

LAPORAN PRAKTIKUM

MODUL 8

QUEUE



Nama :

Candra Dinata (2311104061)

Kelas :

S1SE 07 02

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Tujuan pembelajaran dari praktikum queue dalam C++ adalah untuk memahami konsep dasar struktur data queue sebagai salah satu implementasi dari antrian FIFO (First In, First Out), serta mampu mengimplementasikannya menggunakan bahasa pemrograman C++. Mahasiswa diharapkan dapat mengenali operasi-operasi dasar seperti enqueue (menambahkan elemen ke belakang antrian), dequeue (menghapus elemen dari depan antrian), peek (melihat elemen di depan antrian tanpa menghapusnya), serta memahami cara menangani kondisi antrian penuh atau kosong. Selain itu, praktikum ini bertujuan untuk melatih kemampuan mahasiswa dalam menggunakan pustaka standar C++ seperti std::queue, membandingkan implementasi menggunakan array dan linked list, serta mengaplikasikan queue dalam kasus-kasus nyata seperti simulasi antrian, penjadwalan proses, atau sistem pemrosesan data. Dengan demikian, mahasiswa dapat memahami pentingnya struktur data queue dalam pengelolaan data yang efisien dan penggunaannya dalam pengembangan perangkat lunak.

II. DASAR TEORI

Queue adalah salah satu struktur data linier yang bekerja berdasarkan prinsip First In, First Out (FIFO), di mana elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan. Struktur data ini sering dianalogikan seperti antrian di dunia nyata, misalnya antrian di loket tiket, di mana orang yang datang lebih dulu akan dilayani lebih dulu. Dalam C++, queue

dapat diimplementasikan secara manual menggunakan array atau linked list, maupun dengan memanfaatkan pustaka standar `std::queue` yang disediakan oleh STL (Standard Template Library). Operasi dasar pada queue meliputi enqueue (menambahkan elemen di belakang antrian), dequeue (menghapus elemen dari depan antrian), peek atau front (mengakses elemen terdepan tanpa menghapusnya), serta fungsi untuk memeriksa apakah queue kosong (`isEmpty`) atau penuh (`isFull`, jika diimplementasikan dengan array berukuran tetap). Queue memiliki banyak aplikasi dalam pengembangan perangkat lunak, seperti penjadwalan proses, sistem antrean, simulasi, atau pengelolaan buffer data, menjadikannya salah satu struktur data yang penting untuk dipahami dalam pemrograman.

