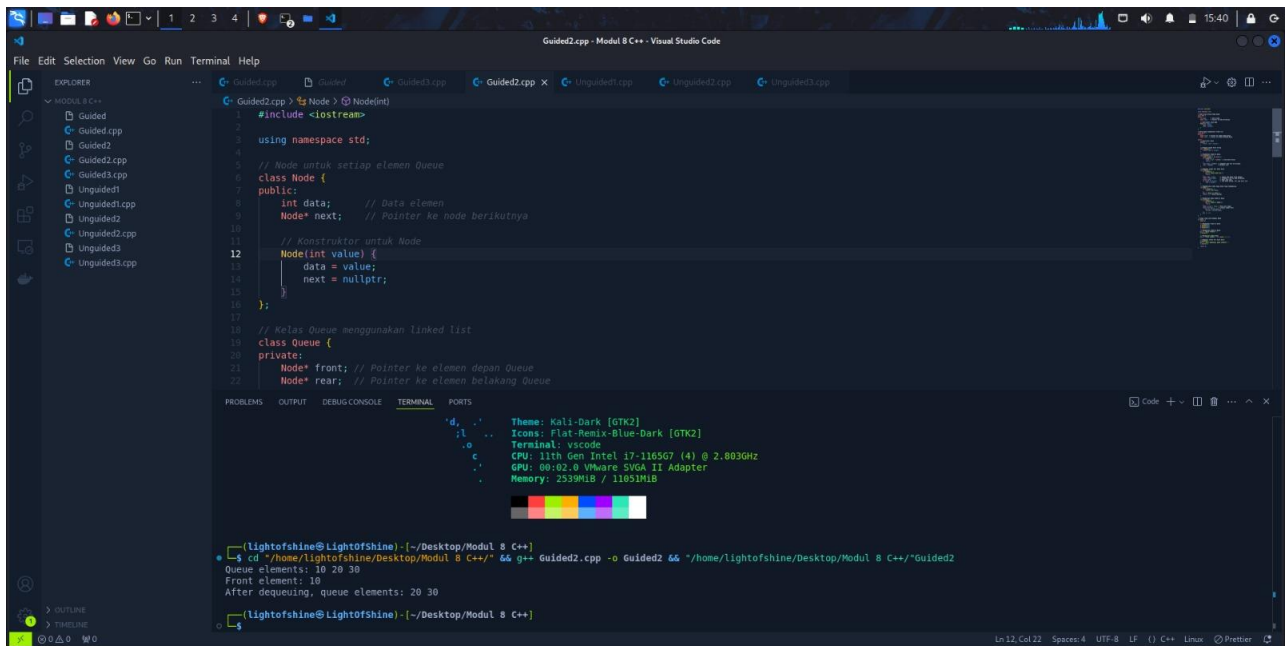
```

## File Guided2.cpp



```
1 // Guided2.cpp
2 #include <iostream>
3 using namespace std;
4 // Node untuk setiap elemen Queue
5 class Node {
6 public:
7     int data; // Data elemen
8     Node* next; // Pointer ke node berikutnya
9
10 // Konstruktor untuk Node
11 Node(int value) {
12     data = value;
13     next = nullptr;
14 }
15 };
16
17 // Kelas Queue menggunakan linked list
18 class Queue {
19 private:
20     Node* front; // Pointer ke elemen depan Queue
21     Node* rear; // Pointer ke elemen belakang Queue
22
23 public:
24     // Konstruktor Queue
25     Queue() {
26         front = rear = nullptr;
27     }
28
29 // Mengecek apakah Queue kosong
30 bool isEmpty() {
31     return front == nullptr;
32 }
33
34 // Menambahkan elemen ke Queue
35 void enqueue(int x) {
36     Node* newNode = new Node(x);
37     if (isEmpty()) {
38         front = rear = newNode; // Jika Queue kosong
39     }
40     return;
41 }
```

## Outputnya



```
Theme: Kali-Dark [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: vscode
CPU: 11th Gen Intel i7-1165G7 (4) @ 2.803GHz
GPU: 60-02-0 VMware SVGA II Adapter
Memory: 2539M1B / 11051M1B

(Lightofshine@Lightofshine) [~/Desktop/Modul 8 C++]
$ cd ~/home/lightofshine/Desktop/Modul 8 C++/ && g++ Guided2.cpp -o Guided2 && ./Guided2
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
$
```

Untuk Source Codenya Lebih Lengkap ada di bawah ini :

### **Guided2.cpp**

