

LAPORAN PRAKTIKUM
Modul 8
“QUEUE”



Disusun Oleh:

Ahmad Al - Farizi - 2311104054

Kelas :

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. 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

2. Landasan Teori

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out), di mana data yang pertama dimasukkan akan menjadi data yang pertama dikeluarkan. Konsep ini mirip dengan antrian di kehidupan sehari – hari, di mana konsumen yang datang lebih dulu akan dilayani lebih dulu. Queue dapat diimplementasikan menggunakan array atau linked list dan terdiri dari dua pointer utama: front dan rear. Pointer front mengacu pada elemen pertama dalam queue, sedangkan pointer rear mengacu pada elemen terakhir.

Perbedaan utama antara stack dan queue terletak pada aturan penambahan dan penghapusan elemen. Pada stack, penambahan dan penghapusan elemen dilakukan di satu ujung yang disebut top. Elemen terakhir yang dimasukkan akan berada di posisi paling atas dan menjadi elemen pertama yang dihapus, mengikuti prinsip LIFO (Last In, First Out). Analogi sederhana stack adalah tumpukan piring, di mana piring terakhir yang ditambahkan berada di posisi paling atas dan akan diambil terlebih dahulu.

Sebaliknya, pada queue, penambahan dan penghapusan elemen dilakukan di dua ujung yang berbeda. Elemen baru ditambahkan di ujung belakang (rear atau tail) dan dihapus dari ujung depan (front atau head), mengikuti prinsip FIFO (First In, First Out). Operasi penambahan elemen disebut Enqueue, sementara operasi penghapusan elemen disebut Dequeue. Dalam proses Enqueue, elemen ditambahkan di belakang queue setelah elemen terakhir yang ada. Pada proses Dequeue, elemen paling depan (head) dihapus, dan posisi head akan bergeser ke elemen berikutnya. Contoh nyata penggunaan queue adalah antrian di kasir, di mana orang pertama yang datang adalah yang pertama dilayani.

Operasi-operasi dasar 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.

3. Guided

1. Guided 8.1

Kode program di bawah mengimplementasikan struktur data queue menggunakan array di C++. Kelas Queue memiliki atribut front dan rear untuk melacak elemen pertama dan terakhir, serta array arr dengan kapasitas maksimum yang ditentukan oleh MAX. Konstruktor Queue menginisialisasi front dan rear ke -1, menandakan queue kosong. Fungsi isFull() memeriksa apakah queue penuh, dan isEmpty() memeriksa apakah queue kosong. Fungsi enqueue(int x) menambahkan elemen ke belakang queue jika tidak penuh, sedangkan dequeue() menghapus elemen dari depan queue jika tidak kosong. Fungsi peek() mengembalikan elemen terdepan tanpa menghapusnya, dan display() menampilkan semua elemen dalam queue. Dalam fungsi main(), dilakukan beberapa operasi queue: enqueue() untuk menambahkan elemen, display() untuk menampilkan elemen, dan dequeue() untuk menghapus elemen, menunjukkan penggunaan dasar dari queue.

Kode Program:

```
#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();

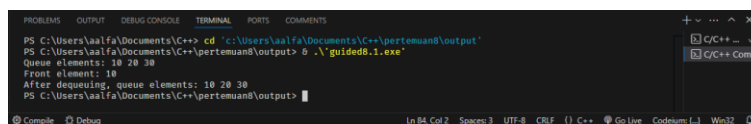
    cout << "Front element: " << q.peek() << "\n";

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

    return 0;
}

```

Output dari Kode Program:



```

PS C:\Users\aaifa\Documents\C++> cd 'c:\Users\aaifa\Documents\C++\pertemuan8\output'
PS C:\Users\aaifa\Documents\C++\pertemuan8\output> .\guided8.1.exe
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS C:\Users\aaifa\Documents\C++\pertemuan8\output>

```

2. Guided 8.2

Kode program di bawah mengimplementasikan struktur data queue menggunakan linked list di C++. Kelas Node menyimpan data dan pointer ke node berikutnya, sedangkan kelas Queue menggunakan dua pointer, front dan rear, untuk melacak elemen pertama dan terakhir. Konstruktor Queue menginisialisasi front dan rear ke nullptr. Fungsi isEmpty() mengecek apakah queue kosong. Fungsi enqueue(int x) menambahkan node baru ke belakang queue. Jika queue kosong, node baru menjadi elemen pertama. Jika tidak, node baru ditambahkan setelah rear, dan rear diperbarui. Fungsi dequeue() menghapus elemen dari depan queue. Jika queue kosong, mencetak pesan