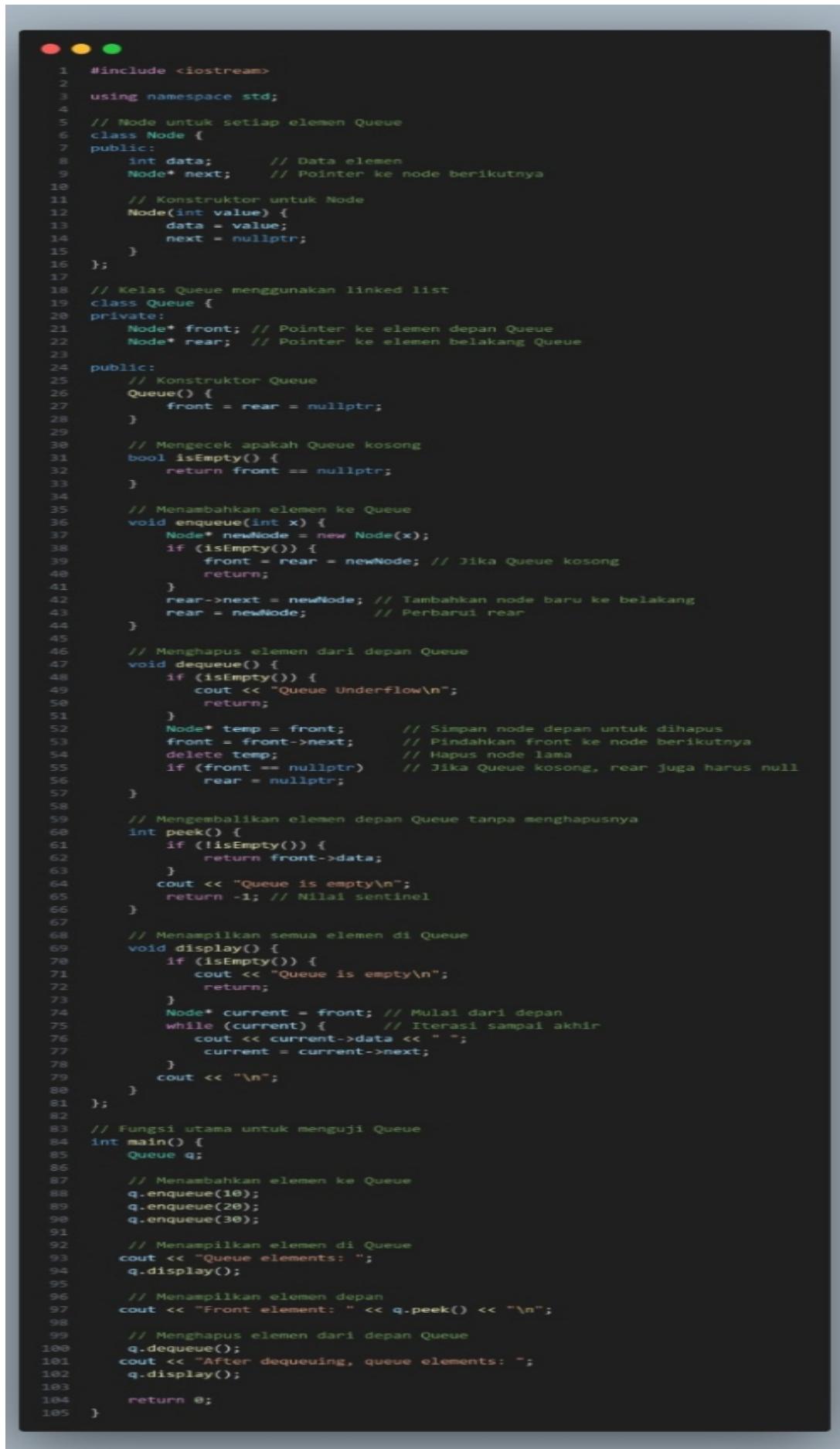
III. GUIDED

1.

```
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
22     bool isEmpty() {
23         return front == -1 || front > rear;
24     }
25
26
27     void enqueue(int x) {
28         if (isFull()) {
29             cout << "Queue Overflow\n";
30             return;
31         }
32         if (front == -1) front = 0;
33         arr[++rear] = x;
34     }
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
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 int main() {
68     Queue q;
69
70     q.enqueue(10);
71     q.enqueue(20);
72     q.enqueue(30);
73
74     cout << "Queue elements: ";
75     q.display();
76
77     cout << "Front element: " << q.peek() << "\n";
78
79     cout << "After dequeuing, queue elements: ";
80     q.display();
81
82     return 0;
83 }
84 }
```

Kode di atas merupakan implementasi struktur data queue menggunakan array statis dengan ukuran maksimum 100 elemen. Kelas Queue memiliki atribut front dan rear untuk menunjukkan indeks awal dan akhir antrian, serta array arr untuk menyimpan elemen-elemen antrian. Konstruktor Queue menginisialisasi front dan rear ke nilai -1 untuk menandakan bahwa antrian kosong. Fungsi isFull memeriksa apakah antrian penuh dengan membandingkan nilai rear dengan batas maksimum array, sementara isEmpty memeriksa apakah antrian kosong dengan mengevaluasi kondisi front dan rear. Fungsi enqueue menambahkan elemen baru ke belakang antrian, mengatur front ke 0 jika elemen pertama dimasukkan, dan meningkatkan nilai rear. Fungsi dequeue menghapus elemen dari depan antrian dengan menaikkan nilai front. Fungsi peek mengembalikan elemen terdepan tanpa menghapusnya, sementara display mencetak semua elemen antrian dari indeks front ke rear. Pada fungsi main, objek Queue dibuat, kemudian elemen 10, 20, dan 30 ditambahkan ke antrian menggunakan enqueue. Program mencetak elemen-elemen antrian, menampilkan elemen terdepan menggunakan peek, lalu mencoba menampilkan isi antrian setelah proses penghapusan elemen dilakukan dengan dequeue. Namun, terdapat kesalahan kecil dalam kode karena elemen antrian tidak ditampilkan kembali setelah dequeue, sehingga hasil akhir tidak sesuai.

