LAPORAN PRAKTIKUM
Modul 8
QUEUE

Disusun Oleh:
Jauhar Fajar Zuhair
2311104072
S1SE-07-2

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan Praktikum

1. Mahasiswa diharapkan dapat memahami dan menjelaskan konsep dasar dari struktur data queue
2. Mahasiswa dapat mengimplementasikan operasi-operasi dasar queue (penambahan dan penghapusan data)
3. Mahasiswa mampu menerapkan operasi untuk menampilkan data dalam queue

Landasan Teori

Queue (antrian) adalah sebuah struktur data yang menggunakan prinsip FIFO (First-In First-Out), yang berarti data yang pertama kali dimasukkan akan menjadi data yang pertama kali dikeluarkan. Konsep ini mirip dengan antrian dalam kehidupan sehari-hari, misalnya antrian di bank dimana nasabah yang datang lebih awal akan dilayani terlebih dahulu.

Dalam implementasinya, queue dapat dibuat menggunakan array atau linked list. Struktur queue memiliki dua pointer penting:

- Front/head: menunjuk ke elemen pertama dalam queue
- Rear/tail/back: menunjuk ke elemen terakhir dalam queue

Perbedaan Stack dan Queue

1. **Stack (LIFO - Last In First Out)**
    - Operasi penambahan dan penghapusan dilakukan pada satu ujung (top)
    - Mirip seperti tumpukan piring, dimana piring yang terakhir ditumpuk akan menjadi yang pertama diambil
2. **Queue (FIFO - First In First Out)**
    - Operasi dilakukan pada dua ujung berbeda
    - Penambahan elemen (Enqueue) dilakukan di ujung belakang (rear/tail)
    - Penghapusan elemen (Dequeue) dilakukan di ujung depan (front/head)

Operasi Dasar Queue

1. enqueue(): Menambahkan data baru ke dalam queue
2. dequeue(): Mengeluarkan data dari queue
3. peek(): Melihat data di queue tanpa menghapusnya
4. isEmpty(): Memeriksa apakah queue kosong
5. isFull(): Memeriksa apakah queue sudah penuh

6. size(): Menghitung jumlah elemen dalam queue

Guided

Guided1.cpp

```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;
}
```

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;
}
```

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;
}
```

Unguided

unGuided1.cpp

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

// Struktur Node untuk Linked List
struct Node {
    string nama;
    string nim;
    Node* next;
};

// Kelas Queue
class Queue {
private:
    Node *front, *rear;
    int count;

public:
    // Constructor
    Queue() {
        front = NULL;
        rear = NULL;
        count = 0;
    }

    // Cek apakah queue kosong
    bool isEmpty() {
        return front == NULL;
    }

    // Menambah data ke queue (enqueue)
    void enqueue(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
```

```
        newNode->next = NULL;

        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
        count++;
        cout << "Data berhasil ditambahkan ke antrian!" << endl;
    }

    // Menghapus data dari queue (dequeue)
    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }

        Node* temp = front;
        front = front->next;
        delete temp;
        count--;

        if (front == NULL) {
            rear = NULL;
        }
        cout << "Data terdepan berhasil dihapus dari antrian!" << endl;
    }

    // Menampilkan semua data dalam queue
    void display() {
        if (isEmpty()) {
            cout << "Antrian kosong!" << endl;
            return;
        }

        cout << "\nData dalam antrian:" << endl;
        cout << "--------------------------------" << endl;
        cout << "No.\tNIM\t\tNama" << endl;
        cout << "--------------------------------" << endl;

        Node* current = front;
        int i = 1;
        while (current != NULL) {
            cout << i << "\t" << current->nim << "\t" << current->nama <<
endl;
            current = current->next;
            i++;
        }
        cout << "--------------------------------" << endl;
    }

    // Menghitung jumlah data dalam queue
    int size() {
        return count;
    }
```

```
    // Membersihkan queue
    void clear() {
        while (!isEmpty()) {
            dequeue();
        }
        cout << "Antrian telah dibersihkan!" << endl;
    }
};

int main() {
    Queue q;
    int pilihan;
    string nama, nim;

    do {
        cout << "\n=== MENU PROGRAM QUEUE MAHASISWA ===" << endl;
        cout << "1. Tambah Data (Enqueue)" << endl;
        cout << "2. Hapus Data Terdepan (Dequeue)" << endl;
        cout << "3. Tampilkan Data" << endl;
        cout << "4. Jumlah Data" << endl;
        cout << "5. Hapus Semua Data" << endl;
        cout << "0. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1:
                cout << "Masukkan Nama: ";
                getline(cin, nama);
                cout << "Masukkan NIM: ";
                getline(cin, nim);
                q.enqueue(nama, nim);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.display();
                break;
            case 4:
                cout << "Jumlah data dalam antrian: " << q.size() << endl;
                break;
            case 5:
                q.clear();
                break;
            case 0:
                cout << "Program selesai!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    } while (pilihan != 0);