"Queue Underflow". Jika tidak, node pertama dihapus dan front diperbarui ke node berikutnya. Jika queue kosong setelah penghapusan, rear diatur ke nullptr. Fungsi peek() mengembalikan nilai elemen terdepan tanpa menghapusnya, dan mencetak pesan jika queue kosong. Fungsi display() menampilkan semua elemen dari front ke rear. Di fungsi main(), dilakukan operasi enqueue() untuk menambah elemen, display() untuk menampilkan elemen, dequeue() untuk menghapus elemen, dan peek() untuk melihat elemen terdepan, menunjukkan penggunaan dasar queue.

Kode Program

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

class Node {
public:
    int data;
    Node* next;

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

class Queue {
private:
    Node* front;
    Node* rear;
public:

    Queue() {
        front = rear = nullptr;
    }
};
```

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

void enqueue(int x) {
    Node* newNode = new Node(x);
    if (isEmpty()) {
        front = rear = newNode;
        return;
    }
    rear->next = newNode;
    rear = newNode;
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow\n";
        return;
    }
    Node* temp = front;
    front = front->next;
    delete temp;
    if (front == nullptr)
        rear = nullptr;
}

int peek() {
    if (!isEmpty()) {
        return front->data;
    }
    cout << "Queue is empty\n";
}
```

```
        return -1;
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is empty\n";
            return;
        }
        Node* current = front;
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << "\n";
    }
};

int main() {
    Queue q;

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

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

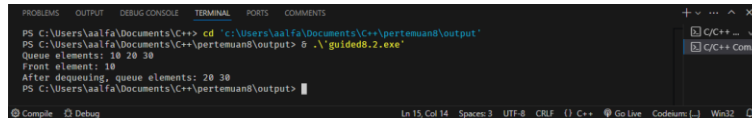
    cout << "Front element: " << q.peek() << "\n";

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



```
return 0;  
}
```

Output dari Kode Program



3. Guided 8.3

Kode program di bawah ini mengimplementasikan struktur data queue menggunakan array dengan kapasitas maksimum 5. Dua variabel, front dan back, melacak posisi depan dan belakang queue, sedangkan array queueTeller menyimpan elemen queue. Fungsi isFull() memeriksa apakah queue penuh, dan isEmpty() memeriksa apakah queue kosong. Fungsi enqueueAntrian(string data) menambahkan elemen ke belakang queue jika tidak penuh. Fungsi dequeueAntrian() menghapus elemen dari depan queue dengan menggeser elemen ke kiri. Fungsi countQueue() mengembalikan jumlah elemen dalam queue, dan clearQueue() mengosongkan queue. Fungsi viewQueue() menampilkan elemen-elemen queue. Dalam fungsi main(), program menambahkan elemen ke queue, menampilkan elemen, menghapus elemen, dan mengosongkan queue, menunjukkan operasi dasar queue.

Kode Program :

```
#include<iostream>  
using namespace std;  
  
const int maksimalQueue = 5;  
int front = 0;  
int back = 0;  
string queueTeller[5];  
  
bool isFull() {  
    if (back == maksimalQueue) {
```

```
        return true;
    } else {
        return false;
    }
}

bool isEmpty() {
    if (back == 0) {
        return true;
    } else {
        return false;
    }
}

void enqueueAntrian(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        if (isEmpty()) {
            queueTeller[0] = data; front++;
            back++;
        } else {
        }
    }
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
        }
    }
}
```

```
        back--;  
    }  
}  
  
int countQueue() {  
    return back;  
}  
  
void clearQueue() {  
    if (isEmpty()) {  
        cout << "Antrian kosong" << endl;  
    } else {  
        for (int i = 0; i < back; i++) { queueTeller[i] = "";  
        }  
        back = 0;  
        front = 0;  
    }  
}  
  
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;