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     }
58
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 }
```

Kode di atas adalah implementasi struktur data queue menggunakan linked list, yang memungkinkan penambahan dan penghapusan elemen secara dinamis. Kelas Node mendefinisikan elemen dari antrian, dengan setiap node memiliki dua atribut: data yang menyimpan nilai elemen, dan next yang menunjuk ke node berikutnya. Kelas Queue menggunakan dua pointer, front dan rear, untuk menunjuk ke elemen depan dan belakang antrian. Dalam konstruktor Queue, kedua pointer ini diinisialisasi ke nullptr untuk menunjukkan bahwa antrian kosong. Fungsi isEmpty memeriksa apakah antrian kosong dengan memeriksa apakah front adalah nullptr. Fungsi enqueue menambahkan elemen ke belakang antrian dengan membuat node baru dan menghubungkannya ke node belakang yang ada, serta memperbarui rear untuk menunjuk ke node baru. Fungsi dequeue menghapus elemen depan dengan memindahkan pointer front ke node berikutnya dan menghapus node lama. Jika antrian kosong setelah penghapusan, rear diatur menjadi nullptr. Fungsi peek mengembalikan elemen depan tanpa menghapusnya, dan fungsi display mencetak semua elemen dari depan hingga belakang antrian dengan melakukan iterasi melalui linked list. Pada fungsi main, program menguji operasi antrian dengan menambahkan elemen-elemen 10, 20, dan 30, menampilkan elemen-elemen antrian, menampilkan elemen depan menggunakan peek, kemudian menghapus elemen depan dan menampilkan kembali elemen yang tersisa di antrian. Kode ini juga mencakup penanganan kondisi kosong dengan mencetak pesan yang sesuai jika antrian kosong saat mencoba melakukan operasi tertentu.

3.



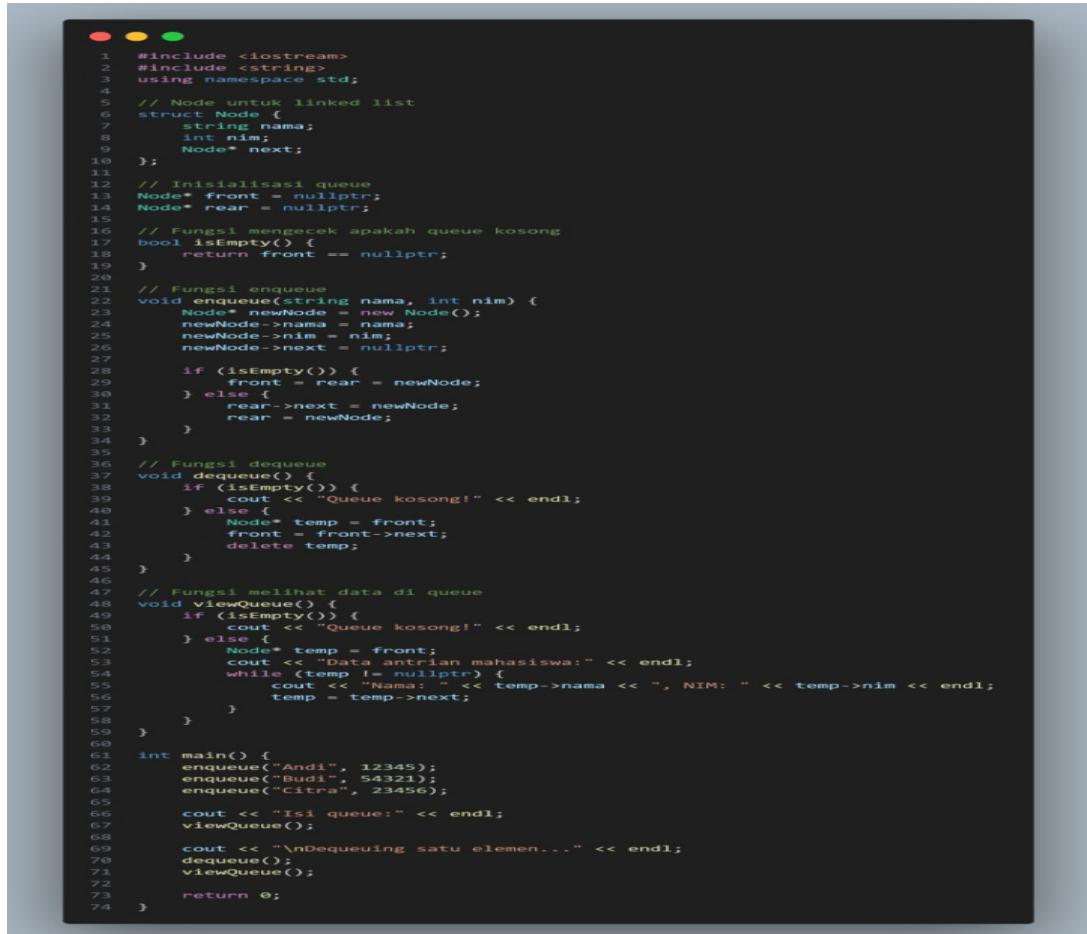
```
1 #include<iostream>
2
3 using namespace std;
4
5 const int maksimalQueue = 5; // Maksimal antrian
6 int front = 0; // Penanda antrian
7 int back = 0; // Penanda
8 string queueTeller[5]; // Fungsi pengecekan
9
10 bool isFull() { // Pengecekan antrian penuh atau tidak
11     if (back == maksimalQueue) { 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 { // Antriannya ada isi queueTeller[back] = data; back++;
32             }
33         }
34     }
35
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         }
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 }
```

Kode di atas mengimplementasikan sistem antrian menggunakan array dengan ukuran maksimal 5 elemen, di mana setiap elemen adalah string yang mewakili nama orang dalam antrian. Program ini memiliki beberapa fungsi utama, yaitu isFull() dan isEmpty() yang memeriksa apakah antrian sudah penuh atau kosong berdasarkan nilai penanda back (indeks elemen terakhir yang terisi). Fungsi enqueueAntrian(string data) digunakan untuk

