

LAPORAN PRAKTIKUM
Modul 8
QUEUE



Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
S1SE-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 adalah struktur data yang menyimpan elemen dalam urutan linier dengan prinsip “First-In-First-Out” (FIFO), di mana elemen yang pertama kali dimasukkan adalah yang pertama kali dikeluarkan. Queue sering digunakan dalam berbagai aplikasi, seperti manajemen antrian dalam sistem operasi dan pengelolaan tugas dalam pemrograman.

Queue adalah struktur data yang sangat penting dalam pemrograman dan pengolahan data. Memahami cara kerja dan penerapannya dapat membantu dalam mengelola alur data secara efisien dalam berbagai aplikasi teknologi.

Melalui dua operasi dasar, yaitu enqueue dan dequeue, queue memungkinkan penambahan dan penghapusan elemen secara teratur, yang mendukung pengolahan data yang efisien. Implementasi queue dapat dilakukan menggunakan struktur data seperti array atau linked list, masing-masing dengan kelebihan dan kekurangan tersendiri. Queue dengan array menawarkan akses cepat tetapi memiliki ukuran tetap, sedangkan queue dengan linked list lebih fleksibel dalam hal penambahan elemen, meskipun memerlukan lebih banyak memori untuk menyimpan penunjuk.

Secara keseluruhan, queue adalah alat yang sangat penting dalam pengembangan perangkat lunak dan sistem informasi, dan pemahaman yang baik tentangnya akan sangat bermanfaat dalam praktik pemrograman dan pengelolaan data.

3. Guided

Guided 1

Outputnya:

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

Code programnya

```
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 }
```

Guided 2

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3>
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
```

Code programnya

```

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 }

```

Guided 3

Code programnya

```

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         }
66     }
67     endl;
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 }

```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3>
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
```

4. Unguided

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

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING>
1. Andi
2. Maya
Jumlah antrian = 2
1. Maya
Jumlah antrian = 1
Queue kosong
Jumlah antrian = 0
```

Code programnya

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class QueueLinkedList {
6 private:
7     struct Node {
8         string data;
9         Node* next;
10        Node(string val) : data(val), next(nullptr) {}
11    };
12    Node *front, *rear;
13    int size;
14
15 public:
16    QueueLinkedList() : front(nullptr), rear(nullptr), size(0) {}
17
18    void enqueue(string data) {
19        Node* newNode = new Node(data);
20        if (isEmpty()) {
21            front = rear = newNode;
22        } else {
23            rear->next = newNode;
24            rear = newNode;
25        }
26        size++;
27    }
28
29    void dequeue() {
30        if (isEmpty()) {
31            cout << "Queue kosong" << endl;
32            return;
33        }
34        Node* temp = front;
35        front = front->next;
36        delete temp;
37        size--;
38        if (front == nullptr) rear = nullptr;
39    }
40
41    void viewQueue() {
42        if (isEmpty()) {
43            cout << "Queue kosong" << endl;
44            return;
45        }
46        Node* current = front;
47        int i = 1;
48        while (current) {
49            cout << i++ << ". " << current->data << endl;
50            current = current->next;
51        }
52    }
53
54    bool isEmpty() { return front == nullptr; }
55    int getSize() { return size; }
56
57    void clearQueue() {
58        while (!isEmpty()) {
59            dequeue();
60        }
61    }
62 };
63
64 int main() {
65     QueueLinkedList queue;
66     queue.enqueue("Andi");
67     queue.enqueue("Maya");
68     queue.viewQueue();
69     cout << "Jumlah antrian = " << queue.getSize() << endl;
70     queue.dequeue();
71     queue.viewQueue();
72     cout << "Jumlah antrian = " << queue.getSize() << endl;
73     queue.clearQueue();
74     queue.viewQueue();
75     cout << "Jumlah antrian = " << queue.getSize() << endl;
76     return 0;
77 }
```

2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIMMahasiswa
Code programnya