```
#include <iostream>

using namespace std;

// Node untuk setiap elemen Queue
class Node {
public:
    int data;    // Data elemen
    Node* next; // Pointer ke node berikutnya

    // Konstruktor untuk Node
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

// Kelas Queue menggunakan linked list
class Queue {
private:
    Node* front; // Pointer ke elemen depan Queue
    Node* rear;  // Pointer ke elemen belakang Queue

public:
    // Konstruktor Queue
    Queue() {
        front = rear = nullptr;
    }

    // Mengecek apakah Queue kosong
```

```

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

// Menambahkan elemen ke Queue
void enqueue(int x) {
    Node* newNode = new Node(x);
    if (isEmpty()) {
        front = rear = newNode; // Jika Queue kosong
        return;
    }
    rear->next = newNode; // Tambahkan node baru ke
    belakang
    rear = newNode;    // Perbarui rear
}

// Menghapus elemen dari depan Queue
void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow\n";
        return;
    }
    Node* temp = front;    // Simpan node depan untuk
    dihapus
    front = front->next;    // Pindahkan front ke node
    berikutnya
    delete temp;          // Hapus node lama
    if (front == nullptr) // Jika Queue kosong, rear juga harus
    null
        rear = nullptr;
}

// Mengembalikan elemen depan Queue tanpa
menghapusnya
int peek() {

```



```

    if (!isEmpty()) {
        return front->data;
    }

    cout << "Queue is empty\n";

    return -1; // Nilai sentinel
}

// Menampilkan semua elemen di Queue
void display() {
    if (isEmpty()) {
        cout << "Queue is empty\n";

        return;
    }

    Node* current = front; // Mulai dari depan
    while (current) { // Iterasi sampai akhir

        cout << current->data << " ";

        current = current->next;
    }

    cout << "\n";
}

};

// Fungsi utama untuk menguji Queue
int main() {
    Queue q;

    // Menambahkan elemen ke Queue
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    // Menampilkan elemen di Queue
    cout << "Queue elements: ";

```

```
q.display();
```

```
// Menampilkan elemen depan
```

```
cout << "Front element: " << q.peek() << "\n";
```

```
// Menghapus elemen dari depan Queue
```

```
q.dequeue();
```

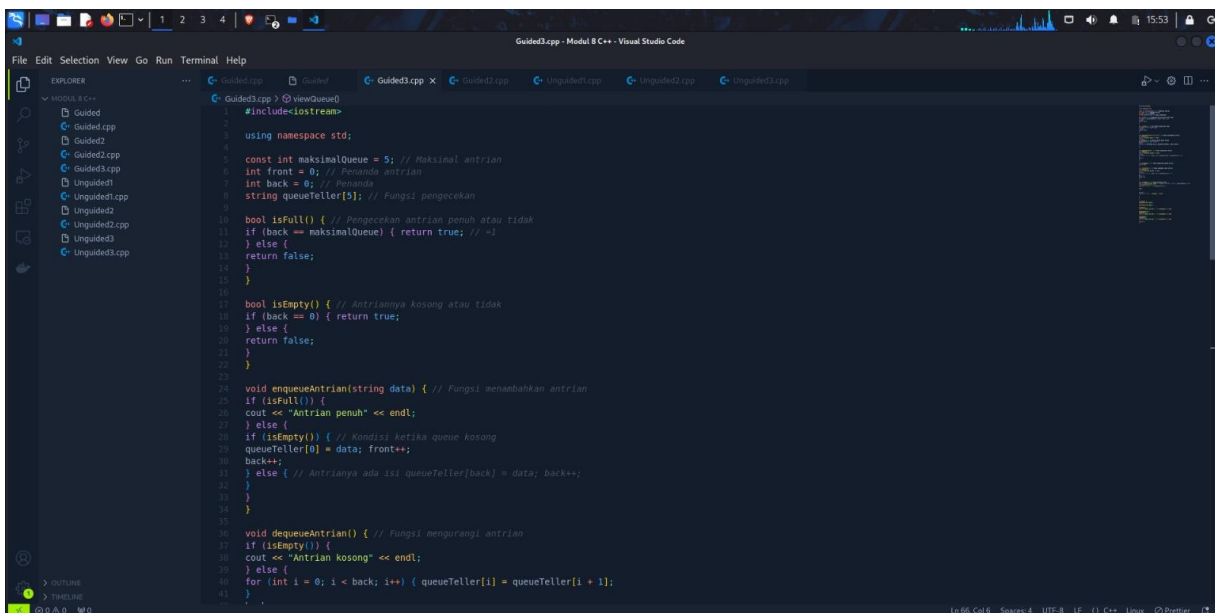
```
cout << "After dequeuing, queue elements: ";
```