menambahkan elemen ke dalam antrian, dengan pengecekan apakah antrian penuh atau kosong. Jika antrian penuh, akan mencetak pesan "Antrian penuh"; jika kosong, elemen pertama ditambahkan, dan untuk antrian yang tidak kosong, elemen baru ditambahkan di posisi yang sesuai. Fungsi dequeueAntrian() menghapus elemen pertama dari antrian dan menggeser elemen-elemen berikutnya untuk mengisi posisi kosong tersebut, serta mengurangi nilai back. Fungsi countQueue() mengembalikan jumlah elemen yang ada dalam antrian, sedangkan clearQueue() menghapus semua elemen antrian dan mengatur penanda front dan back kembali ke 0. Fungsi viewQueue() menampilkan seluruh isi antrian, memeriksa setiap posisi apakah kosong atau terisi dengan data. Dalam fungsi main(), beberapa operasi dilakukan untuk menambah elemen ("Andi" dan "Maya"), menampilkan antrian, menghitung jumlah antrian, menghapus elemen pertama, dan akhirnya mengosongkan seluruh antrian, menampilkan hasil setiap langkah. Kode ini secara keseluruhan mendemonstrasikan cara kerja antrian dengan array dalam program sederhana.

IV. UNGUIDED

1.



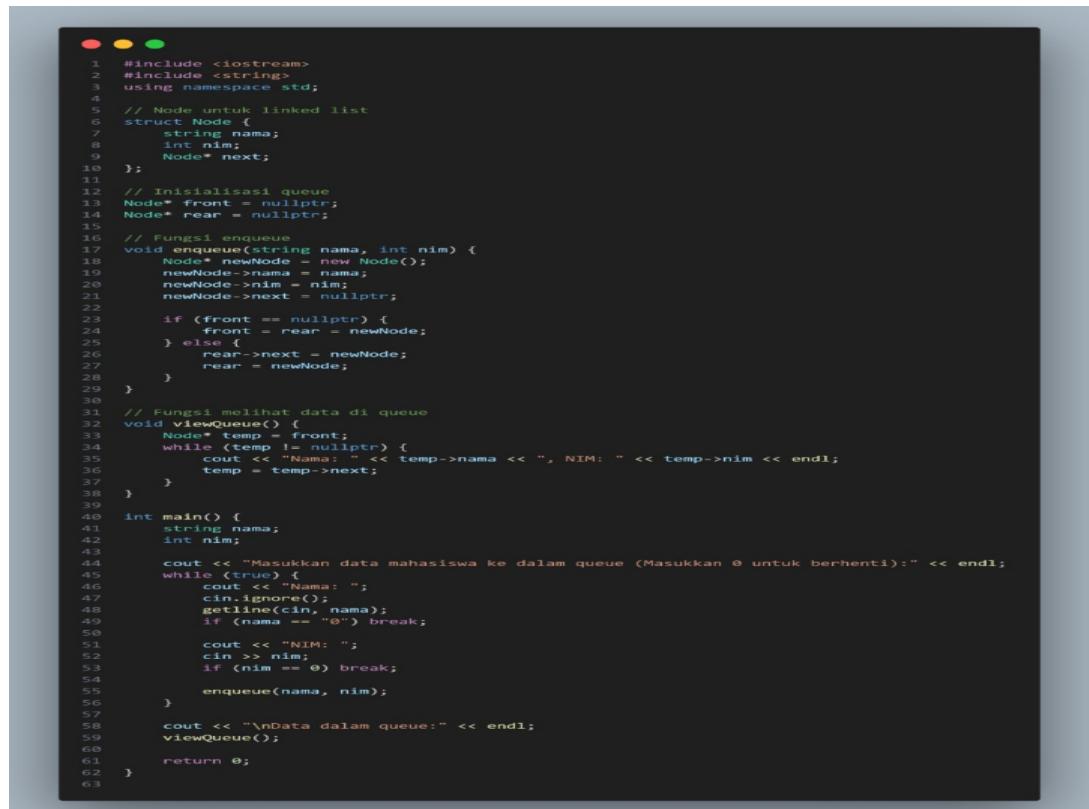
```
1 //include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node untuk linked list
6 struct Node {
7     string nama;
8     int nim;
9     Node* next;
10 };
11
12 // Inisialisasi queue
13 Node* front = nullptr;
14 Node* rear = nullptr;
15
16 // Fungsi mengecek apakah queue kosong
17 bool isEmpty() {
18     return front == nullptr;
19 }
20
21 // Fungsi enqueue
22 void enqueue(string nama, int nim) {
23     Node* newNode = new Node();
24     newNode->nama = nama;
25     newNode->nim = nim;
26     newNode->next = nullptr;
27
28     if (isEmpty()) {
29         front = rear = newNode;
30     } else {
31         rear->next = newNode;
32         rear = newNode;
33     }
34 }
35
36 // Fungsi dequeue
37 void dequeue() {
38     if (isEmpty()) {
39         cout << "Queue kosong!" << endl;
40     } else {
41         Node* temp = front;
42         front = front->next;
43         delete temp;
44     }
45 }
46
47 // Fungsi melihat data di queue
48 void viewQueue() {
49     if (isEmpty()) {
50         cout << "Queue kosong!" << endl;
51     } else {
52         Node* temp = front;
53         cout << "Data antrian mahasiswa:" << endl;
54         while (temp != nullptr) {
55             cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
56             temp = temp->next;
57         }
58     }
59 }
60
61 int main() {
62     enqueue("Andi", 12345);
63     enqueue("Budi", 54321);
64     enqueue("Citra", 23456);
65
66     cout << "Isi queue:" << endl;
67     viewQueue();
68
69     cout << "\nDequeueing satu elemen..." << endl;
70     dequeue();
71     viewQueue();
72
73     return 0;
74 }
```

