

LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN 8

QUEUE



Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

- Mahasiswa mampu menjelaskan definisi dan konsep dari queue
- Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
- Mahasiswa mampu menerapkan operasi tampil data pada queue

II. TOOL

- Visual Studio Code
- GCC

III. DASAR TEORI

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep **antrian** pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. **Front/head** adalah pointer ke elemen pertama dalam queue dan **rear/tail/back** adalah pointer ke elemen terakhir dalam queue.



FIRST IN FIRST OUT (FIFO)

Perbedaan antara *stack* dan *queue* terletak pada aturan penambahan dan penghapusan elemen. Pada *stack*, operasi penambahan dan penghapusan elemen dilakukan di satu ujung yang disebut *top* (ujung atas). Elemen yang terakhir kali dimasukkan ke dalam *stack* akan berada di posisi paling atas dan akan menjadi elemen pertama yang dihapus. Sifat ini dikenal dengan istilah *LIFO* (Last In, First Out). Contoh analogi sederhana dari *stack* adalah tumpukan piring, di mana piring terakhir yang ditambahkan berada di posisi paling atas dan akan diambil atau dihapus terlebih dahulu.

Sebaliknya, pada *queue*, operasi penambahan dan penghapusan elemen dilakukan di dua ujung yang berbeda. Elemen baru ditambahkan di ujung belakang (*rear* atau *tail*), dan elemen dihapus dari ujung depan (*front* atau *head*). Proses ini mengikuti prinsip *FIFO* (First In, First Out), yang berarti elemen pertama yang dimasukkan ke dalam *queue* akan menjadi elemen pertama yang dikeluarkan. Dalam konteks *queue*, operasi penambahan elemen dikenal sebagai **Enqueue**, dan operasi penghapusan elemen disebut **Dequeue**.

Pada *Enqueue*, elemen ditambahkan di belakang *queue* setelah elemen terakhir yang ada, sementara pada *Dequeue*, elemen paling depan (*head*) dihapus, dan posisi *head* akan

bergeser ke elemen berikutnya. Contoh penggunaan *queue* dalam kehidupan sehari-hari adalah antrean di kasir, di mana orang pertama yang datang adalah yang pertama dilayani.

Operasi pada Queue

- `enqueue()` : menambahkan data ke dalam queue.
- `dequeue()` : mengeluarkan data dari queue.
- `peek()` : mengambil data dari queue tanpa menghapusnya.
- `isEmpty()` : mengecek apakah queue kosong atau tidak.
- `isFull()` : mengecek apakah queue penuh atau tidak.
- `size()` : menghitung jumlah elemen dalam queue

IV. GUIDED

1. Guided1.cpp

```

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     void dequeue() {
35         if (isEmpty()) {
36             cout << "Queue Underflow\n";
37             return;
38         }
39         front++;
40     }
41
42     int peek() {
43         if (!isEmpty()) {
44             return arr[front];
45         }
46         cout << "Queue is empty\n";
47         return -1;
48     }
49
50     void display() {
51         if (isEmpty()) {
52             cout << "Queue is empty\n";
53             return;
54         }
55         for (int i = front; i <= rear; i++) {
56             cout << arr[i] << " ";
57         }
58         cout << "\n";
59     }
60 };
61
62 int main() {
63     Queue q;
64
65     q.enqueue(10);
66     q.enqueue(20);
67     q.enqueue(30);
68
69     cout << "Queue elements: ";
70     q.display();
71
72     cout << "Front element: " << q.peek() << "\n";
73
74     cout << "After dequeuing, queue elements: ";
75     q.display();
76
77     return 0;
78 }

```

Output :

```
PS D:\PRAKTIKUM\Struktur Data\pertemuan8> &
er.exe' '--stdin=Microsoft-MIEngine-In-2h1iww
Microsoft-MIEngine-Pid-qr5kyqun.1ta' '--dbgExe
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS D:\PRAKTIKUM\Struktur Data\pertemuan8>
```