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class StudentQueue {
6 private:
7     struct Student {
8         string nama;
9         string nim;
10        Student* next;
11        Student(string n, string id) : nama(n), nim(id), next(nullptr) {}
12    };
13    Student* front;
14    Student* rear;
15
16 public:
17    StudentQueue() : front(nullptr), rear(nullptr) {}
18
19    void enqueue() {
20        string nama, nim;
21        cout << "Masukkan nama mahasiswa: ";
22        cin.ignore(); // Membersihkan buffer sebelum getline
23        getline(cin, nama);
24        cout << "Masukkan NIM mahasiswa: ";
25        cin >> nim;
26
27        Student* newStudent = new Student(nama, nim);
28        if (isEmpty()) {
29            front = rear = newStudent;
30        } else {
31            rear->next = newStudent;
32            rear = newStudent;
33        }
34        cout << "Mahasiswa " << nama << " ditambahkan ke antrian." << endl;
35    }
36
37    void dequeue() {
38        if (isEmpty()) {
39            cout << "Antrian kosong!" << endl;
40            return;
41        }
42        Student* temp = front;
43        front = front->next;
44        cout << "Mahasiswa " << temp->nama << " dikeluarkan dari antrian." << endl;
45        delete temp;
46        if (front == nullptr) rear = nullptr;
47    }
48
49    void viewQueue() {
50        if (isEmpty()) {
51            cout << "Antrian kosong!" << endl;
52            return;
53        }
54        Student* current = front;
55        int i = 1;
56        while (current) {
57            cout << i++ << ". Nama: " << current->nama
58                << ", NIM: " << current->nim << endl;
59            current = current->next;
60        }
61    }
62
63    bool isEmpty() {
64        return front == nullptr;
65    }
66 };
67
68 int main() {
69     StudentQueue queue;
70     int pilihan;
71     do {
72         cout << "Menu Antrian Mahasiswa:" << endl;
73         cout << "1. Tambah Mahasiswa" << endl;
74         cout << "2. Hapus Mahasiswa" << endl;
75         cout << "3. Lihat Antrian" << endl;
76         cout << "4. Keluar" << endl;
77         cout << "Pilih menu: ";
78         cin >> pilihan;
79
80         switch(pilihan) {
81             case 1: queue.enqueue(); break;
82             case 2: queue.dequeue(); break;
83             case 3: queue.viewQueue(); break;
84             case 4: cout << "Keluar dari program." << endl; break;
85             default: cout << "Pilihan tidak valid!" << endl; break;
86         }
87     } while (pilihan != 4);
88
89     return 0;
90 }
```

Outputnya:

PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3\

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 1

Masukkan nama mahasiswa: Andi

Masukkan NIM mahasiswa: 2311104054

Mahasiswa Andi ditambahkan ke antrian.

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 1

Masukkan nama mahasiswa: Maya

Masukkan NIM mahasiswa: 2311104065

Mahasiswa Maya ditambahkan ke antrian.

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 3

1. Nama: Andi, NIM: 2311104054

2. Nama: Maya, NIM: 2311104065

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 2

Mahasiswa Andi dikeluarkan dari antrian.

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 2

Mahasiswa Maya dikeluarkan dari antrian.

Menu Antrian Mahasiswa:

1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar

Pilih menu: 4

Keluar dari program.

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

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class PrioritizedStudentQueue {
6 private:
7     struct Student {
8         string nama;
9         string nim;
10        Student* next;
11    };
12    Student* front;
13    Student* rear;
14
15 public:
16    PrioritizedStudentQueue() : front(nullptr), rear(nullptr) {}
17
18    void enqueue() {
19        string nama, nim;
20        cout << "Masukkan nama mahasiswa: ";
21        cin.ignore(); // Membersihkan buffer sebelum getline
22        getline(cin, nama);
23        cout << "Masukkan NIM mahasiswa: ";
24        cin >> nim;
25
26        Student* newStudent = new Student(nama, nim);
27
28        // Queue kosong
29        if (isEmpty()) {
30            front = rear = newStudent;
31            return;
32        }
33
34        // NIM lebih kecil dari front, masukkan di depan
35        if (nim < front->nim) {
36            newStudent->next = front;
37            front = newStudent;
38            return;
39        }
40
41        // Cari posisi yang tepat berdasarkan NIM
42        Student* current = front;
43        while (current->next && current->next->nim < nim) {
44            current = current->next;
45        }
46
47        newStudent->next = current->next;
48        current->next = newStudent;
49
50        // Update rear jika diperlukan
51        if (newStudent->next == nullptr) {
52            rear = newStudent;
53        }
54    }
55
56    void dequeue() {
57        if (isEmpty()) {
58            cout << "Antrian kosong!" << endl;
59            return;
60        }
61        Student* temp = front;
62        front = front->next;
63        cout << "Mahasiswa " << temp->nama << " dikeluarkan dari antrian." << endl;
64        delete temp;
65        if (front == nullptr) rear = nullptr;
66    }
67
68    void viewQueue() {
69        if (isEmpty()) {
70            cout << "Antrian kosong!" << endl;
71            return;
72        }
73        Student* current = front;
74        int i = 1;
75        while (current) {
76            cout << i++ << ". Nama: " << current->nama
77                << ", NIM: " << current->nim << endl;
78            current = current->next;
79        }
80    }
81
82    bool isEmpty() { return front == nullptr; }
83 };
84
85 int main() {
86     PrioritizedStudentQueue queue;
87     int pilihan;
88     do {
89         cout << "Menu Antrian Mahasiswa Prioritas:" << endl;
90         cout << "1. Tambah Mahasiswa" << endl;
91         cout << "2. Hapus Mahasiswa" << endl;
92         cout << "3. Lihat Antrian" << endl;
93         cout << "4. Keluar" << endl;
94         cout << "Pilih menu: ";
95         cin >> pilihan;
96
97         switch(pilihan) {
98             case 1: queue.enqueue(); break;
99             case 2: queue.dequeue(); break;
100            case 3: queue.viewQueue(); break;
101            case 4: cout << "Keluar dari program." << endl; break;
102            default: cout << "Pilihan tidak valid!" << endl; break;
103        }
104    } while (pilihan != 4);
105
106    return 0;
107 }

```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3>
Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Andi
Masukkan NIM mahasiswa: 2311104054

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 1
Masukkan nama mahasiswa: Maya
Masukkan NIM mahasiswa: 2311104065

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 3
1. Nama: Andi, NIM: 2311104054
2. Nama: Maya, NIM: 2311104065

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Andi dikeluarkan dari antrian.

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 2
Mahasiswa Maya dikeluarkan dari antrian.

Menu Antrian Mahasiswa Prioritas:
1. Tambah Mahasiswa
2. Hapus Mahasiswa
3. Lihat Antrian
4. Keluar
Pilih menu: 4
Keluar dari program.
```

5. Kesimpulan

Pada praktikum ini, kita mempelajari tentang struktur data Queue, termasuk konsep dasar, karakteristik, serta cara kerjanya. Selain itu, kita juga akan memahami bagaimana menerapkan Queue dalam berbagai kasus nyata, seperti sistem antrian, manajemen proses dalam sistem operasi, atau penanganan data secara terorganisir. Dengan memahami materi ini, kita diharapkan mampu mengimplementasikan Queue secara efektif dalam program dan memanfaatkan keunggulannya untuk memecahkan masalah secara efisien.