```
PS C:\Users\Candra Dinata\pertemuan1> cd 'C:\Users\Candra Dinata\pertemuan1\stdmod8\output' & .\unguided1.exe'
Isi queue:
Data antrian mahasiswa:
Nama: Andi, NIM: 12345
Nama: Budi, NIM: 54321
Nama: Citra, NIM: 23456

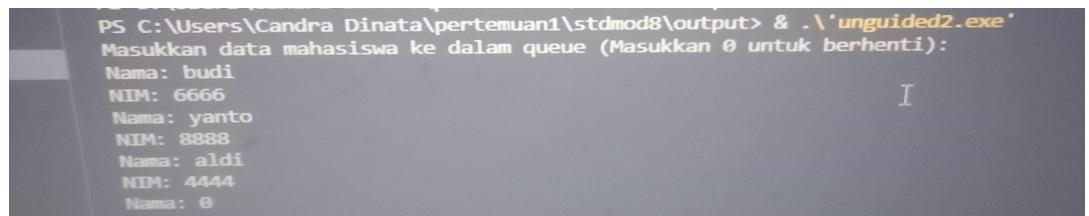
Dequeuing satu elemen...
Data antrian mahasiswa:
Nama: Budi, NIM: 54321
Nama: Citra, NIM: 23456
PS C:\Users\Candra Dinata\pertemuan1\stdmod8\output>
```

Kode ini mengimplementasikan struktur data queue menggunakan linked list untuk menyimpan data mahasiswa, yang terdiri dari nama dan NIM. Setiap elemen queue diwakili oleh sebuah node, yang memiliki tiga atribut: nama (string), nim (integer), dan next (pointer ke node berikutnya). Fungsi isEmpty() digunakan untuk memeriksa apakah antrian kosong dengan mengecek apakah pointer front adalah nullptr. Fungsi enqueue() menambahkan elemen baru ke belakang antrian dengan membuat node baru, mengisinya dengan data mahasiswa, dan menyambungkannya ke node terakhir yang ada; jika antrian kosong, node baru akan menjadi elemen pertama yang menunjuk ke front dan rear. Fungsi dequeue() menghapus elemen pertama dari antrian dengan memindahkan pointer front ke node berikutnya dan menghapus node yang lama. Fungsi viewQueue() digunakan untuk menampilkan seluruh data antrian, dengan cara melintasi setiap node dari front hingga rear, mencetak nama dan NIM setiap mahasiswa. Pada fungsi main(), beberapa operasi dilakukan untuk menambah elemen ke antrian (nama dan NIM mahasiswa "Andi", "Budi", dan "Citra"), menampilkan antrian yang ada, kemudian menghapus elemen pertama menggunakan dequeue(), dan akhirnya menampilkan hasil antrian setelah penghapusan. Dengan demikian, kode ini mendemonstrasikan penggunaan linked list untuk mengelola antrian dinamis dalam aplikasi sederhana.

2.



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node untuk linked list
6 struct Node {
7     string nama;
8     int nim;
9     Node* next;
10 };
11
12 // Inisialisasi queue
13 Node* front = nullptr;
14 Node* rear = nullptr;
15
16 // Fungsi enqueue
17 void enqueue(string nama, int nim) {
18     Node* newNode = new Node();
19     newNode->nama = nama;
20     newNode->nim = nim;
21     newNode->next = nullptr;
22
23     if (front == nullptr) {
24         front = rear = newNode;
25     } else {
26         rear->next = newNode;
27         rear = newNode;
28     }
29 }
30
31 // Fungsi melihat data di queue
32 void viewQueue() {
33     Node* temp = front;
34     while (temp != nullptr) {
35         cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
36         temp = temp->next;
37     }
38 }
39
40 int main() {
41     string nama;
42     int nim;
43
44     cout << "Masukkan data mahasiswa ke dalam queue (Masukkan 0 untuk berhenti):" << endl;
45     while (true) {
46         cout << "Nama: ";
47         cin.ignore();
48         getline(cin, nama);
49         if (nama == "0") break;
50
51         cout << "NIM: ";
52         cin >> nim;
53         if (nim == 0) break;
54
55         enqueue(nama, nim);
56     }
57
58     cout << "\nData dalam queue:" << endl;
59     viewQueue();
60
61     return 0;
62 }
```