2. Guided2.cpp

```

1  #include <iostream>
2
3  using namespace std;
4
5  // Node untuk setiap elemen 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     // Mengecek 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; // Jika Queue k
40 osong
41             return;
42         }
43         rear->next = newNode; // Tambahkan node bar
44 u ke belakang
45         rear = newNode;      // Perbarui rear
46     }
47
48     // Menghapus elemen dari depan Queue
49     void dequeue() {
50         if (isEmpty()) {
51             cout << "Queue Underflow\n";
52             return;
53         }
54         Node* temp = front;    // Simpan node dep
55 an untuk dihapus
56         front = front->next;   // Pindahkan front
57 ke node berikutnya
58         delete temp;           // Hapus node lama
59         if (front == nullptr)  // Jika Queue koso
60 ng, rear juga harus null
61             rear = nullptr;
62     }
63
64     // Mengembalikan elemen depan Queue tanpa mengh
65 apusnya
66     int peek() {
67         if (!isEmpty()) {
68             return front->data;
69         }
70         cout << "Queue is empty\n";
71         return -1; // Nilai sentinel
72     }
73
74     // Menampilkan semua elemen di Queue
75     void display() {
76         if (isEmpty()) {
77             cout << "Queue is empty\n";
78             return;
79         }
80         Node* current = front; // Mulai dari depan
81         while (current) {      // Iterasi sampai ak
82 hir
83             cout << current->data << " ";
84             current = current->next;
85         }
86         cout << "\n";
87     }
88 };
89
90 // Fungsi utama untuk menguji Queue
91 int main() {
92     Queue q;
93
94     // Menambahkan elemen ke Queue
95     q.enqueue(10);
96     q.enqueue(20);
97     q.enqueue(30);
98
99     // Menampilkan elemen di Queue
100    cout << "Queue elements: ";
101    q.display();
102
103    // Menampilkan elemen depan
104    cout << "Front element: " << q.peek() << "\n";
105
106    // Menghapus elemen dari depan Queue
107    q.dequeue();
108    cout << "After dequeuing, queue elements: ";
109    q.display();
110
111    return 0;
112 }

```

Output :

```
Microsoft-MIEngine-Pid-wyzdognd.jys' '--dbgE  
Queue elements: 10 20 30  
Front element: 10  
After dequeuing, queue elements: 20 30  
PS D:\PRAKTIKUM\Struktur Data\pertemuan8>
```

3. 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 ti
    dak
11 if (back == maksimalQueue) { return true; // =1
12 } else {
13 return false;
14 }
15 }
16
17 bool isEmpty() { // Antriannya kosong atau tidak
18 if (back == 0) { return true;
19 } else {
20 return false;
21 }
22 }
23
24 void enqueueAntrian(string data) { // Fungsi menamb
    ahkan 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 { // Antrianya ada isi queueTeller[back] = d
    ata; back++;
32 }
33 }
34 }
35
36 void dequeueAntrian() { // Fungsi mengurangi antria
    n
37 if (isEmpty()) {
38 cout << "Antrian kosong" << endl;
39 } else {
40 for (int i = 0; i < back; i++) { queueTeller[i] = q
    ueueTeller[i + 1];
41 }
42 back--;
43 }
44 }
45
46 int countQueue() { // Fungsi menghitung banyak antri
    an
47 return back;
48 }
49
50 void clearQueue() { // Fungsi menghapus semua antri
    an
51 if (isEmpty()) {
52 cout << "Antrian kosong" << endl;
53 } else {
54 for (int i = 0; i < back; i++) { queueTeller[i] =
    "";
55 }
56 back = 0;
57 front = 0;
58 }
59 }
60
61 void viewQueue() { // Fungsi melihat antrian
62 cout << "Data antrian teller:" << endl; for (int i
    = 0; i < maksimalQueue; i++) {
63 if (queueTeller[i] != "") {
64 cout << i + 1 << ". " << queueTeller[i] <<
65 endl;
66 }
67 }
68 } else {
69 cout << i + 1 << ". (kosong)" << endl;
70 }
71 }
72 }
73 }
74 }
75
76 int main() {
77 enqueueAntrian("Andi");
78
79 enqueueAntrian("Maya");
80
81 viewQueue();
82 cout << "Jumlah antrian = " << countQueue() << end
    l;
83
84 dequeueAntrian();
85 viewQueue();
86 cout << "Jumlah antrian = " << countQueue() << end
    l;
87
88 clearQueue();
89 viewQueue();
90 cout << "Jumlah antrian = " << countQueue() << end
    l;
91
92 return 0;
93 }

```


Output :