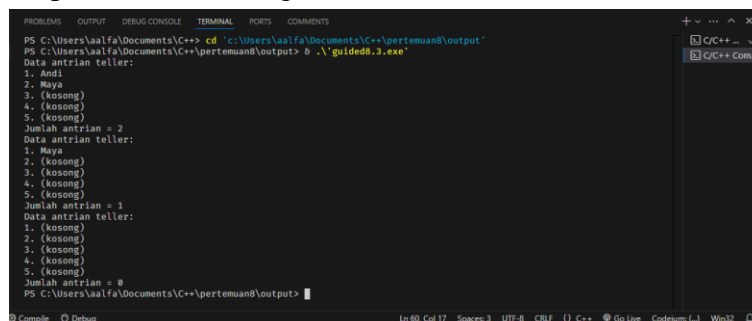
dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

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

return 0;
}

```

Output dari Kode Program:



```

PS C:\Users\aa\Documents\C++> cd 'c:\Users\aa\Documents\C++\pertemuan8\output'
PS C:\Users\aa\Documents\C++\pertemuan8\output> g++ guided8.3.exe
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\aa\Documents\C++\pertemuan8\output>

```

4. Unguided

1. Soal No 1

Kode Kode program di atas mengimplementasikan struktur data queue menggunakan array dengan kapasitas maksimum 5. Dua variabel, front dan back, melacak posisi depan dan belakang queue, sedangkan array queueTeller menyimpan elemen queue. Fungsi isFull() memeriksa apakah queue penuh, dan isEmpty() memeriksa apakah queue kosong. Fungsi enqueueAntrian(string data) menambahkan elemen ke belakang queue jika tidak penuh. Fungsi

dequeueAntrian() menghapus elemen dari depan queue dengan menggeser elemen ke kiri. Fungsi countQueue() mengembalikan jumlah elemen dalam queue, dan clearQueue() mengosongkan queue. Fungsi viewQueue() menampilkan elemen-elemen queue. Dalam fungsi main(), program menambahkan elemen ke queue, menampilkan elemen, menghapus elemen, dan mengosongkan queue, menunjukkan operasi dasar queue.

Kode Program:

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

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

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

bool isFull() {
    return false;
}

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

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

    if (isEmpty()) {
```

```
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

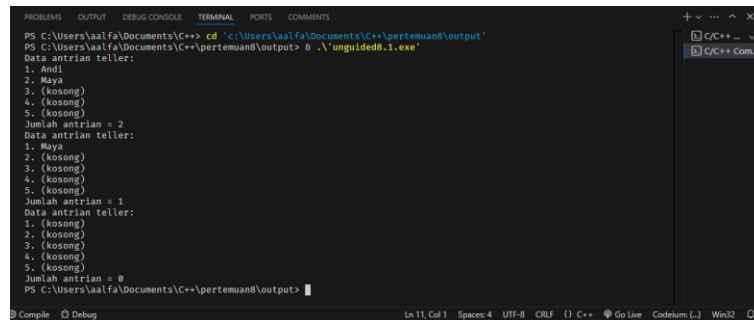
void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr) {
            back = nullptr;
        }
    }
}

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    for (int i = 0; i < 5; i++) {
        if (temp != nullptr) {
            cout << index << ". " << temp->data << endl;
            temp = temp->next;
        } else {
            cout << index << ". (kosong)" << endl;
        }
        index++;
    }
}
```

```
    }  
}  
  
int countQueue() {  
    int count = 0;  
    Node* temp = front;  
    while (temp != nullptr) {  
        count++;  
        temp = temp->next;  
    }  
    return count;  
}  
  
void clearQueue() {  
    while (!isEmpty()) {  
        dequeue();  
    }  
}  
  
int main() {  
    enqueue("Andi");  
    enqueue("Maya");  
  
    viewQueue();  
    cout << "Jumlah antrian = " << countQueue() << endl;  
  
    dequeue();  
    viewQueue();  
    cout << "Jumlah antrian = " << countQueue() << endl;  
  
    clearQueue();  
    viewQueue();  
}
```

```
cout << "Jumlah antrian = " << countQueue() << endl;  
  
return 0;  
  
}
```

Output dari Kode Program:



The screenshot shows a C++ IDE with a terminal window displaying the output of a program. The output shows a queue of names: 1. Andi, 2. Maya, 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 2. The output then shows the queue after removing the first element: 1. Maya, 2. (kosong), 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 1. The output then shows the queue after removing the first element again: 1. (kosong), 2. (kosong), 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 0. The output then shows the queue after adding a new element: 1. Andi, 2. Maya, 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 2. The output then shows the queue after removing the first element: 1. Maya, 2. (kosong), 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 1. The output then shows the queue after removing the first element again: 1. (kosong), 2. (kosong), 3. (kosong), 4. (kosong), 5. (kosong). The total number of elements in the queue is 0.

