

LAPORAN PRAKTIKUM

MODUL 8

“QUEUE”



Disusun Oleh:

Tiurma Grace Angelina 2311104042

SE-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: Struktur Data FIFO (First-In First-Out)

Queue adalah struktur data yang menerapkan prinsip FIFO (First-In First-Out), di mana elemen yang pertama masuk akan menjadi elemen yang pertama dikeluarkan. Konsep ini mirip dengan antrian dalam kehidupan sehari-hari, misalnya saat menunggu giliran di kasir, di mana orang yang datang terlebih dahulu akan dilayani lebih dulu.

Dalam implementasinya, queue dapat dibangun menggunakan array atau linked list. Struktur dasar queue terdiri dari dua pointer utama:

Front: Menunjuk elemen pertama dalam queue.

Rear: Menunjuk elemen terakhir dalam queue.

Prinsip FIFO pada Queue

Perbedaan utama antara queue dan stack terletak pada cara menambah dan menghapus elemen:

- Stack: Menggunakan metode LIFO (Last-In, First-Out), di mana elemen terakhir yang masuk akan menjadi elemen pertama yang keluar. Operasi pada stack dilakukan di satu ujung, yaitu top. Contohnya adalah tumpukan piring, di mana piring yang terakhir ditambahkan akan diambil pertama kali.
- Queue: Menggunakan metode FIFO, di mana elemen yang masuk lebih awal akan keluar terlebih dahulu. Operasi penambahan elemen

dilakukan di ujung belakang (rear), sedangkan penghapusan elemen dilakukan di ujung depan (front). Misalnya, dalam antrean pelanggan, orang pertama yang datang akan dilayani terlebih dahulu.

Operasi Utama pada Queue

- Enqueue: Menambahkan elemen baru ke bagian belakang (rear) queue.
- Dequeue: Menghapus elemen dari bagian depan (front) queue.
- Peek: Mengambil elemen di depan queue tanpa menghapusnya.
- isEmpty: Mengecek apakah queue kosong.
- isFull: Mengecek apakah queue penuh.
- Size: Menghitung jumlah elemen dalam queue.

3. Guided

- **Array**
 1. CODE :

```

1  #include <iostream>
2  #define MAX 100
3
4  using namespace std;
5
6  class Queue {
7  private:
8      int front, rear;
9      int arr[MAX];
10 public:
11
12     Queue() {
13         front = -1;
14         rear = -1;
15     }
16
17
18     bool isFull() {
19         return rear == MAX - 1;
20     }
21
22
23     bool isEmpty() {
24         return front == -1 || front > rear;
25     }
26
27
28     void enqueue(int x) {
29         if (isFull()) {
30             cout << "Queue Overflow\n";
31             return;
32         }
33         if (front == -1) front = 0;
34         arr[++rear] = x;
35     }
36

```

```

37
38     void dequeue() {
39         if (isEmpty()) {
40             cout << "Queue Underflow\n";
41             return;
42         }
43         front++;
44     }
45
46     int peek() {
47         if (!isEmpty()) {
48             return arr[front];
49         }
50         cout << "Queue is empty\n";
51         return -1;
52     }
53
54
55     void display() {
56         if (isEmpty()) {
57             cout << "Queue is empty\n";
58             return;
59         }
60         for (int i = front; i <= rear; i++) {
61             cout << arr[i] << " ";
62         }
63         cout << "\n";
64     }
65 };
66
67
68 int main() {
69     Queue q;
70
71     q.enqueue(10);
72     q.enqueue(20);
73

```

```

62     }
63     cout << "\n";
64 }
65 };
66
67
68 int main() {
69     Queue q;
70
71     q.enqueue(10);
72     q.enqueue(20);
73     q.enqueue(30);
74
75     cout << "Queue elements: ";
76     q.display();
77
78     cout << "Front element: " << q.peek() << "\n";
79
80     cout << "After dequeuing, queue elements: ";
81     q.display();
82
83     return 0;
84 }
85

```

OUTPUT :

```
C:\Users\USER\Music\laprak\guided1\bin\Debug\guided1.exe
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30

Process returned 0 (0x0)   execution time : 0.105 s
Press any key to continue.
```