```
Microsoft-MIEngine-Pid-pora1hp1.rvq' '--dbg
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
PS D:\PRAKTIKUM\Struktur Data\pertemuan8>
```

V. UNGUIDED

1. Unguided1.cpp

```

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

```

Output :

```
er.exe' '--stdin=Microsoft-MIEngine-In-3kmdcj  
Microsoft-MIEngine-Pid-rvey2gd0.iak' '--dbgExe  
Queue elements: 10 20 30  
Front element: 10  
After dequeuing, queue elements: 20 30  
PS D:\PRAKTIKUM\Struktur Data\pertemuan8>
```

2. Unguided2.cpp

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class Node {
6 public:
7     string name; // Nama Mahasiswa
8     string nim; // NIM Mahasiswa
9     Node* next; // Pointer ke node berikutnya
10
11     Node(string name, string nim) {
12         this->name = name;
13         this->nim = nim;
14         this->next = nullptr;
15     }
16 };
17
18 class Queue {
19 private:
20     Node* front; // Pointer ke elemen terdepan
21     Node* rear; // Pointer ke elemen terakhir
22
23 public:
24     Queue() {
25         front = nullptr;
26         rear = nullptr;
27     }
28
29     // Cek apakah queue kosong
30     bool isEmpty() {
31         return front == nullptr;
32     }
33
34     // Menambahkan mahasiswa ke dalam queue (enqueue)
35     void enqueue(string name, string nim) {
36         Node* newNode = new Node(name, nim);
37         if (rear == nullptr) { // Jika queue kosong
38             front = rear = newNode;
39         } else {
40             rear->next = newNode; // tambahkan node di akhir
41             rear = newNode; // update rear
42         }
43         cout << "Mahasiswa " << name << " dengan NIM " << nim << " telah ditambahkan ke antrian\n";
44     }
45
46     // Menghapus mahasiswa dari queue (dequeue)
47     void dequeue() {
48         if (!isEmpty()) {
49             cout << "Queue Underflow\n";
50             return;
51         }
52         Node* temp = front; // Simpan elemen yang akan dihapus
53         front = front->next; // Geser front ke elemen berikutnya
54         if (front == nullptr) // Jika queue kosong, reset rear
55             rear = nullptr;
56         cout << "Mahasiswa " << temp->name << " dengan NIM " << temp->nim << " telah dihapus dari antrian\n";
57         delete temp; // dealokasi node
58     }
59
60     // Menampilkan mahasiswa di depan tanpa menghapusnya
61     void peek() {
62         if (!isEmpty()) {
63             cout << "Nama Mahasiswa: " << front->name << ", NIM: " << front->nim << "\n";
64         } else {
65             cout << "Queue is empty\n";
66         }
67     }
68
69     // Menampilkan semua mahasiswa dalam queue
70     void display() {
71         if (!isEmpty()) {
72             cout << "Queue is empty\n";
73             return;
74         }
75         Node* temp = front;
76         cout << "Antrian Mahasiswa:\n";
77         while (temp != nullptr) {
78             cout << "Nama: " << temp->name << ", NIM: " << temp->nim << "\n";
79             temp = temp->next;
80         }
81     }
82
83     // Destructor untuk membebaskan memori
84     ~Queue() {
85         while (!isEmpty()) {
86             dequeue();
87         }
88     }
89 };
90
91 int main() {
92     Queue q;
93     int choice;
94     string name, nim;
95
96     do {
97         cout << "\nMenu Antrian Mahasiswa\n";
98         cout << "1. Tambah Mahasiswa (enqueue)\n";
99         cout << "2. Hapus Mahasiswa (dequeue)\n";
100         cout << "3. Lihat Mahasiswa di Depan (peek)\n";
101         cout << "4. Tampilkan Semua Mahasiswa\n";
102         cout << "5. Keluar\n";
103         cout << "Masukkan pilihan: ";
104         cin >> choice;
105
106         if (cin.fail()) {
107             cin.clear(); // Reset error flag
108             cin.ignore(1000, '\n'); // Hapus input yang sudah dari buffer
109             cout << "Input tidak valid! Masukkan angka.\n";
110             continue; // Kembali ke awal loop
111         }
112
113         switch (choice) {
114             case 1:
115                 cout << "Masukkan Nama Mahasiswa: ";
116                 cin.ignore();
117                 getline(cin, name);
118                 cout << "Masukkan NIM Mahasiswa: ";
119                 cin >> nim;
120                 q.enqueue(name, nim);
121                 break;
122             case 2:
123                 q.dequeue();
124                 break;
125             case 3:
126                 q.peek();
127                 break;
128             case 4:
129                 q.display();
130                 break;
131             case 5:
132                 cout << "Keluar dari program.\n";
133                 break;
134             default:
135                 cout << "Pilihan tidak valid. Coba lagi.\n";
136                 break;
137         }
138     } while (choice != 5);
139
140     return 0;
141 }