```
q.display();
```

```
return 0;
```

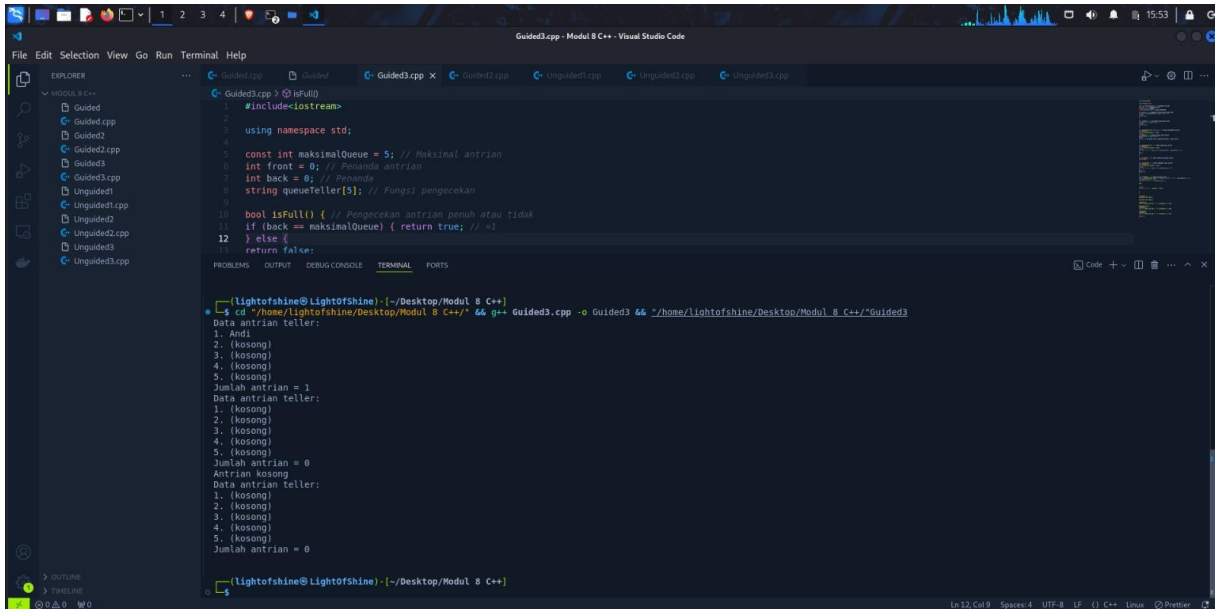
```
}
```

## File Guided3.cpp



```
1  #include<iostream>
2
3  using namespace std;
4
5  const int maksimalQueue = 5; // Maksimal antrian
6  int front = 0; // Penanda antrian
7  int back = 0; // Penanda
8  string queueTeller[5]; // Fungsi pengecekan
9
10 bool isFull() { // Pengecekan antrian penuh atau tidak
11     if (back == maksimalQueue) { return true; // +1
12     } else {
13         return false;
14     }
15 }
16
17 bool isEmpty() { // Antrian kosong atau tidak
18     if (back == 0) { return true;
19     } else {
20         return false;
21     }
22 }
23
24 void enqueueAntrian(string data) { // Fungsi menambahkan antrian
25     if (isFull()) {
26         cout << "Antrian penuh" << endl;
27     } else {
28         if (isEmpty()) { // Kondisi ketika queue kosong
29             queueTeller[0] = data; front++;
30             back++;
31         } else { // Antrian ada isi queueTeller[back] = data; back++;
32             queueTeller[back] = data; back++;
33         }
34     }
35 }
36
37 void dequeueAntrian() { // Fungsi mengurangi antrian
38     if (isEmpty()) {
39         cout << "Antrian kosong" << endl;
40     } else {
41         for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
42         }
43     }
44 }
```

Outputnya :