2. CODE :

```
1 #include <iostream>
2
3 using namespace std;
4
5 // Node untuk menyimpan Queue
6 class Node {
7 public:
8     int data; // Data elemen
9     Node* next; // Pointer ke node berikutnya
10
11 // Konstruktor untuk Node
12 Node(int value) {
13     data = value;
14     next = nullptr;
15 }
16 };
17
18 // Kelas Queue menggunakan linked list
19 class Queue {
20 private:
21     Node* front; // Pointer ke elemen depan Queue
22     Node* rear; // Pointer ke elemen belakang Queue
23
24 public:
25     // Konstruktor Queue
26     Queue() {
27         front = rear = nullptr;
28     }
29
30     // Meneriksa apakah Queue kosong
31     bool isEmpty() {
32         return front == nullptr;
33     }
34
35     // Menambahkan elemen ke Queue
36     void enqueue(int x) {
37         Node* newNode = new Node(x);
38         if (isEmpty()) {
39             front = rear = newNode;
40         } else {
41             rear->next = newNode;
42             rear = newNode;
43         }
44     }
45
46     // Menghapus elemen dari Queue
47     void dequeue() {
48         if (isEmpty()) {
49             cout << "Queue Underflow!";
50             return;
51         }
52         Node* temp = front;
53         front = front->next;
54         delete temp;
55         if (front == nullptr) {
56             rear = nullptr;
57         }
58     }
59
60     // Menampilkan elemen Queue
61     int peek() {
62         if (isEmpty()) {
63             return -1;
64         }
65         return front->data;
66     }
67
68     // Menampilkan semua elemen di Queue
69     void display() {
70         if (isEmpty()) {
71             cout << "Queue is empty!";
72             return;
73         }
74         Node* current = front;
75         while (current) {
76             cout << current->data << " ";
77             current = current->next;
78         }
79         cout << "\n";
80     }
81 };
82
83 // Fungsi utama untuk menguji Queue
84 int main() {
85     Queue q;
86
87     // Menambahkan elemen ke Queue
88     q.enqueue(10);
89     q.enqueue(20);
90     q.enqueue(30);
91
92     // Menampilkan elemen di Queue
93     cout << "Queue elements: ";
94     q.display();
95
96     // Menampilkan elemen depan
97     cout << "Front element: " << q.peek() << "\n";
98
99     // Menghapus elemen dari Queue
100    q.dequeue();
101    cout << "After dequeuing, queue elements: ";
102    q.display();
103
104    return 0;
105 }
```

OUTPUT :

```
C:\Users\USER\Music\laprak\guided2\bin\Debug\guided2.exe
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30

Process returned 0 (0x0)   execution time : 0.128 s
Press any key to continue.
```

3. CODE :

```

1  #include<iostream>
2
3  using namespace std;
4
5  const int maksimalQueue = 5; // Maksimal antrian
6  int front = 0; // Indeks antrian
7  int back = 0; // Indeks
8  string queueTeller(5); // Fungsi untuk antrian
9
10 bool isFull() { // Memeriksa apakah antrian sudah penuh atau tidak
11     if (back == maksimalQueue) { return true; // =1
12     } else {
13         return false;
14     }
15 }
16
17 bool isEmpty() { // Memeriksa apakah antrian sudah kosong atau tidak
18     if (back == 0) { return true;
19     } else {
20         return false;
21     }
22 }
23
24 void enqueueAntrian(string data) { // Fungsi untuk menambahkan antrian
25     if (isFull()) {
26         cout << "Antrian penuh" << endl;
27     } else {
28         if (isEmpty()) { // Menjalankan ketika queue kosong
29             queueTeller[0] = data; front++;
30             back++;
31         } else { // Menjalankan jika ada queueTeller[back] = data; back++;
32             queueTeller[back] = data; back++;
33         }
34     }
35 }
36
37 void dequeueAntrian() { // Fungsi untuk 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             back--;
43         }
44     }
45 }
46
47 int countQueue() { // Fungsi untuk menghitung banyak antrian
48     return back;
49 }
50
51 void clearQueue() { // Fungsi untuk menghapus semua antrian
52     if (isEmpty()) {
53         cout << "Antrian kosong" << endl;
54     } else {
55         for (int i = 0; i < back; i++) { queueTeller[i] = "";
56             back = 0;
57             front = 0;
58         }
59     }
60 }
61
62 void viewQueue() { // Fungsi untuk melihat antrian
63     cout << "Data antrian teller:" << endl; for (int i = 0; i < maksimalQueue; i++) {
64         if (queueTeller[i] != "") {
65             cout << i + 1 << ". " << queueTeller[i] <<
66             endl;
67         }
68     }
69     } else {
70         cout << i + 1 << ". (kosong)" << endl;
71     }
72 }
73
74
75
76 int main() {
77     enqueueAntrian("Andi");
78
79     enqueueAntrian("Maya");
80
81     viewQueue();
82     cout << "Jumlah antrian = " << countQueue() << endl;
83
84     dequeueAntrian();
85     viewQueue();
86     cout << "Jumlah antrian = " << countQueue() << endl;
87
88     clearQueue();
89     viewQueue();
90     cout << "Jumlah antrian = " << countQueue() << endl;
91
92     return 0;
93 }
94
95

