

**LAPORAN PRAKTIKUM
STRUKTUR DATA
MODUL 8
“ QUEUE ”**



Disusun Oleh:
Dhiya Ulhaq Ramadhan 2211104053
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

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

2. Landasan Teori

Queue merupakan struktur data yang mengimplementasikan prinsip FIFO (First-In First-Out), dimana data yang pertama dimasukkan akan menjadi data yang pertama dikeluarkan. Konsep ini dapat dianalogikan dengan sistem antrian dalam kehidupan sehari-hari, seperti antrian di loket pembelian tiket kereta api dimana orang yang pertama datang akan dilayani terlebih dahulu.

Dalam implementasinya, queue dapat direpresentasikan menggunakan array atau linked list. Struktur queue memiliki dua pointer utama: front (atau head) yang menunjuk ke elemen pertama, dan rear (atau tail/back) yang menunjuk ke elemen terakhir dalam antrian. Queue memiliki beberapa operasi dasar seperti enqueue untuk menambah elemen di akhir antrian, dequeue untuk menghapus elemen dari depan antrian, peek untuk melihat elemen terdepan tanpa menghapusnya, dan operasi pengecekan seperti isEmpty dan isFull.

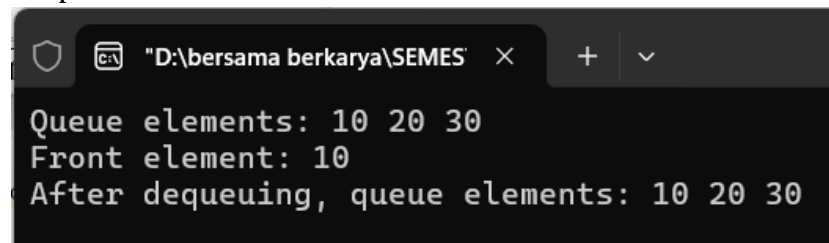
3. Guided 1

Source code :

```
guided1.cpp X
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     }
```

```
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     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 }
```

Output :



```
"D:\bersama berkarya\SEMES" x + v
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
```

Deskripsi program

Program ini mengimplementasikan struktur data Queue menggunakan array statis dengan kapasitas maksimum 100 elemen. Dalam fungsi main(), program mendemonstrasikan penggunaan Queue dengan menambahkan tiga elemen (10, 20, dan 30), menampilkan isi queue, melihat elemen depan, dan kemudian mengeluarkan

elemen pertama. Implementasi ini menggunakan array statis dengan batasan ukuran, sehingga memiliki kapasitas terbatas dan akan menghasilkan pesan "Queue Overflow" jika mencoba menambahkan elemen melebihi kapasitas maksimum. Metode isEmpty() dan isFull() membantu mencegah operasi yang tidak valid pada queue, seperti mengeluarkan elemen dari queue kosong atau menambahkan elemen ke queue yang sudah penuh.

4. Guided 2

```

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 kosong
40             return;
41         }
42         rear->next = newNode; // Tambahkan node baru ke belakang
43         rear = newNode;      // Perbarui rear
44     }
45
46     // Menghapus elemen dari depan Queue
47     void dequeue() {
48         if (isEmpty()) {
49             cout << "Queue Underflow\n";
50             return;
51         }
52         Node* temp = front; // Simpan node depan untuk dihapus
53         front = front->next; // Pindahkan front ke node berikutnya
54         delete temp;         // Hapus node lama
55         if (front == nullptr) // Jika Queue kosong, rear juga harus null
56             rear = nullptr;
57     }

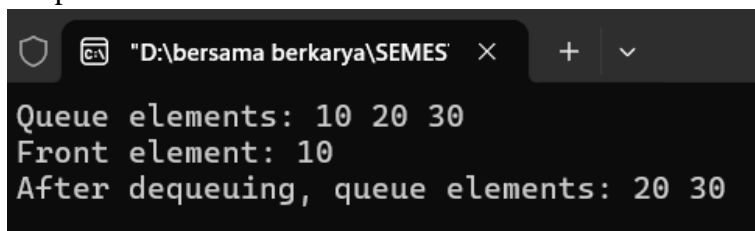
```

```

59 // Mengembalikan elemen depan Queue tanpa menghapusnya
60 int peek() {
61     if (!isEmpty()) {
62         return front->data;
63     }
64     cout << "Queue is empty\n";
65     return -1; // Nilai sentinel
66 }
67
68 // Menampilkan semua elemen di Queue
69 void display() {
70     if (isEmpty()) {
71         cout << "Queue is empty\n";
72         return;
73     }
74     Node* current = front; // Mulai dari depan
75     while (current) {      // Iterasi sampai akhir
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 depan Queue
100    q.dequeue();
101    cout << "After dequeuing, queue elements: ";
102    q.display();
103
104    return 0;
105 }