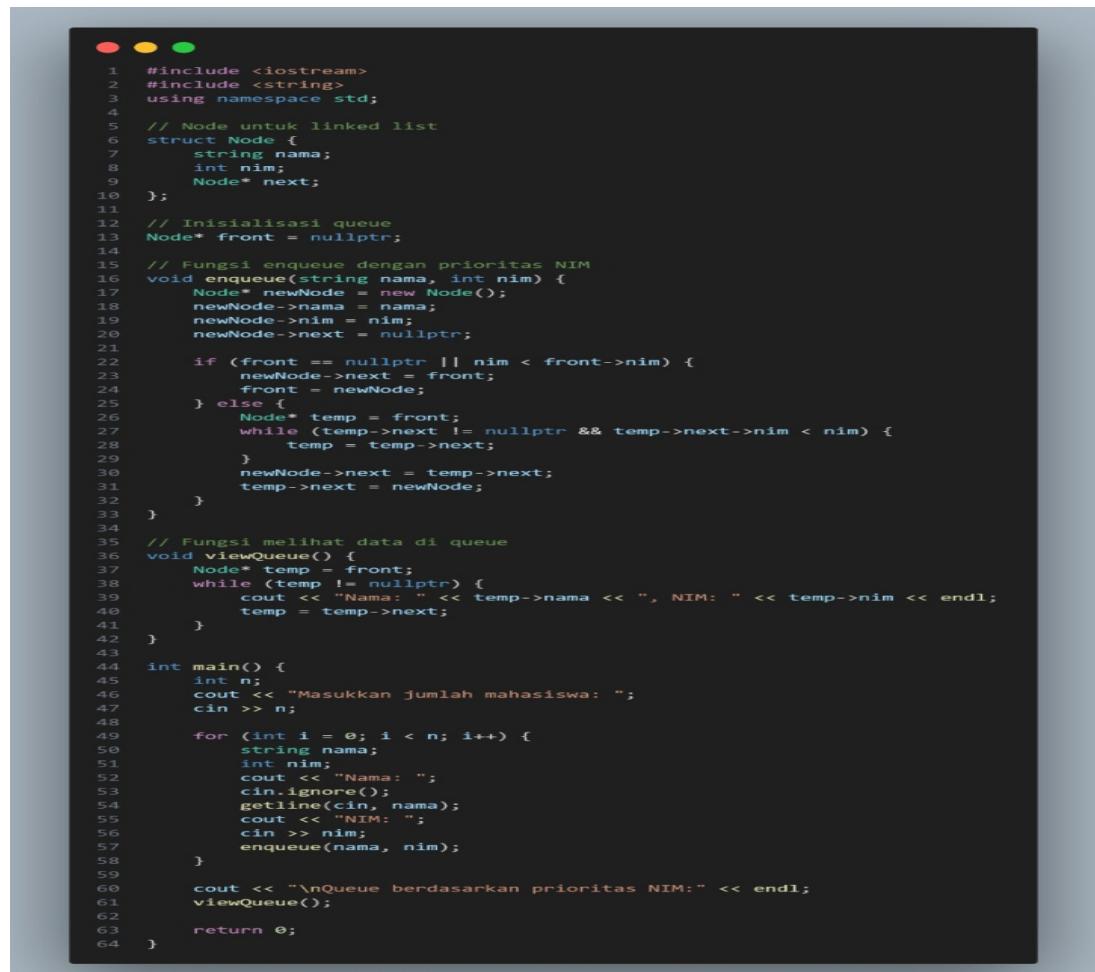


```
PS C:\Users\Candra Dinata\pertemuan1\stdmod8\output> & .\unguided2.exe
Masukkan data mahasiswa ke dalam queue (Masukkan 0 untuk berhenti):
Nama: budi
NIM: 6666
Nama: yanto
NIM: 8888
Nama: aldi
NIM: 4444
Nama: 0
```