```

OUTPUT :

```
C:\Users\USER\Music\laprak\guided3\bin\Debug\guided3.exe
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
Process returned 0 (0x0)   execution time : 0.103 s
Press any key to continue.
```

4. Unguided

1. Buatlah program yang menerima *input*-an dua buah bilangan betipe float, kemudian memberikan *output*-an hasil penjumlahan, pengurangan, perkalian, dan pembagian dari dua bilangan tersebut.

CODE :

```

1  #include <iostream>
2  #define MAX 100
3
4  using namespace std;
5
6  class Queue {
7  private:
8      int front, rear;
9      int arr[MAX];
10 public:
11
12     Queue() {
13         front = -1;
14         rear = -1;
15     }
16
17     bool isFull() {
18         return rear == MAX - 1;
19     }
20
21     bool isEmpty() {
22         return front == -1 || front > rear;
23     }
24
25     void enqueue(int x) {
26         if (isFull()) {
27             cout << "Queue Overflow\n";
28             return;
29         }
30         if (front == -1) front = 0;
31         arr[++rear] = x;
32     }
33
34
35
36

```

```

34     arr[++rear] = x;
35 }
36
37 void dequeue() {
38     if (isEmpty()) {
39         cout << "Queue Underflow\n";
40         return;
41     }
42     front++;
43 }
44
45 int peek() {
46     if (!isEmpty()) {
47         return arr[front];
48     }
49     cout << "Queue is empty\n";
50     return -1;
51 }
52
53 void display() {
54     if (isEmpty()) {
55         cout << "Queue is empty\n";
56         return;
57     }
58     for (int i = front; i <= rear; i++) {
59         cout << arr[i] << " ";
60     }
61     cout << "\n";
62 }
63
64 };
65
66
67
68 int main() {
69     Queue q;
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85

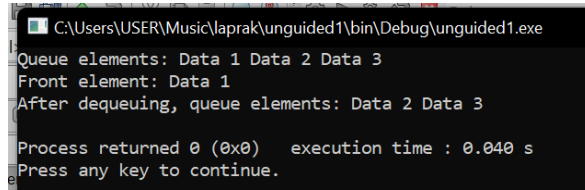
```

```

54 void display() {
55     if (isEmpty()) {
56         cout << "Queue is empty\n";
57         return;
58     }
59     for (int i = front; i <= rear; i++) {
60         cout << arr[i] << " ";
61     }
62     cout << "\n";
63 }
64
65 };
66
67
68 int main() {
69     Queue q;
70
71     q.enqueue(10);
72     q.enqueue(20);
73     q.enqueue(30);
74
75     cout << "Queue elements: ";
76     q.display();
77
78     cout << "Front element: " << q.peek() << "\n";
79
80     cout << "After dequeuing, queue elements: ";
81     q.display();
82
83     return 0;
84 }
85