```

Output :



```

Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30

```

Deskripsi Program

Program dimulai dengan membuat objek Queue q di fungsi main(). Kemudian dilakukan tiga operasi enqueue berturut-turut dengan nilai 10, 20, dan 30. Saat proses ini berlangsung, setiap pemanggilan enqueue membuat node baru dan

menghubungkannya ke struktur linked list yang ada.

Ketika method display() dipanggil pertama kali, program akan mencetak semua elemen queue, yaitu 10 20 30. Proses ini dilakukan dengan mengiterasi seluruh node mulai dari front hingga node terakhir.

Selanjutnya, peek() dipanggil untuk menampilkan elemen depan queue, yang akan mencetak 10 - yaitu elemen pertama yang dimasukkan.

Kemudian method dequeue() dipanggil, yang akan menghapus node pertama (berisi angka 10) dari queue. Pointer front akan dipindahkan ke node berikutnya, dan memori untuk node lama akan dikembalikan.

Setelah dequeue, method display() dipanggil kembali, kali ini akan mencetak 20 30 - menunjukkan bahwa elemen pertama (10) telah berhasil dihapus dari queue.

5. Guided 3

Source code :

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

```

36 void dequeueAntrian() { // Fungsi mengurangi antrian
37     if (isEmpty()) {
38         cout << "Antrian kosong" << endl;
39     } else {
40         for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
41         }
42         back--;
43     }
44 }
45
46 int countQueue() { // Fungsi menghitung banyak antrian
47     return back;
48 }
49
50 void clearQueue() { // Fungsi menghapus semua antrian
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         } else {
67             cout << i + 1 << ". (kosong)" << endl;
68         }
69     }
70 }
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 }

```

Output

```
"D:\bersama berkarya\SEMES" X
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
```

Penjelasan Program

Program ini mendemonstrasikan implementasi struktur data antrian (queue) menggunakan array statis dengan kapasitas maksimal 5 elemen.

pertama-tama dilakukan penambahan dua nama ke dalam antrian: "Andi" dan "Maya" menggunakan fungsi enqueueAntrian(). Proses ini akan menempatkan "Andi" di posisi pertama, kemudian "Maya" di posisi kedua.

Pemanggilan viewQueue() pertama akan menampilkan seluruh daftar antrian, menunjukkan posisi dan nama yang sudah masuk. Pada tahap ini akan terlihat "1. Andi" dan "2. Maya", sementara slot 3-5 akan bertuliskan "(kosong)".

Fungsi countQueue() akan mengembalikan jumlah antrian aktif, yang dalam kasus ini adalah 2. Selanjutnya, pemanggilan dequeueAntrian() akan menghapus elemen pertama, yaitu "Andi".

Setelah dequeue, viewQueue() kembali dipanggil, yang sekarang akan menampilkan "1. Maya" di posisi pertama. Jumlah antrian pun berkurang menjadi 1.

Terakhir, clearQueue() dipanggil untuk mengosongkan seluruh antrian. Pemanggilan terakhir viewQueue() akan menampilkan semua slot sebagai "(kosong)", dan countQueue() akan mengembalikan 0.

UNGUIDED

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

Jawaban :