    return 0;
}
```

unGuided2.cpp

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

// Struktur untuk data mahasiswa
struct Mahasiswa {
    string nama;
    string nim;
};

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

class AntrianMahasiswa {
private:
    Node* front;
    Node* rear;
    int jumlahMahasiswa;

public:
    // Constructor
    AntrianMahasiswa() {
        front = NULL;
        rear = NULL;
        jumlahMahasiswa = 0;
    }

    // Mengecek apakah antrian kosong
    bool isEmpty() {
        return front == NULL;
    }

    // Menambah mahasiswa ke antrian
    void tambahMahasiswa() {
        Mahasiswa mhs;
        cout << "\nMasukkan Data Mahasiswa" << endl;
        cout << "----------------------" << endl;
        cout << "Nama Mahasiswa : ";
        getline(cin, mhs.nama);
        cout << "NIM Mahasiswa  : ";
        getline(cin, mhs.nim);

        Node* newNode = new Node();
        newNode->data = mhs;
        newNode->next = NULL;

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

```
        jumlahMahasiswa++;
        cout << "\nMahasiswa berhasil ditambahkan ke antrian!" << endl;
    }

    // Menghapus mahasiswa dari antrian
    void panggilMahasiswa() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }

        Node* temp = front;
        cout << "\nMemanggil mahasiswa dengan data:" << endl;
        cout << "Nama: " << temp->data.nama << endl;
        cout << "NIM : " << temp->data.nim << endl;

        front = front->next;
        delete temp;
        jumlahMahasiswa--;

        if (front == NULL) {
            rear = NULL;
        }
    }

    // Menampilkan daftar antrian mahasiswa
    void tampilkanAntrian() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }

        cout << "\nDaftar Antrian Mahasiswa" << endl;
        cout << "------------------------" << endl;
        cout << "No.\tNIM\t\tNama" << endl;
        cout << "------------------------" << endl;

        Node* current = front;
        int nomor = 1;
        while (current != NULL) {
            cout << nomor << "\t"
                 << current->data.nim << "\t"
                 << current->data.nama << endl;
            current = current->next;
            nomor++;
        }
        cout << "------------------------" << endl;
        cout << "Total mahasiswa dalam antrian: " << jumlahMahasiswa << endl;
    }

    // Melihat data mahasiswa di depan antrian
    void lihatDepan() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }
```

```
        cout << "\nMahasiswa di depan antrian:" << endl;
        cout << "Nama: " << front->data.nama << endl;
        cout << "NIM : " << front->data.nim << endl;
    }
};

int main() {
    AntrianMahasiswa antrian;
    int pilihan;

    do {
        cout << "\n=== PROGRAM ANTRIAN MAHASISWA ===" << endl;
        cout << "1. Tambah Mahasiswa ke Antrian" << endl;
        cout << "2. Panggil Mahasiswa (Hapus dari Antrian)" << endl;
        cout << "3. Tampilkan Semua Antrian" << endl;
        cout << "4. Lihat Mahasiswa di Depan Antrian" << endl;
        cout << "0. Keluar Program" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1:
                antrian.tambahMahasiswa();
                break;
            case 2:
                antrian.panggilMahasiswa();
                break;
            case 3:
                antrian.tampilkanAntrian();
                break;
            case 4:
                antrian.lihatDepan();
                break;
            case 0:
                cout << "Program selesai!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    } while (pilihan != 0);