```


OUTPUT :



```
C:\Users\USER\Music\laprak\unguided1\bin\Debug\unguided1.exe
Queue elements: Data 1 Data 2 Data 3
Front element: Data 1
After dequeuing, queue elements: Data 2 Data 3
Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

Penjelasan

1. Kelas Node

Kelas ini merepresentasikan setiap elemen (node) di dalam Queue.

- Atribut:
 - string data : Menyimpan data string dari elemen node.
 - Node* next : Pointer yang menunjuk ke node berikutnya dalam antrean.
- Constructor:
 - Node(string value) : Menginisialisasi data node dengan nilai yang diberikan dan next diset ke nullptr.

2. Kelas Queue

Kelas ini merepresentasikan Queue yang diimplementasikan menggunakan linked list.

- Atribut:
 - Node front : Pointer menunjuk ke elemen depan Queue.
 - Node rear : Pointer menunjuk ke elemen belakang Queue.

Method

1.Queue()

Konstruktor untuk menginisialisasi Queue kosong dengan front dan rear diatur ke nullptr.

2. bool isEmpty()

Mengecek apakah Queue kosong. Mengembalikan true jika front == nullptr, false jika tidak.

3.void enqueue(string x)

Menambahkan elemen baru ke belakang Queue:

- Membuat node baru dengan data x.
- Jika Queue kosong (isEmpty()), front dan rear menunjuk ke node baru.
- Jika tidak kosong, node baru ditambahkan di belakang, dan rear diperbarui.

4. void dequeue()

Menghapus elemen dari depan Queue:

- Jika Queue kosong, tampilkan pesan "Queue Underflow".
- Jika ada elemen, node depan dihapus, dan front dipindahkan ke node berikutnya.
- Jika elemen terakhir dihapus, rear juga diatur ke nullptr.

5. string peek()

Mengembalikan elemen depan tanpa menghapusnya:

- Jika Queue tidak kosong, kembalikan front->data.
- Jika kosong, tampilkan pesan "Queue is empty" dan kembalikan string kosong.

6. void display()

Menampilkan semua elemen Queue:

- Jika Queue kosong, tampilkan "Queue is empty".
- Jika tidak, iterasi dari front ke rear dan tampilkan data dari setiap node.

2. Buatlah sebuah program yang menerima masukan angka dan mengeluarkan *output* nilai angka tersebut dalam bentuk tulisan. Angka yang akan di- *input*-kan user adalah bilangan bulat positif mulai dari 0 s.d 100.

CODE :

```

1 #include <iostream>
2 using namespace std;
3
4 // Node untuk setiap elemen Mahasiswa
5 class Node {
6 public:
7     string nama; // Nama mahasiswa
8     string nim; // NIM mahasiswa
9     Node* next; // Pointer ke node berikutnya
10
11     // Konstruktor untuk Node
12     Node(string n, string no) {
13         nama = n;
14         nim = no;
15         next = nullptr;
16     }
17 };
18
19 // Kelas Queue Mahasiswa menggunakan linked list
20 class Queue {
21 private:
22     Node* front; // Pointer ke elemen depan Queue
23     Node* rear; // Pointer ke elemen belakang Queue
24 public:
25     // Konstruktor Queue
26     Queue() {
27         front = rear = nullptr;
28     }
29
30     // Menonaktifkan Queue kosong
31     bool isEmpty() {
32         return front == nullptr;
33     }
34
35     // Menambahkan mahasiswa ke Queue
36     void enqueue(string nama, string nim) {
37         Node* newNode = new Node(nama, nim);
38         if (isEmpty()) {
39             front = rear = newNode;
40         } else {
41             rear->next = newNode;
42             rear = newNode;
43         }
44     }
45
46     // Menghapus mahasiswa dari depan Queue
47     void dequeue() {
48         if (isEmpty()) {
49             cout << "Queue Underflow\n";
50             return;
51         }
52         Node* temp = front;
53         front = front->next; // Menghapus front ke node berikutnya
54         delete temp; // Hapus node lama
55         if (front == nullptr) {
56             rear = nullptr; // Jika Queue kosong, rear juga harus null
57         }
58     }
59
60     // Menampilkan mahasiswa dalam Queue dalam urutan tertentu
61     void peek() {
62         if (isEmpty()) {
63             cout << "Nama: " << front->nama << ", NIM: " << front->nim << endl;
64             return;
65         }
66         cout << "Queue is empty\n";
67     }
68
69     // Menampilkan semua mahasiswa di Queue
70     void display() {
71         if (isEmpty()) {
72             cout << "Queue is empty\n";
73             return;
74         }
75         Node* current = front; // Mulai dari depan
76         int nomor = 1; // Inisiasi variabel nomor
77         while (current) { // Iterasi sampai akhir
78             cout << nomor << ". Nama: " << current->nama << ", NIM: " << current->nim << endl;
79             current = current->next;
80             nomor++;
81         }
82     }
83 };
84
85 // Fungsi utama untuk menguji Queue Mahasiswa
86 int main() {
87     Queue q;
88
89     // Menambahkan mahasiswa ke Queue
90     q.enqueue("Andi", "2311110011");
91     q.enqueue("Budi", "2311110022");
92     q.enqueue("Cici", "2311110033");
93
94     // Menampilkan mahasiswa di Queue
95     cout << "Queue Mahasiswa: \n";
96     q.display();
97
98     // Menampilkan mahasiswa dalam
99     cout << "\nMahasiswa di depan antrian: \n";
100    q.peek();
101
102    // Menampilkan mahasiswa dalam
103    cout << "\nSetelah mengeluarkan mahasiswa pertama: \n";
104    q.dequeue();
105    q.display();
106
107    return 0;
108 }

```