Source code


```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Node {
6  public:
7      int data;
8      Node* next;
9
10     Node(int nilai) {
11         data = nilai;
12         next = nullptr;
13     }
14 };
15
16 // Kelas Queue menggunakan linked list
17 class Queue {
18 private:
19     Node* depan;
20     Node* belakang;
21     int jumlahData;
22
23 public:
24     Queue() {
25         depan = nullptr;
26         belakang = nullptr;
27         jumlahData = 0;
28     }
29
30     bool apakahKosong() {
31         return depan == nullptr;
32     }
33
34     void tambahData(int nilai) {
35         Node* nodeBaru = new Node(nilai);
36         jumlahData++;
37
38         if (apakahKosong()) {
39             depan = belakang = nodeBaru;
40             cout << "\nData " << nilai << " berhasil ditambahkan sebagai data pertama\n";
41             return;
42         }
43
44         belakang->next = nodeBaru;
45         belakang = nodeBaru;
46         cout << "\nData " << nilai << " berhasil ditambahkan ke queue\n";
47     }
48
49     void hapusData() {
50         if (apakahKosong()) {
51             cout << "\nQueue masih kosong!\n";
52             return;
53         }
54
55         Node* temp = depan;
56         depan = depan->next;
57         cout << "\nData " << temp->data << " berhasil dihapus dari queue\n";
58         delete temp;
59         jumlahData--;
60
61         if (depan == nullptr) {
62             belakang = nullptr;
```

```

63     }
64 }
65
66 void tampilkanData() {
67     if (apakahKosong()) {
68         cout << "\nQueue masih kosong!\n";
69         return;
70     }
71
72     cout << "\nIsi Queue (Total: " << jumlahData << " data):\n";
73     cout << "=====\n";
74     Node* current = depan;
75     int nomor = 1;
76     while (current != nullptr) {
77         cout << nomor << ". " << current->data << endl;
78         current = current->next;
79         nomor++;
80     }
81     cout << "=====\n";
82 }
83 };
84
85 int main() {
86     Queue antrian;
87     int pilihan, nilai;
88
89     do {
90         cout << "\n=== PROGRAM QUEUE LINKED LIST ===\n";
91         cout << "1. Tambah Data\n";
92         cout << "2. Hapus Data\n";
93         cout << "3. Tampilkan Queue\n";
94         cout << "4. Keluar\n";
95         cout << "Pilihan Anda (1-4): ";
96         cin >> pilihan;
97
98         switch (pilihan) {
99             case 1:
100                 cout << "Masukkan nilai: ";
101                 cin >> nilai;
102                 antrian.tambahData(nilai);
103                 break;
104             case 2:
105                 antrian.hapusData();
106                 break;
107             case 3:
108                 antrian.tampilkanData();
109                 break;
110             case 4:
111                 cout << "\nProgram selesai!\n";
112                 break;
113             default:
114                 cout << "\nPilihan tidak valid!\n";
115         }
116     } while (pilihan != 4);
117
118     return 0;
119 }

```

Output :

```
"D:\bersama berkarya\SEMES" x + v

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 1
Masukkan nilai: 12

Data 12 berhasil ditambahkan sebagai data pertama

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 1
Masukkan nilai: 23

Data 23 berhasil ditambahkan ke queue

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 1
Masukkan nilai: 45

Data 45 berhasil ditambahkan ke queue

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 3

Isi Queue (Total: 3 data):
=====
1. 12
2. 23
3. 45
=====

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 2

Data 12 berhasil dihapus dari queue

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 3

Isi Queue (Total: 2 data):
=====
1. 23
2. 45
=====

=== PROGRAM QUEUE LINKED LIST ===
1. Tambah Data
2. Hapus Data
3. Tampilkan Queue
4. Keluar
Pilihan Anda (1-4): 4

Program selesai!
```

Deskripsi program :

Program dimulai dengan mendefinisikan dua kelas utama :

Kelas pertama adalah Node yang berfungsi sebagai struktur dasar untuk menyimpan data. Setiap Node memiliki dua komponen: data bertipe integer dan pointer next yang menunjuk ke Node berikutnya.

Kelas kedua adalah Queue yang mengatur seluruh operasi antrian. Queue memiliki tiga atribut private: pointer depan dan belakang untuk melacak posisi awal dan akhir antrian, serta variabel jumlahData untuk menghitung jumlah elemen dalam antrian.

Ketika program dijalankan, pengguna akan melihat menu dengan empat pilihan: Tambah Data, Hapus Data, Tampilkan Queue, dan Keluar.

Jika pengguna memilih opsi 1 (Tambah Data), program akan meminta input nilai yang ingin ditambahkan. Fungsi tambahData akan membuat Node baru dengan nilai tersebut. Jika Queue kosong, Node baru akan menjadi elemen pertama.

Untuk opsi 2 (Hapus Data), fungsi hapusData akan menghapus elemen terdepan dari Queue sesuai prinsip FIFO. Jika Queue kosong, program akan menampilkan pesan peringatan. Jika tidak, elemen terdepan dihapus dengan memindahkan pointer depan ke Node berikutnya dan menghapus Node lama dari memori.

Ketika pengguna memilih opsi 3 (Tampilkan Queue), fungsi tampilkanData akan menampilkan semua data dalam Queue secara berurutan dari depan ke belakang, beserta jumlah total data yang tersimpan.