2. Soal No 2

Program ini mengimplementasikan antrian (queue) menggunakan linked list dalam C++. Struktur Mahasiswa menyimpan atribut nama dan nim. Pointer front dan back menunjukkan elemen pertama dan terakhir antrian. Fungsi enqueue() menambahkan mahasiswa ke belakang antrian, sedangkan dequeue() menghapus mahasiswa dari depan antrian. Fungsi isEmpty() memeriksa apakah antrian kosong. Fungsi viewQueue() menampilkan hingga 5 elemen antrian, dengan elemen kosong ditampilkan sebagai "(kosong)". Fungsi countQueue() menghitung jumlah elemen antrian, dan clearQueue() menghapus semua elemen. Dalam fungsi main(), program menambah mahasiswa ke antrian, menampilkan antrian, menghitung elemen, menghapus elemen, dan mengosongkan antrian, serta menampilkan hasil setiap operasi.

Kode Program:

```
#include <iostream>  
  
using namespace std;  
  
struct Mahasiswa {  
    string nama;  
    int nim;
```



```
Mahasiswa* next;

};

Mahasiswa* front = nullptr;
Mahasiswa* back = nullptr;

bool isFull() {
    return false;
}

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

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

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

void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
```

```
Mahasiswa* temp = front;
front = front->next;
delete temp;
if (front == nullptr) {
    back = nullptr;
}
}
}

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Mahasiswa* temp = front;
    int index = 1;
    for (int i = 0; i < 5; i++) {
        if (temp != nullptr) {
            cout << index << ". " << temp->nama << " (" << temp->nim << ")"
            << endl;
            temp = temp->next;
        } else {
            cout << index << ". (kosong)" << endl;
        }
        index++;
    }
}

int countQueue() {
    int count = 0;
    Mahasiswa* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
}
```

```
        return count;
    }

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

    int main() {
        enqueue("Andi", 12345);
        enqueue("Maya", 67890);

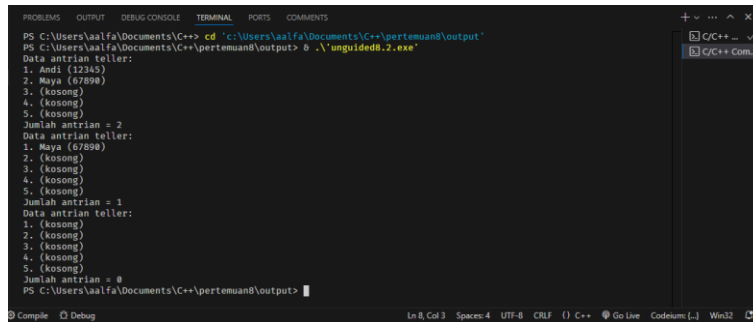
        viewQueue();
        cout << "Jumlah antrian = " << countQueue() << endl;

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

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

        return 0;
    }
```

Output Kode Program:



```

PS C:\Users\aaifa\Documents\C++> cd "c:\Users\aaifa\Documents\C++\pertemuan8\output"
PS C:\Users\aaifa\Documents\C++\pertemuan8\output> b .\unguided1.2.exe
Data antrian teller:
1. Andi (12345)
2. Maya (67890)
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya (67890)
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\aaifa\Documents\C++\pertemuan8\output>
  
```

3. Soal No 3

Program ini membuat antrian prioritas (priority queue) menggunakan linked list dalam C++. Struktur Mahasiswa menyimpan nama, nim, dan pointer next. Pointer front dan back menunjukkan elemen pertama dan terakhir antrian. Fungsi enqueuePrioritized() menambahkan mahasiswa baru berdasarkan urutan NIM, dengan NIM lebih kecil ditempatkan lebih depan. Fungsi dequeue() menghapus elemen terdepan. Fungsi viewQueue() menampilkan hingga 5 elemen dalam antrian atau "(Kosong)" jika tidak ada elemen. Fungsi countQueue() menghitung jumlah elemen, dan clearQueue() menghapus semua elemen. Fungsi main() menambahkan mahasiswa, menampilkan, menghitung, menghapus, dan mengosongkan antrian, lalu menampilkan hasil setiap operasi.