```

34 }
35
36 // Menambahkan mahasiswa ke Queue
37 void enqueue(string nama, string nim) {
38     Node* newNode = new Node(nama, nim);
39     if (isEmpty()) {
40         front = rear = newNode; // Jika Queue kosong
41     } else {
42         rear->next = newNode; // Menambahkan node baru ke belakang
43         rear = newNode; // Mengubah rear
44     }
45 }
46
47 // Menghapus mahasiswa dari depan Queue
48 void dequeue() {
49     if (isEmpty()) {
50         cout << "Queue Underflow\n";
51         return;
52     }
53     Node* temp = front;
54     front = front->next; // Menghapus front ke node berikutnya
55     delete temp; // Hapus node lama
56     if (front == nullptr) {
57         rear = nullptr; // Jika Queue kosong, rear juga harus null
58     }
59 }
60
61 // Menampilkan mahasiswa dalam Queue dalam urutan tertentu
62 void peek() {
63     if (isEmpty()) {
64         cout << "Nama: " << front->nama << ", NIM: " << front->nim << endl;
65         return;
66     }
67     cout << "Queue is empty\n";
68 }
69
70 // Menampilkan semua mahasiswa di Queue
71 void display() {
72     if (isEmpty()) {
73         cout << "Queue is empty\n";
74         return;
75     }
76     Node* current = front; // Mulai dari depan
77     int nomor = 1; // Inisiasi variabel nomor
78     while (current) { // Iterasi sampai akhir
79         cout << nomor << ". Nama: " << current->nama << ", NIM: " << current->nim << endl;
80         current = current->next;
81         nomor++;
82     }
83 }
84
85 // Fungsi utama untuk menguji Queue Mahasiswa
86 int main() {
87     Queue q;
88
89     // Menambahkan mahasiswa ke Queue
90     q.enqueue("Andi", "2311110011");
91     q.enqueue("Budi", "2311110022");
92     q.enqueue("Cici", "2311110033");
93
94     // Menampilkan mahasiswa di Queue
95     cout << "Queue Mahasiswa: \n";
96     q.display();
97
98     // Menampilkan mahasiswa dalam
99     cout << "\nMahasiswa di depan antrian: \n";
100    q.peek();
101
102    // Menampilkan mahasiswa dalam
103    cout << "\nSetelah mengeluarkan mahasiswa pertama: \n";
104    q.dequeue();
105    q.display();
106
107    return 0;
108 }

```

OUTPUT :

```

C:\Users\USER\Music\laprak\unguided2\bin\Debug\unguided2.exe
Queue Mahasiswa:
1. Nama: Andi, NIM: 2311110011
2. Nama: Budi, NIM: 2311110022
3. Nama: Cici, NIM: 2311110033