Program akan terus berjalan hingga pengguna memilih opsi 4 (Keluar), yang akan menghentikan loop dan mengakhiri program.

2. Dari nomor 1 buatlah konsep antri dengan atribut nama mahasiswa dan NIM mahasiswa

JAWABAN :

Source code :

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Node {
6  public:
7      string nama;
8      string nim;
9      Node* next;
10
11  Node(string namaMhs, string nimMhs) {
12      nama = namaMhs;
13      nim = nimMhs;
14      next = nullptr;
15  }
16 };
17
18 class QueueMahasiswa {
19 private:
20     Node* depan;
21     Node* belakang;
22     int jumlahMahasiswa;
23
24 public:
25     QueueMahasiswa() {
26         depan = nullptr;
27         belakang = nullptr;
28         jumlahMahasiswa = 0;
29     }
30
31     bool apakahKosong() {
32         return depan == nullptr;
33     }
34
35     void tambahMahasiswa(string nama, string nim) {
36         Node* nodeBaru = new Node(nama, nim);
37         jumlahMahasiswa++;
38
39         if (apakahKosong()) {
40             depan = belakang = nodeBaru;
41             cout << "\nMahasiswa pertama berhasil ditambahkan\n";
42             return;
43         }
44
45         belakang->next = nodeBaru;
46         belakang = nodeBaru;
47         cout << "\nMahasiswa berhasil ditambahkan ke queue\n";
48     }
49
50     void hapusMahasiswa() {
51         if (apakahKosong()) {
52             cout << "\nQueue masih kosong!\n";
53             return;
54         }
55
56         Node* temp = depan;
57         depan = depan->next;
58         cout << "\nMahasiswa " << temp->nama << " (NIM: " << temp->nim << ") berhasil dihapus\n";
59         delete temp;
60         jumlahMahasiswa--;
61
62         if (depan == nullptr) {
```

```

63         belakang = nullptr;
64     }
65 }
66
67 void tampilkanMahasiswa() {
68     if (apakahKosong()) {
69         cout << "\nQueue masih kosong!\n";
70         return;
71     }
72
73     cout << "\nDaftar Mahasiswa (Total: " << jumlahMahasiswa << " mahasiswa):\n";
74     cout << "=====\n";
75     Node* current = depan;
76     int nomor = 1;
77     while (current != nullptr) {
78         cout << nomor << ". Nama: " << current->nama << "\n  NIM : " << current->nim << endl;
79         current = current->next;
80         nomor++;
81     }
82     cout << "=====\n";
83 }
84 };
85
86 int main() {
87     QueueMahasiswa antrian;
88     int pilihan;
89     string nama, nim;
90
91     do {
92         cout << "\n=== PROGRAM ANTRIAN MAHASISWA ===\n";
93         cout << "1. Tambah Mahasiswa\n";
94         cout << "2. Hapus Mahasiswa\n";
95         cout << "3. Tampilkan Daftar Mahasiswa\n";
96         cout << "4. Keluar\n";
97         cout << "Pilihan Anda (1-4): ";
98         cin >> pilihan;
99         cin.ignore();
100
101         switch (pilihan) {
102             case 1:
103                 cout << "\nMasukkan Data Mahasiswa\n";
104                 cout << "Nama: ";
105                 getline(cin, nama);
106                 cout << "NIM : ";
107                 getline(cin, nim);
108                 antrian.tambahMahasiswa(nama, nim);
109                 break;
110             case 2:
111                 antrian.hapusMahasiswa();
112                 break;
113             case 3:
114                 antrian.tampilkanMahasiswa();
115                 break;
116             case 4:
117                 cout << "\nProgram selesai!\n";
118                 break;
119             default:
120                 cout << "\nPilihan tidak valid!\n";
121         }
122     } while (pilihan != 4);
123
124     return 0;
125 }

```

Output :

```

D:\bersama berkarya\SEMES x + v D:\bersama berkarya\SEMES x + v
=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 1

Masukkan Data Mahasiswa
Nama: Dhiya
NIM : 053

Mahasiswa pertama berhasil ditambahkan

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 1

Masukkan Data Mahasiswa
Nama: Ade
NIM : 051

Mahasiswa berhasil ditambahkan ke queue

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 1

Masukkan Data Mahasiswa
Nama: Rapli
NIM : 050

Mahasiswa berhasil ditambahkan ke queue

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 3

Pilihan Anda (1-4): 3

Daftar Mahasiswa (Total: 3 mahasiswa):
=====
1. Nama: Dhiya
   NIM : 053
2. Nama: Ade
   NIM : 051
3. Nama: Rapli
   NIM : 050
=====