```
Guided3.cpp - Modul 8 C++ - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Explorer
  Guided
  Guided1.cpp
  Guided2.cpp
  Guided3.cpp
  Guided3.cpp
  Unguided1.cpp
  Unguided2.cpp
  Unguided3.cpp

Guided3.cpp
1 #include<iostream>
2 using namespace std;
3
4 const int maksimalQueue = 5; // Maksimal antrian
5 int front = 0; // Penanda antrian
6 int back = 0; // Penanda
7 string queueTeller[5]; // Fungsi pengecekan
8
9
10 bool isFull() { // Pengecekan antrian penuh atau tidak
11   if (back == maksimalQueue) { return true; // =1
12 } else {
13   return false;
14 }
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

Terminal
(Lightofshine@Lightofshine) [~/Desktop/Modul 8 C++]
$ cd ~/home/lightofshine/Desktop/Modul 8 C++/ && g++ Guided3.cpp -o Guided3 && ./Guided3 &&
Data antrian teller:
1. Andi
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
Antrian kosong
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
(Lightofshine@Lightofshine) [~/Desktop/Modul 8 C++]
$
```

Untuk Source Codenya Lebih Lengkap dibawah ini :

### Guided3.cpp

```
#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;
```

```
}
```

#### IV. Unguided

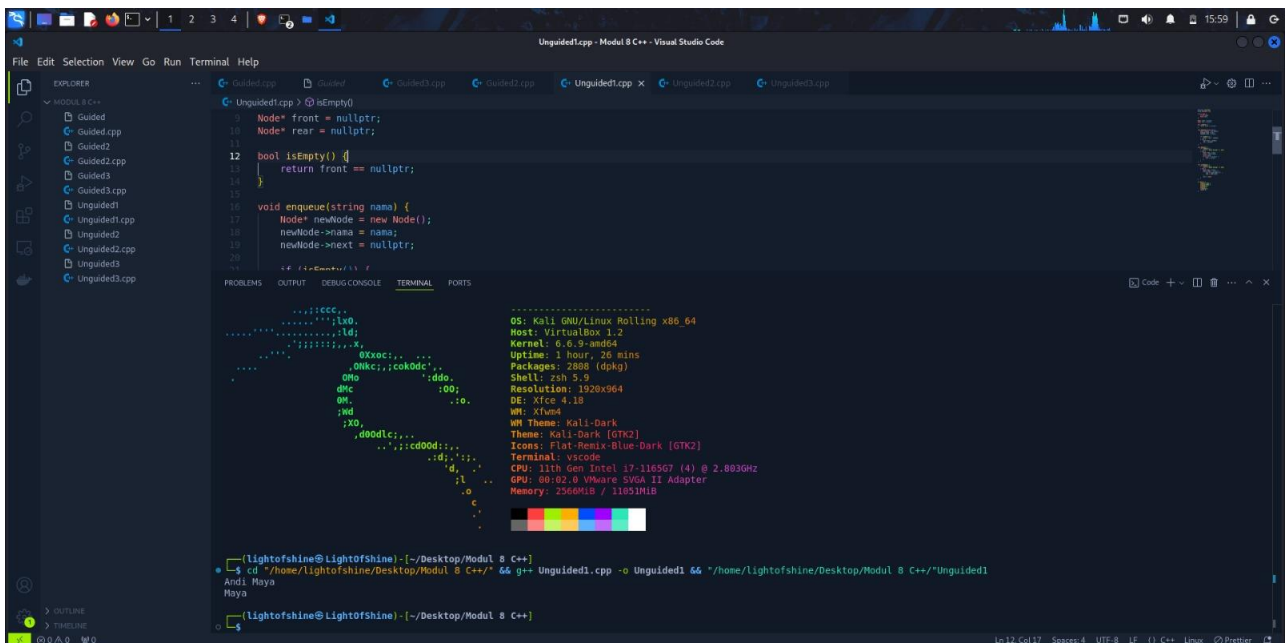
#### D. UNGUIDED

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa
3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

Noted : Untuk data mahasiswa dan nim dimasukan oleh user

**Jawab :**

## 1. File Unguided1.cpp



Untuk Source codenya lebih lengkap dibawah ini :

## Unguided1.cpp

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

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

```
Node* front = nullptr;
Node* rear = nullptr;
```

```

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

void enqueue(string nama) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->next = nullptr;

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

void dequeue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr) {
            rear = nullptr;
        }
    }
}