Mahasiswa di depan antrian:
Nama: Andi, NIM: 2311110011

Setelah mengeluarkan mahasiswa pertama:
1. Nama: Budi, NIM: 2311110022
2. Nama: Cici, NIM: 2311110033

Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.

```

Penjelasan :

1. nama: (string)

- Menyimpan nama mahasiswa dalam node.

2. **nim:** (*string*)
Menyimpan NIM (Nomor Induk Mahasiswa) dari mahasiswa.
3. **next:** (*Node pointer*)
Menunjuk ke node berikutnya dalam antrian.
4. **Node(string n, string no):**
Konstruktor yang menginisialisasi atribut nama dan nim berdasarkan parameter n dan no. Pointer next diatur ke nullptr.
5. **front:** (*Node pointer*)
Menunjuk ke elemen depan antrian.
Elemen ini adalah node yang pertama masuk (FIFO).
6. **rear:** (*Node pointer*)
Menunjuk ke elemen belakang antrian.
Node terakhir yang ditambahkan.
7. **Queue():**
Konstruktor yang menginisialisasi front dan rear menjadi nullptr, menandakan bahwa antrian kosong saat pertama kali dibuat.
8. **isEmpty():**
Mengembalikan nilai true jika antrian kosong (front == nullptr), dan false jika tidak.
9. **enqueue(string nama, string nim):**
Menambahkan mahasiswa baru ke belakang antrian:
 - Membuat node baru dengan data nama dan NIM.
 - Jika antrian kosong, node baru menjadi front dan rear.
 - Jika tidak kosong, node baru ditambahkan di belakang antrian dengan memperbarui pointer rear.

10. dequeue():

Menghapus mahasiswa dari depan antrian:

- Jika antrian kosong, menampilkan pesan "Queue Underflow".
- Jika ada elemen, node depan dihapus, dan pointer front diperbarui ke node berikutnya.
- Jika elemen yang dihapus adalah satu-satunya elemen, maka rear juga diatur ke nullptr.

11. peek():

Menampilkan data mahasiswa yang berada di depan antrian tanpa menghapusnya:

- Menampilkan nama dan nim dari node di front.
- Jika antrian kosong, menampilkan pesan "Queue is empty".

12. display():

Menampilkan seluruh mahasiswa dalam antrian dari depan ke belakang:

- Jika antrian kosong, menampilkan pesan "Queue is empty".
- Jika ada elemen, iterasi dari front hingga rear untuk menampilkan semua data mahasiswa.

3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

CODE :

```
1 #include <iostream>
2 #define MAX 100
3 using namespace std;
4
5 // Node untuk setiap elemen Mahasiswa
6 class Node {
7 public:
8     string nama; // Nama mahasiswa
9     string nim; // NIM mahasiswa
10    Node* next; // Pointer ke node berikutnya
11
12    // Konstruktors untuk Node
13    Node(string n, string no) {
14        nama = n;
15        nim = no;
16        next = nullptr;
17    }
18 };
19
20 // Kelas Queue Mahasiswa menggunakan linked list
21 class Queue {
22 private:
23     Node* front; // Pointer ke elemen depan Queue
24     Node* rear; // Pointer ke elemen belakang Queue
25     int size; // Ukuran Queue
26
27 public:
28     // Konstruktors Queue
29     Queue() {
30         front = rear = nullptr;
31         size = 0;
32     }
33
34     // Mengetahui apakah Queue kosong
35     bool isEmpty() {
36         return front == nullptr;
37     }
38
39     // Mengetahui apakah Queue penuh
```