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 2

Mahasiswa Dhiya (NIM: 053) berhasil dihapus

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 3

Daftar Mahasiswa (Total: 2 mahasiswa):
=====
1. Nama: Ade
   NIM : 051
2. Nama: Rapli
   NIM : 050
=====

=== PROGRAM ANTRIAN MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 4

Program selesai!

```

Deskripsi program

Untuk Penjelasan program masih sama dengan yang nomor 1, hanya saja ada perubahan dengan atribut nama mahasiswa dan NIM mahasiswa

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

JAWABAN :

Source code

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Node {
6  public:
7      string nama;
8      string nim;
9      Node* next;
10
11      Node(string namaMhs, string nimMhs) {
12          nama = namaMhs;
13          nim = nimMhs;
14          next = nullptr;
15      }
16 };
17
18 class QueuePrioritasMahasiswa {
19 private:
20     Node* depan;
21     Node* belakang;
22     int jumlahMahasiswa;
23
24 public:
25     QueuePrioritasMahasiswa() {
26         depan = nullptr;
27         belakang = nullptr;
28         jumlahMahasiswa = 0;
29     }
30
31     bool apakahKosong() {
32         return depan == nullptr;
33     }
34
35     void tambahMahasiswa(string nama, string nim) {
36         Node* nodeBaru = new Node(nama, nim);
37         jumlahMahasiswa++;
38
39         if (apakahKosong() || nim < depan->nim) {
40             nodeBaru->next = depan;
41             depan = nodeBaru;
42             if (belakang == nullptr) {
43                 belakang = nodeBaru;
44             }
45             cout << "\nMahasiswa ditambahkan di awal queue\n";
46             return;
47         }
48
49         Node* current = depan;
50         while (current->next != nullptr && current->next->nim <= nim) {
51             current = current->next;
52         }
53
54         nodeBaru->next = current->next;
55         current->next = nodeBaru;
56
57         if (current == belakang) {
58             belakang = nodeBaru;
59         }
60
61         cout << "\nMahasiswa ditambahkan ke queue sesuai prioritas NIM\n";
62     }
```



```

64 void hapusMahasiswa() {
65     if (apakahKosong()) {
66         cout << "\nQueue masih kosong!\n";
67         return;
68     }
69
70     Node* temp = depan;
71     depan = depan->next;
72     cout << "\nMahasiswa " << temp->nama << " (NIM: " << temp->nim << ") berhasil dihapus\n";
73     delete temp;
74     jumlahMahasiswa--;
75
76     if (depan == nullptr) {
77         belakang = nullptr;
78     }
79 }
80
81 void tampilkanMahasiswa() {
82     if (apakahKosong()) {
83         cout << "\nQueue masih kosong!\n";
84         return;
85     }
86
87     cout << "\nDaftar Mahasiswa Berdasarkan Prioritas NIM";
88     cout << "\n(Total: " << jumlahMahasiswa << " mahasiswa):\n";
89     cout << "=====\n";
90     Node* current = depan;
91     int nomor = 1;
92     while (current != nullptr) {
93         cout << nomor << ". Nama: " << current->nama << "\n    NIM : " << current->nim << endl;
94         current = current->next;
95         nomor++;
96     }
97     cout << "=====\n";
98 }
99 };
100
101 int main() {
102     QueuePrioritasMahasiswa antrian;
103     int pilihan;
104     string nama, nim;
105
106     do {
107         cout << "\n=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===\n";
108         cout << "1. Tambah Mahasiswa\n";
109         cout << "2. Hapus Mahasiswa\n";
110         cout << "3. Tampilkan Daftar Mahasiswa\n";
111         cout << "4. Keluar\n";
112         cout << "Pilihan Anda (1-4): ";
113         cin >> pilihan;
114         cin.ignore();
115
116         switch (pilihan) {
117             case 1:
118                 cout << "\nMasukkan Data Mahasiswa\n";
119                 cout << "Nama: ";
120                 getline(cin, nama);
121                 cout << "NIM : ";
122                 getline(cin, nim);
123                 antrian.tambahMahasiswa(nama, nim);
124                 break;
125             case 2:

```

```
126         antrian.hapusMahasiswa();  
127         break;  
128     case 3:  
129         antrian.tampilkanMahasiswa();  
130         break;  
131     case 4:  
132         cout << "\nProgram selesai!\n";  
133         break;  
134     default:  
135         cout << "\nPilihan tidak valid!\n";  
136     }  
137 } while (pilihan != 4);  
138  
139 return 0;  
140 }
```

Output



```
=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===  
1. Tambah Mahasiswa  
2. Hapus Mahasiswa  
3. Tampilkan Daftar Mahasiswa  
4. Keluar  
Pilihan Anda (1-4): 1  
  
Masukkan Data Mahasiswa  
Nama: Dhiya  
NIM : 053  
  
Mahasiswa ditambahkan di awal queue  
  
=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===  
1. Tambah Mahasiswa  
2. Hapus Mahasiswa  
3. Tampilkan Daftar Mahasiswa  
4. Keluar  
Pilihan Anda (1-4): 1  
  
Masukkan Data Mahasiswa  
Nama: Ojan  
NIM : 045  
  
Mahasiswa ditambahkan di awal queue  
  
=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===  
1. Tambah Mahasiswa  
2. Hapus Mahasiswa  
3. Tampilkan Daftar Mahasiswa  
4. Keluar  
Pilihan Anda (1-4): 1  
  
Masukkan Data Mahasiswa  
Nama: Ganesh  
NIM : 058  
  
Mahasiswa ditambahkan ke queue sesuai prioritas NIM
```

```
=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 3

Daftar Mahasiswa Berdasarkan Prioritas NIM
(Total: 3 mahasiswa):
=====
1. Nama: Ojan
   NIM : 045
2. Nama: Dhiya
   NIM : 053
3. Nama: Ganesh
   NIM : 058
=====

=== PROGRAM ANTRIAN PRIORITAS MAHASISWA ===
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Tampilkan Daftar Mahasiswa
4. Keluar
Pilihan Anda (1-4): 4

Program selesai!
```

Penjelasan Program

Program ini merupakan pengembangan dari program antrian mahasiswa sebelumnya, dengan perbedaan utama pada penerapan sistem prioritas berdasarkan NIM.

Ketika program pertama kali dijalankan, sistem membuat sebuah antrian kosong menggunakan kelas `QueuePrioritasMahasiswa`. Berbeda dengan antrian biasa yang menambahkan elemen baru di belakang, antrian prioritas ini akan menempatkan mahasiswa berdasarkan urutan NIM dari yang terkecil ke terbesar.

Proses penambahan mahasiswa (fungsi `tambahMahasiswa`) menjadi lebih kompleks karena harus mempertimbangkan prioritas. Ketika pengguna memasukkan data mahasiswa baru, sistem akan melakukan beberapa pengecekan: Pertama, jika antrian masih kosong atau NIM mahasiswa baru lebih kecil dari NIM mahasiswa di depan antrian, maka mahasiswa baru akan ditempatkan di depan antrian. Ini ditangani oleh kondisi `"if (apakahKosong() || nim < depan->nim)"`. Jika tidak memenuhi kondisi pertama, sistem akan menelusuri antrian untuk mencari posisi yang tepat. Penelusuran dilakukan dengan membandingkan NIM mahasiswa baru dengan NIM mahasiswa-mahasiswa yang sudah ada dalam antrian. Proses ini menggunakan loop `"while (current->next != nullptr && current->next->nim <= nim)"`.

Ketika menampilkan daftar mahasiswa (fungsi `tampilkanMahasiswa`), data akan ditampilkan sesuai urutan dalam antrian, yang secara otomatis sudah terurut berdasarkan NIM dari terkecil ke terbesar.

Kesimpulan

Dari yang telah saya pelajari ada dua jenis implementasi struktur data Queue yang berbeda untuk mengelola data mahasiswa.

Program pertama menggunakan antrian biasa (regular queue) yang mengikuti prinsip FIFO (First In First Out), di mana mahasiswa yang pertama masuk akan pertama dilayani, tanpa mempertimbangkan faktor lain.

Program kedua memperkenalkan konsep yang lebih kompleks yaitu antrian prioritas (priority queue) berdasarkan NIM. Perbedaan utamanya terletak pada logika penempatan data, di mana sistem secara otomatis mengurutkan mahasiswa berdasarkan NIM dari yang terkecil ke terbesar. Ini menunjukkan bahwa struktur data yang sama (Queue) dapat dimodifikasi untuk memenuhi kebutuhan yang berbeda.