Kode Program:

```

#include <iostream>

using namespace std;

struct Mahasiswa {
    string nama;
    int nim;
    Mahasiswa* next;
};

Mahasiswa* front = nullptr;
Mahasiswa* back = nullptr;
  
```

```
bool isFull() {
    return false;
}

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

void enqueuePrioritized(string nama, int nim) {
    Mahasiswa* newNode = new Mahasiswa();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

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

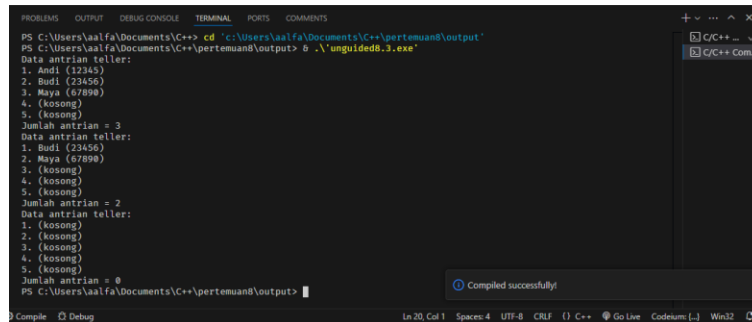
```
void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Mahasiswa* temp = front;
        front = front->next;
        delete temp;
        if (front == nullptr) {
            back = nullptr;
        }
    }
}

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Mahasiswa* temp = front;
    int index = 1;
    for (int i = 0; i < 5; i++) {
        if (temp != nullptr) {
            cout << index << ". " << temp->nama << " (" << temp->nim << ")"
            << endl;
            temp = temp->next;
        } else {
            cout << index << ". (kosong)" << endl;
        }
        index++;
    }
}

int countQueue() {
    int count = 0;
    Mahasiswa* temp = front;
```

```
while (temp != nullptr) {  
    count++;  
    temp = temp->next;  
}  
return count;  
}  
  
void clearQueue() {  
    while (!isEmpty()) {  
        dequeue();  
    }  
}  
  
int main() {  
    enqueuePrioritized("Andi", 12345);  
    enqueuePrioritized("Maya", 67890);  
    enqueuePrioritized("Budi", 23456);  
  
    viewQueue();  
    cout << "Jumlah antrian = " << countQueue() << endl;  
  
    dequeue();  
    viewQueue();  
    cout << "Jumlah antrian = " << countQueue() << endl;  
  
    clearQueue();  
    viewQueue();  
    cout << "Jumlah antrian = " << countQueue() << endl;  
  
    return 0;  
}
```

Output dari Kode Program:



```
PS C:\Users\aaifa\Documents\C++> cd 'c:\Users\aaifa\Documents\C++\pertemuan8\output'
PS C:\Users\aaifa\Documents\C++\pertemuan8\output> g++ 'unguided8.3.exe'
Data antrian teller:
1. Andi (32245)
2. Budi (23456)
3. Maya (67890)
4. (kosong)
5. (kosong)
Jumlah antrian = 3
Data antrian teller:
1. Budi (23456)
2. Maya (67890)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS C:\Users\aaifa\Documents\C++\pertemuan8\output>
```

5. Kesimpulan

Queue adalah struktur data yang mengikuti prinsip FIFO (First-In First-Out), di mana elemen yang pertama kali masuk akan menjadi elemen yang pertama kali keluar. Queue diimplementasikan menggunakan array atau linked list dan terdiri dari dua pointer utama: front dan rear. Operasi dasar pada queue meliputi enqueue() untuk menambahkan elemen di belakang antrian, dequeue() untuk menghapus elemen dari depan antrian, peek() untuk melihat elemen depan tanpa menghapusnya, isEmpty() untuk mengecek apakah antrian kosong, isFull() untuk mengecek apakah antrian penuh, dan size() untuk menghitung jumlah elemen dalam antrian. Queue banyak digunakan dalam berbagai aplikasi nyata seperti antrian di kasir, karena sifatnya yang mirip dengan antrian dalam kehidupan sehari-hari.