```

37     }
38
39     // Mengetahui apakah Queue penuh
40     bool isFull() {
41         return size >= MAX;
42     }
43
44     // Menambahkan mahasiswa ke Queue
45     void enqueue(string nama, string nim) {
46         if (isFull()) {
47             cout << "Queue Overflow: Tidak bisa menambahkan lebih banyak mahasiswa.\n";
48             return;
49         }
50
51         Node* newNode = new Node(nama, nim);
52         if (isEmpty()) {
53             front = rear = newNode; // Jika Queue kosong
54         } else {
55             rear->next = newNode; // Tambahkan node baru ke belakang
56             rear = newNode; // Ubahlah rear
57         }
58         size++;
59     }
60
61     // Menghapus mahasiswa dari dalam Queue
62     void dequeue() {
63         if (isEmpty()) {
64             cout << "Queue Underflow\n";
65             return;
66         }
67         Node* temp = front; // Simpan node depan untuk dihapus
68         front = front->next; // Ubahlah front ke node berikutnya
69         delete temp; // Hapus node lama
70         if (front == nullptr) // Jika Queue kosong, rear juga harus null
71             rear = nullptr;
72         size--;
73     }
74
75     // Menampilkan Queue, apakah penuh, apakah kosong, apakah overflow
76 }
77
78 // Menampilkan mahasiswa dalam Queue sama dengan peek()
79 void peek() {
80     if (isEmpty()) {
81         cout << "Queue: " << front->nama << ", NIM: " << front->nim << endl;
82     } else {
83         cout << "Queue is empty\n";
84     }
85 }
86
87 // Menampilkan semua mahasiswa di Queue
88 void display() {
89     if (isEmpty()) {
90         cout << "Queue is empty\n";
91         return;
92     }
93     Node* current = front; // Mulai dari depan
94     int nomor = 1;
95     while (current) { // Sampai semua akhir
96         cout << nomor << ", Nama: " << current->nama
97             << ", NIM: " << current->nim << endl;
98         current = current->next;
99         nomor++;
100     }
101 }
102
103 // Menampilkan seluruh dengan prioritas NIM
104 void displayPrioritas() {
105     if (isEmpty()) {
106         cout << "Queue is empty\n";
107         return;
108     }
109
110     // Salin data ke array untuk sorting
111     Node* array[MAX];
112     Node* current = front;
113     int count = 0;
114     while (current) {
115         array[count++] = current;
116         current = current->next;
117     }
118
119     // Sorting array berdasarkan NIM (Bubble Sort)
120     for (int i = 0; i < count - 1; i++) {
121         for (int j = 0; j < count - i - 1; j++) {
122             if (array[j]-nim > array[j + 1]-nim) {
123                 Node* temp = array[j];
124                 array[j] = array[j + 1];
125                 array[j + 1] = temp;
126             }
127         }
128     }
129
130     // Tampilkan dengan urutan prioritas
131     int nomor = 1;
132     for (int i = 0; i < count; i++) {
133         cout << nomor << ", Nama: " << array[i]-nama
134             << ", NIM: " << array[i]-nim << endl;
135         nomor++;
136     }
137 }
138
139 // Fungsi utama untuk memulai Queue Mahasiswa
140 int main() {
141     Queue q;
142
143     // Menambahkan mahasiswa ke Queue
144     q.enqueue("Andi", "2311110022");
145     q.enqueue("Budi", "2311110011");
146     q.enqueue("Cici", "2311110033");
147
148     // Menampilkan mahasiswa di Queue
149     q.display();
150 }

```

```

129     for (int i = 0; i < count; i++) {
130         cout << nomor << ". Nama: " << array[i] << "nama\n";
131         << "NIM: " << array[i] << "nim << endl;
132         nomor++;
133     }
134 }
135 };
136
137 // Fungsi utama untuk menguji Queue Mahasiswa
138 int main()
139 {
140     Queue q;
141
142     // Menambahkan mahasiswa ke Queue
143     q.enqueue("Andi", "2311110022");
144     q.enqueue("Budi", "2311110011");
145     q.enqueue("Cici", "2311110033");
146
147     // Menampilkan mahasiswa di Queue
148     cout << "Queue Mahasiswa: \n";
149     q.display();
150
151     // Menampilkan mahasiswa dengan prioritas
152     cout << "\nMahasiswa di depan antrian: \n";
153     q.peek();
154
155     // Menampilkan antrian dengan prioritas NIM
156     cout << "\nQueue Mahasiswa dengan Prioritas NIM: \n";
157     q.displayPrioritas();
158
159     // Menghapus mahasiswa dari depan Queue
160     q.dequeue();
161
162     cout << "\nSetelah mengeluarkan mahasiswa pertama: \n";
163     q.display();
164
165     return 0;
166 }

```

OUTPUT :