```

Output :

```
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Budi Santoso
Masukkan NIM Mahasiswa: 12345
Mahasiswa Budi Santoso dengan NIM 12345 telah ditambahkan ke antrian
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Asep Kopling
Masukkan NIM Mahasiswa: 67890
Mahasiswa Asep Kopling dengan NIM 67890 telah ditambahkan ke antrian
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Rizal Sosis
Masukkan NIM Mahasiswa: 54321
Mahasiswa Rizal Sosis dengan NIM 54321 telah ditambahkan ke antrian
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 3
Nama Mahasiswa: Budi Santoso, NIM: 12345
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 4
Antrian Mahasiswa:
Nama: Budi Santoso, NIM: 12345
Nama: Asep Kopling, NIM: 67890
Nama: Rizal Sosis, NIM: 54321
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 2
Mahasiswa Budi Santoso dengan NIM 12345 telah dihapus dari antrian

Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 4
Antrian Mahasiswa:
Nama: Asep Kopling, NIM: 67890
Nama: Rizal Sosis, NIM: 54321
```

3. Unguided3.cpp

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class Node {
6 public:
7     string name; // Nama Mahasiswa
8     string nim; // NIM Mahasiswa
9     Node* next; // Pointer ke node berikutnya
10
11     Node(string name, string nim) {
12         this->name = name;
13         this->nim = nim;
14         this->next = nullptr;
15     }
16 };
17
18 class PriorityQueue {
19 private:
20     Node* front; // Pointer ke elemen terdepan
21
22 public:
23     PriorityQueue() {
24         front = nullptr;
25     }
26
27     // Cek apakah queue kosong
28     bool isEmpty() {
29         return front == nullptr;
30     }
31
32     // Menambahkan mahasiswa ke dalam queue (enqueue)
33     void enqueue(string name, string nim) {
34         Node* newNode = new Node(name, nim);
35         if (!isEmpty()) { // jika front != nullptr [ // Skipkan di depan jika queue kosong atau UTM lebih
36             // ke belakang
37             newNode->next = front;
38             front = newNode;
39         } else {
40             // Cari posisi yang sesuai berdasarkan NIM
41             Node* temp = front;
42             while (temp->next != nullptr && temp->next->nim < nim) {
43                 temp = temp->next;
44             }
45             newNode->next = temp->next;
46             temp->next = newNode;
47         }
48     }
49
50     // Menghapus mahasiswa dari queue (dequeue)
51     void dequeue() {
52         if (!isEmpty()) {
53             cout << "Queue Underflow\n";
54             return;
55         }
56         Node* temp = front;
57         front = front->next;
58         delete temp;
59     }
60
61     // Menampilkan mahasiswa di depan tanpa menghapusnya
62     void peek() {
63         if (!isEmpty()) {
64             cout << "Nama Mahasiswa: " << front->name << ", NIM: " << front->nim << "\n";
65         } else {
66             cout << "Queue is empty\n";
67         }
68     }
69
70     // Menampilkan semua mahasiswa dalam queue
71     void display() {
72         if (!isEmpty()) {
73             cout << "Queue is empty\n";
74             return;
75         }
76         Node* temp = front;
77         cout << "Antrian Mahasiswa (Prioritas NIM Kecil):\n";
78         while (temp != nullptr) {
79             cout << "Nama: " << temp->name << ", NIM: " << temp->nim << "\n";
80             temp = temp->next;
81         }
82     }
83
84     // Destructor untuk membersihkan memori
85     ~PriorityQueue() {
86         while (!isEmpty()) {
87             dequeue();
88         }
89     }
90 };
91
92 int main() {
93     PriorityQueue pq;
94     int choice;