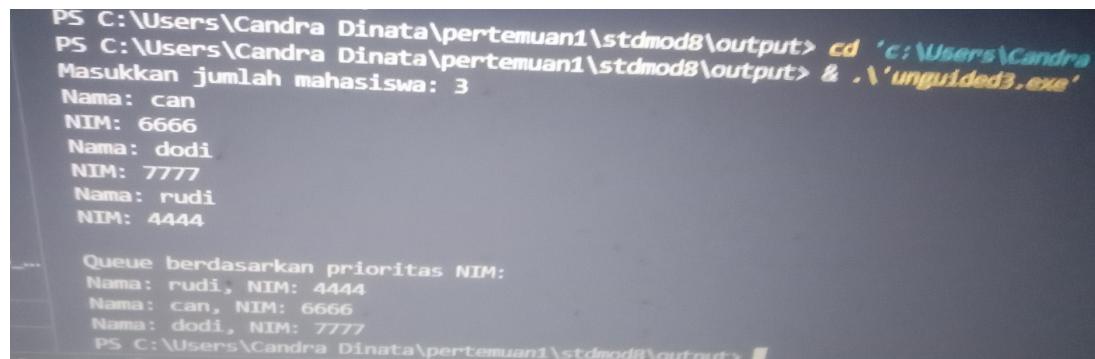
Kode ini mengimplementasikan sebuah queue menggunakan linked list untuk menyimpan data mahasiswa berupa nama dan NIM. Struktur data Node memiliki tiga atribut: nama (string), nim (integer), dan next (pointer ke node berikutnya). Dalam fungsi enqueue(), elemen baru yang berisi data mahasiswa akan ditambahkan ke antrian di bagian belakang, dengan memeriksa apakah antrian kosong (yaitu front adalah nullptr), jika kosong, node baru menjadi elemen pertama yang menunjuk ke front dan rear, sedangkan jika sudah ada elemen, node baru ditambahkan setelah rear dan rear diperbarui. Pada fungsi viewQueue(), seluruh data dalam antrian ditampilkan dengan cara mengiterasi setiap node dari front hingga rear dan mencetak nama dan NIM mahasiswa. Dalam fungsi main(), program meminta pengguna untuk memasukkan nama dan NIM mahasiswa secara berulang untuk dimasukkan ke dalam antrian. Pengguna dapat berhenti dengan memasukkan 0 sebagai input untuk nama atau NIM. Setelah semua data dimasukkan, program akan menampilkan seluruh isi antrian. Dengan demikian, kode ini menyediakan cara

untuk menyimpan dan menampilkan data mahasiswa dalam struktur antrian berbasis linked list secara dinamis.

3.