```

C:\Users\USER\Music\laprak\unguided3\bin\Debug\unguided3.exe
Queue Mahasiswa:
1. Nama: Andi, NIM: 2311110022
2. Nama: Budi, NIM: 2311110011
3. Nama: Cici, NIM: 2311110033

Mahasiswa di depan antrian:
Nama: Andi, NIM: 2311110022

Queue Mahasiswa dengan Prioritas NIM:
1. Nama: Budi, NIM: 2311110011
2. Nama: Andi, NIM: 2311110022
3. Nama: Cici, NIM: 2311110033

Setelah mengeluarkan mahasiswa pertama:
1. Nama: Budi, NIM: 2311110011
2. Nama: Cici, NIM: 2311110033

Process returned 0 (0x0)   execution time : 0.066 s
Press any key to continue.

```

Penjelasan :

- **nama:** (*String*) Menyimpan nama mahasiswa.
- **nim:** (*String*) Menyimpan NIM mahasiswa.
- **next:** (*Pointer to Node*) Menunjuk ke node berikutnya dalam antrian.
- **Node(string n, string no):** Konstruktor untuk menginisialisasi nilai nama, nim, dan menyetel next ke nullptr.

- **front:** (*Pointer to Node*) Menunjuk ke elemen terdepan dalam antrian.
- **rear:** (*Pointer to Node*) Menunjuk ke elemen terakhir dalam antrian.
- **size:** (*int*) Menyimpan jumlah elemen saat ini dalam antrian.
- **Queue():** Konstruktor untuk menginisialisasi antrian kosong dengan front, rear sebagai nullptr, dan size diatur ke 0.
- **isEmpty():** (*bool*) Mengembalikan true jika antrian kosong ($\text{front} == \text{nullptr}$), dan false jika tidak.
- **isFull():** (*bool*) Mengembalikan true jika jumlah elemen mencapai batas maksimal ($\text{size} \geq \text{MAX}$), dan false jika tidak.
- **enqueue(string nama, string nim):**
 - Menambahkan elemen baru ke antrian.
 - Jika antrian penuh, mencetak pesan "Queue Overflow".
 - Menambah elemen di posisi rear.
- **dequeue():**
 - Menghapus elemen dari depan antrian.
 - Jika antrian kosong, mencetak pesan "Queue Underflow".
 - Mengupdate pointer front.
- **peek():**
 - Menampilkan elemen di depan antrian tanpa menghapusnya.
 - Jika antrian kosong, mencetak pesan "Queue is empty".
- **display():**
 - Menampilkan semua elemen dalam antrian dari depan ke belakang.
 - Menampilkan pesan "Queue is empty" jika kosong.
- **displayPrioritas():**

- Menampilkan antrian berdasarkan urutan **NIM** secara terurut (ascending).
- Menggunakan vektor sementara untuk menyimpan dan menyortir data berdasarkan NIM.

5. Kesimpulan

Queue adalah struktur data yang beroperasi dengan prinsip FIFO (First-In First-Out), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar. Praktikum ini berhasil menunjukkan implementasi berbagai operasi utama pada queue, seperti enqueue untuk menambahkan elemen, dequeue untuk menghapus elemen, serta peek untuk melihat elemen di bagian depan tanpa menghapusnya. Selain itu, program mampu menampilkan isi queue dan memeriksa apakah queue kosong atau penuh. Implementasi dilakukan menggunakan linked list dan array, memberikan pemahaman mendalam tentang perbedaan kedua pendekatan tersebut. Modifikasi program juga dilakukan untuk memprioritaskan elemen berdasarkan NIM terkecil, memperkenalkan konsep queue dengan prioritas. Praktikum ini menegaskan peran queue dalam berbagai aplikasi nyata, seperti antrian pelanggan, manajemen proses, dan buffer data, sekaligus menjadi fondasi penting untuk mempelajari struktur data yang lebih kompleks di masa mendatang.