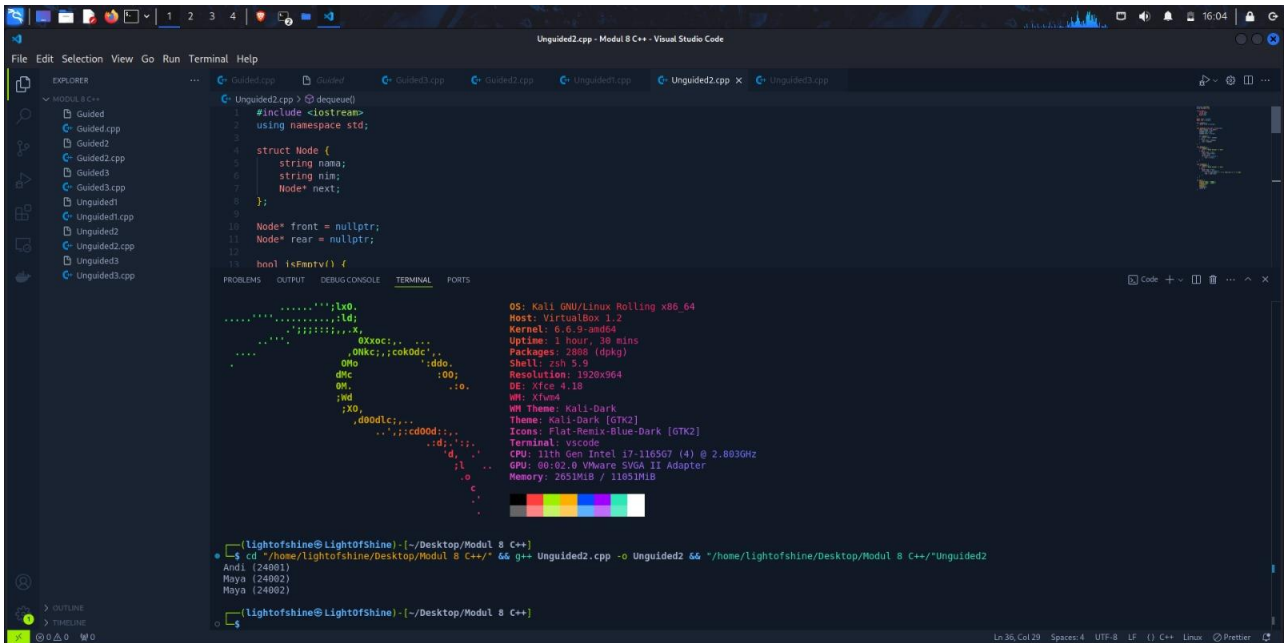
void viewQueue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
        while (temp != nullptr) {
            cout << temp->nama << " ";
            temp = temp->next;
        }
        cout << endl;
    }
}