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node untuk linked list
6 struct Node {
7     string nama;
8     int nim;
9     Node* next;
10 };
11
12 // Inisialisasi queue
13 Node* front = nullptr;
14
15 // Fungsi enqueue dengan prioritas NIM
16 void enqueue(string nama, int nim) {
17     Node* newNode = new Node();
18     newNode->nama = nama;
19     newNode->nim = nim;
20     newNode->next = nullptr;
21
22     if (front == nullptr || nim < front->nim) {
23         newNode->next = front;
24         front = newNode;
25     } else {
26         Node* temp = front;
27         while (temp->next != nullptr && temp->next->nim < nim) {
28             temp = temp->next;
29         }
30         newNode->next = temp->next;
31         temp->next = newNode;
32     }
33 }
34
35 // Fungsi melihat data di queue
36 void viewQueue() {
37     Node* temp = front;
38     while (temp != nullptr) {
39         cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << endl;
40         temp = temp->next;
41     }
42 }
43
44 int main() {
45     int n;
46     cout << "Masukkan jumlah mahasiswa: ";
47     cin >> n;
48
49     for (int i = 0; i < n; i++) {
50         string nama;
51         int nim;
52         cout << "Nama: ";
53         cin.ignore();
54         getline(cin, nama);
55         cout << "NIM: ";
56         cin >> nim;
57         enqueue(nama, nim);
58     }
59
60     cout << "\nQueue berdasarkan prioritas NIM:" << endl;
61     viewQueue();
62
63     return 0;
64 }
```



```
PS C:\Users\Candra Dinata\pertemuan1\stdmod8\output> cd 'c:\Users\Candra
PS C:\Users\Candra Dinata\pertemuan1\stdmod8\output> & .\'unguided3.exe'
Masukkan jumlah mahasiswa: 3
Nama: can
NIM: 6666
Nama: dodi
NIM: 7777
Nama: rudi
NIM: 4444

Queue berdasarkan prioritas NIM:
Nama: rudi, NIM: 4444
Nama: can, NIM: 6666
Nama: dodi, NIM: 7777
PS C:\Users\Candra Dinata\pertemuan1\stdmod8\output>
```

Kode ini mengimplementasikan sebuah antrian dengan prioritas menggunakan linked list, di mana data mahasiswa (nama dan NIM) dimasukkan ke dalam antrian berdasarkan prioritas NIM yang lebih kecil. Struktur data Node memiliki tiga atribut: nama (string), nim (integer), dan next (pointer ke node berikutnya). Fungsi enqueue() menambahkan elemen baru ke antrian dengan cara memeriksa apakah antrian kosong atau apakah NIM elemen baru lebih kecil dari NIM elemen pertama (front). Jika ya, elemen

baru akan menjadi elemen pertama; jika tidak, elemen baru akan disisipkan di posisi yang sesuai dalam antrian, berdasarkan urutan NIM yang lebih kecil. Proses ini dilakukan dengan mengiterasi melalui antrian hingga menemukan posisi yang tepat di mana NIM elemen baru lebih besar daripada NIM elemen berikutnya. Fungsi `viewQueue()` digunakan untuk menampilkan seluruh isi antrian, dengan cara mengiterasi setiap node dari front hingga akhir, mencetak nama dan NIM setiap mahasiswa. Di dalam fungsi `main()`, program meminta pengguna untuk memasukkan jumlah mahasiswa yang akan dimasukkan ke dalam antrian, kemudian untuk setiap mahasiswa, nama dan NIM dimasukkan dan disisipkan ke dalam antrian menggunakan fungsi `enqueue()`. Setelah seluruh data dimasukkan, antrian ditampilkan berdasarkan prioritas NIM terkecil. Dengan demikian, kode ini mendemonstrasikan penerapan antrian prioritas dengan linked list di mana elemen-elemen baru disisipkan sesuai dengan urutan NIM yang lebih kecil.

V. KESIMPULAN

Kesimpulan dari praktikum tentang struktur data Queue menggunakan C++ menunjukkan bahwa Queue merupakan struktur data yang sangat berguna dalam memanajemen data secara terstruktur dengan prinsip antrian, di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar (FIFO). Praktikum ini memberikan pemahaman tentang bagaimana Queue dapat diimplementasikan dengan berbagai cara, seperti menggunakan array statis, linked list, dan dengan tambahan fitur seperti prioritas dalam penambahan elemen. Melalui pengujian berbagai operasi seperti `enqueue`, `dequeue`, dan pengecekan kondisi Queue (kosong atau penuh), peserta praktikum dapat memahami cara kerja dasar Queue dalam berbagai kasus. Selain itu, praktikum ini juga memperlihatkan bagaimana antrian dapat diatur secara dinamis dalam memproses data, baik untuk antrian biasa maupun antrian dengan prioritas. Implementasi Queue di C++ ini mengajarkan keterampilan penting dalam pengelolaan memori dan pengorganisasian data dalam berbagai aplikasi, baik itu dalam pengelolaan tugas, proses, atau antrian pada sistem komputer.