    return 0;
}
```

unGuided3.cpp

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

// Struktur untuk data mahasiswa
struct Mahasiswa {
    string nama;
    string nim;
};

// Struktur node untuk linked list
```

```
struct Node {
    Mahasiswa data;
    Node* next;
};

class PrioritasAntrianMahasiswa {
private:
    Node* front;
    Node* rear;
    int jumlahMahasiswa;

public:
    PrioritasAntrianMahasiswa() {
        front = NULL;
        rear = NULL;
        jumlahMahasiswa = 0;
    }

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

    // Menambah mahasiswa dengan prioritas berdasarkan NIM
    void tambahMahasiswaPrioritas() {
        Mahasiswa mhs;
        cout << "\nMasukkan Data Mahasiswa" << endl;
        cout << "-----------------------" << endl;
        cout << "Nama Mahasiswa : ";
        getline(cin, mhs.nama);
        cout << "NIM Mahasiswa  : ";
        getline(cin, mhs.nim);

        Node* newNode = new Node();
        newNode->data = mhs;
        newNode->next = NULL;

        // Jika antrian kosong
        if (isEmpty()) {
            front = rear = newNode;
        }
        // Jika NIM baru lebih kecil dari NIM di depan (prioritas tertinggi)
        else if (mhs.nim < front->data.nim) {
            newNode->next = front;
            front = newNode;
        }
        // Mencari posisi yang tepat berdasarkan NIM
        else {
            Node* current = front;
            Node* previous = NULL;

            while (current != NULL && mhs.nim >= current->data.nim) {
                previous = current;
                current = current->next;
            }

            if (current == NULL) { // Tambah di akhir
                rear->next = newNode;
```

```
                rear = newNode;
            } else { // Tambah di tengah
                previous->next = newNode;
                newNode->next = current;
            }
        }

        jumlahMahasiswa++;
        cout << "\nMahasiswa berhasil ditambahkan ke antrian!" << endl;
    }

    // Menghapus mahasiswa dari depan antrian
    void panggilMahasiswa() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }

        Node* temp = front;
        cout << "\nMemanggil mahasiswa dengan data:" << endl;
        cout << "Nama: " << temp->data.nama << endl;
        cout << "NIM : " << temp->data.nim << endl;

        front = front->next;
        delete temp;
        jumlahMahasiswa--;

        if (front == NULL) {
            rear = NULL;
        }
    }

    // Menampilkan daftar antrian mahasiswa (sudah terurut berdasarkan NIM)
    void tampilkanAntrian() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }

        cout << "\nDaftar Antrian Mahasiswa (Diurutkan berdasarkan NIM)" <<
endl;
        cout << "-------------------------------------------------" << endl;
        cout << "No.\tNIM\t\tNama" << endl;
        cout << "-------------------------------------------------" << endl;

        Node* current = front;
        int nomor = 1;
        while (current != NULL) {
            cout << nomor << "\t"
                 << current->data.nim << "\t"
                 << current->data.nama << endl;
            current = current->next;
            nomor++;
        }
        cout << "-------------------------------------------------" << endl;
        cout << "Total mahasiswa dalam antrian: " << jumlahMahasiswa << endl;
    }
```

```
    // Melihat data mahasiswa prioritas tertinggi (NIM terkecil)
    void lihatPrioritas() {
        if (isEmpty()) {
            cout << "\nAntrian kosong!" << endl;
            return;
        }

        cout << "\nMahasiswa dengan prioritas tertinggi:" << endl;
        cout << "Nama: " << front->data.nama << endl;
        cout << "NIM : " << front->data.nim << endl;
    }
};

int main() {
    PrioritasAntrianMahasiswa antrian;
    int pilihan;

    do {
        cout << "\n=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===" << endl;
        cout << "1. Tambah Mahasiswa ke Antrian" << endl;
        cout << "2. Panggil Mahasiswa (Hapus dari Antrian)" << endl;
        cout << "3. Tampilkan Semua Antrian" << endl;
        cout << "4. Lihat Mahasiswa Prioritas Tertinggi" << endl;
        cout << "0. Keluar Program" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1:
                antrian.tambahMahasiswaPrioritas();
                break;
            case 2:
                antrian.panggilMahasiswa();
                break;
            case 3:
                antrian.tampilkanAntrian();
                break;
            case 4:
                antrian.lihatPrioritas();
                break;
            case 0:
                cout << "Program selesai!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    } while (pilihan != 0);

    return 0;
}
```

Output

```
"C:\Users\Administrator\Documents\dok kuliah\semester 3\Stuktur Data\10_Pengenalan_CPP_Bagian_10\2311104072_jauharfz\unGuided\cmake-build-debug\unGuided.exe"

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:1

Masukkan Data Mahasiswa
----------------------
Nama Mahasiswa :Arie
 NIM Mahasiswa  :2311104072

Mahasiswa berhasil ditambahkan ke antrian!

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:1

Masukkan Data Mahasiswa
----------------------
Nama Mahasiswa :Budi
 NIM Mahasiswa  :2311104071

Mahasiswa berhasil ditambahkan ke antrian!

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:1

Masukkan Data Mahasiswa
----------------------
Nama Mahasiswa :Setiadi
 NIM Mahasiswa  :2311104073

Mahasiswa berhasil ditambahkan ke antrian!

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:1

Masukkan Data Mahasiswa
----------------------
Nama Mahasiswa :Bermain Judol
 NIM Mahasiswa  :2311104070

Mahasiswa berhasil ditambahkan ke antrian!

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:2

Memanggil mahasiswa dengan data:
Nama: Bermain Judol
NIM : 2311104070

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:3

Daftar Antrian Mahasiswa (Diurutkan berdasarkan NIM)
-------------------------------------------------
No.     NIM             Nama
-------------------------------------------------
1       2311104071      Budi
2       2311104072      Arie
3       2311104073      Setiadi
-------------------------------------------------
Total mahasiswa dalam antrian: 3

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:4

Mahasiswa dengan prioritas tertinggi:
Nama: Budi
NIM : 2311104071

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa ke Antrian
2. Panggil Mahasiswa (Hapus dari Antrian)
3. Tampilkan Semua Antrian
4. Lihat Mahasiswa Prioritas Tertinggi
0. Keluar Program
Pilihan:0
 Program selesai!

Process finished with exit code 0
```