int main() {
    enqueue("Andi");
    enqueue("Maya");
    viewQueue();
    dequeue();
    viewQueue();
    return 0;
}

```



## 2. File Unguied2.cpp



Untuk Source codenya lebih lengkap dibawah ini :

### Unguied2.cpp

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

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

Node* front = nullptr;
Node* rear = nullptr;

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

void enqueue(string nama, string nim) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

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

void dequeue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
```

```

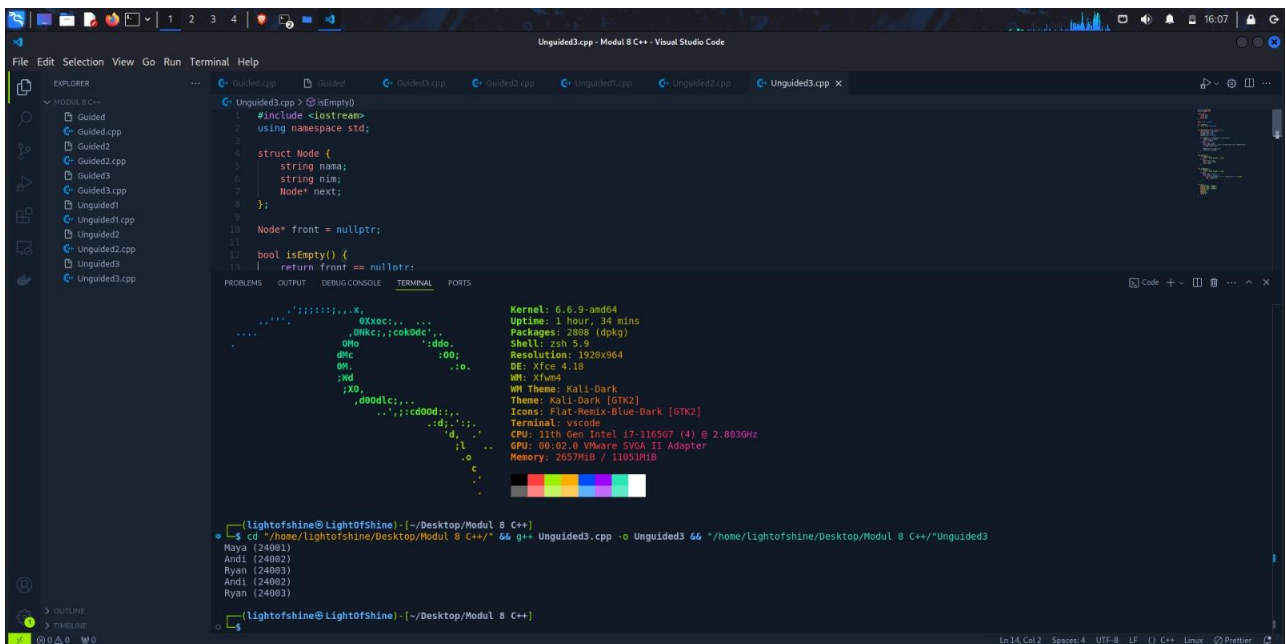
        front = front->next;
        delete temp;
        if (front == nullptr) {
            rear = nullptr;
        }
    }
}

void viewQueue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
        while (temp != nullptr) {
            cout << temp->nama << " (" << temp->nim << ")" << endl;
            temp = temp->next;
        }
    }
}

int main() {
    enqueue("Andi", "24001");
    enqueue("Maya", "24002");
    viewQueue();
    dequeue();
    viewQueue();
    return 0;
}

```

### 3. File Unguided3.cpp



Untuk Source codenya lebih lengkap dibawah ini :

#### Unguided3.cpp

```

#include <iostream>
using namespace std;

```

```

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

Node* front = nullptr;

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

void enqueue(string nama, string nim) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

    if (isEmpty() || newNode->nim < front->nim) {
        newNode->next = front;
        front = newNode;
    } else {
        Node* temp = front;
        while (temp->next != nullptr && temp->next->nim < newNode->nim) {
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
    }
}

void viewQueue() {
    if (isEmpty()) {
        cout << "Queue kosong!" << endl;
    } else {
        Node* temp = front;
        while (temp != nullptr) {
            cout << temp->nama << " (" << temp->nim << ")" << endl;
            temp = temp->next;
        }
    }
}

int main() {
    enqueue("Andi", "24002");
    enqueue("Maya", "24001");
    enqueue("Budi", "24003");
    viewQueue();
    dequeue();
    viewQueue();
    return 0;
}

```

## **V. Kesimpulan**

Queue adalah struktur data yang menggunakan prinsip FIFO (First-In, First-Out), di mana data yang masuk lebih dulu akan dikeluarkan lebih dulu. Konsep ini mirip dengan antrian dalam kehidupan nyata, seperti antrean di kasir, di mana yang datang lebih dulu dilayani lebih dulu. Queue memiliki dua ujung utama, yaitu front untuk menghapus data dan rear untuk menambahkan data.

Perbedaan utama antara queue dan stack terletak pada aturan penanganan datanya. Stack menggunakan prinsip LIFO (Last-In, First-Out), sedangkan queue menggunakan FIFO. Queue mendukung berbagai operasi seperti enqueue (menambah data), dequeue (menghapus data), peek (melihat data depan), isEmpty (mengecek apakah kosong), isFull (mengecek apakah penuh), dan size (menghitung jumlah elemen).

Queue dapat diimplementasikan menggunakan array untuk kapasitas tetap atau linked list untuk fleksibilitas tanpa batasan ukuran. Implementasi queue sering digunakan dalam kehidupan sehari-hari dan aplikasi teknologi, seperti antrean layanan, pengelolaan tugas, atau pemrosesan data pada sistem operasi.

Struktur data ini menjadi solusi penting untuk menyusun dan mengatur data yang perlu diproses secara teratur dan efisien.