95     string name, nim;
96
97     do {
98         cout << "\nMenu Antrian Mahasiswa\n";
99         cout << "1. Tambah Mahasiswa (Enqueue)\n";
100         cout << "2. Hapus Mahasiswa (Dequeue)\n";
101         cout << "3. Lihat Mahasiswa di Depan (Peek)\n";
102         cout << "4. Tampilkan Semua Mahasiswa\n";
103         cout << "5. Keluar\n";
104         cout << "Masukkan pilihan: ";
105         cin >> choice;
106
107         // Cek apakah input valid
108         if (cin.get() != '\n') {
109             cin.clear();
110             cin.ignore(1000, '\n');
111             cout << "Input tidak valid! Masukkan angka.\n";
112             continue;
113         }
114
115         switch (choice) {
116             case 1:
117                 cout << "Masukkan Nama Mahasiswa: ";
118                 cin.ignore();
119                 getline(cin, name);
120                 cout << "Masukkan NIM Mahasiswa: ";
121                 cin >> nim;
122                 pq.enqueue(name, nim);
123                 break;
124             case 2:
125                 pq.dequeue();
126                 break;
127             case 3:
128                 pq.peek();
129                 break;
130             case 4:
131                 pq.display();
132                 break;
133             case 5:
134                 cout << "Keluar dari program.\n";
135                 break;
136             default:
137                 cout << "Pilihan tidak valid. Coba lagi.\n";
138         }
139     } while (choice != 5);
140
141     return 0;
142 }

```


Output :

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Asep Kopling
Masukkan NIM Mahasiswa: 54321
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Sigit Rendang
Masukkan NIM Mahasiswa: 98765
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 1
Masukkan Nama Mahasiswa: Budi Santoso
Masukkan NIM Mahasiswa: 12345
```

```
Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 2

Menu Antrian Mahasiswa
1. Tambah Mahasiswa (Enqueue)
2. Hapus Mahasiswa (Dequeue)
3. Lihat Mahasiswa di Depan (Peek)
4. Tampilkan Semua Mahasiswa
5. Keluar
Masukkan pilihan: 4
Antrian Mahasiswa (Prioritas NIM Kecil):
Nama: Asep Kopling, NIM: 54321
Nama: Sigit Rendang, NIM: 98765
```

VI. KESIMPULAN

Pada praktikum ini, kami mempelajari implementasi struktur data Queue menggunakan bahasa C++. Queue adalah struktur data yang menerapkan prinsip FIFO (First In, First Out), di mana elemen yang pertama kali dimasukkan akan

menjadi yang pertama kali keluar. Praktikum ini melibatkan pembuatan Queue menggunakan array dan linked list.

Langkah pertama adalah mendefinisikan struktur data Queue yang terdiri dari dua operasi utama, yaitu enqueue (menambahkan elemen ke dalam antrian) dan dequeue (menghapus elemen dari antrian). Operasi enqueue memeriksa apakah antrian penuh, sedangkan operasi dequeue memeriksa apakah antrian kosong.

Selanjutnya, implementasi dilakukan menggunakan array dengan ukuran tetap dan menggunakan linked list dinamis. Pada implementasi array, pointer untuk elemen depan dan belakang antrian dikelola untuk memudahkan proses enqueue dan dequeue. Pada implementasi linked list, setiap elemen diwakili oleh sebuah node yang memiliki pointer ke node berikutnya.

Hasil praktikum menunjukkan bahwa implementasi Queue dengan array lebih efisien dalam hal penggunaan memori tetapi terbatas pada ukuran tetap, sementara linked list lebih fleksibel dalam mengelola ukuran antrian namun memiliki overhead lebih besar dalam manajemen memori.

Secara keseluruhan, praktikum ini memberikan pemahaman yang lebih baik tentang cara kerja dan penerapan struktur data Queue dalam bahasa C++.