# LAAPORAN PRAKTIKUK STRUKTUR DATA DAN PEMROGRAMAN

# MODUL VII

# QUEUE


**Disusun Oleh :**

Ahmadan Syaridin

2311102038

IF-11-A

**Dosen Pengampu :**

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



**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**PURWOKERTO**

**2024**

# BAB I

## A. TUJUAN PRAKTIKUM

a. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
b. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
c. Mahasiswa mampu menerapkan operasi tampil data pada queue

# BAB II

## B. DASAR TEORI

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (FirstIn 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 seharihari, 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 terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir diinputkan akan berada paling dengan dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.
Pada Queue, operasi tersebut dilakukan ditempat berbeda (melalui salah satu ujung) karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut **Enqueue** dan **Dequeue** pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

### 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

# BAB III

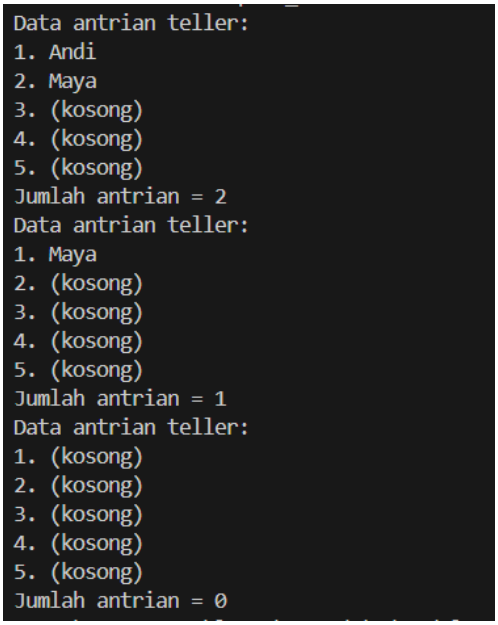
## C. GUIDED

### SOURCE CODE

```
#include <iostream>

using namespace std;

const int maksimalQueue = 5;  // Maksimal antrian
int front = 0;                // Penanda antrian
int back = 0;                 // Penanda
string queueTeller[5];        // Fungsi pengecekan

bool isFull() {
    // Pengecekan antrian penuh atau tidak
  if (back == maksimalQueue) {
    return true;  // =1
  } else {
    return false;
  }
}

bool isEmpty() {  // Antriannya kosong atau tidak
  if (back == 0) {
    return true;
  } else {
    return false;
  }
}

void enqueueAntrian(string data)
{  // Fungsi menambahkan antrian
  if (isFull())
  {
    cout << "Antrian penuh" << endl;
  } else {

    if (isEmpty())
    {  // Kondisi ketika queue kosong
      queueTeller[0] = data;
      front++;
      back++;
    } else {  // Antrianya ada isi
      queueTeller[back] = data;
      back++;
    }
  }
}

void dequeueAntrian()
{ // Fungsi mengurangi antrian
  if (isEmpty())
  {
     cout << "Antrian kosong" << endl;
  } else {
    for (int i = 0; i < back; i++)
    {
```

```
            queueTeller[i] = queueTeller[i + 1];
          }
          back--;
        }
      }

      int countQueue() {  // Fungsi menghitung banyak antrian
        return back;
      }

      void clearQueue()
      {  // Fungsi menghapus semua antrian
        if (isEmpty())
        {
          cout << "Antrian kosong" << endl;
        } else
        {
          for (int i = 0; i < back; i++)
          {
            queueTeller[i] = "";
          }
          back = 0;
          front = 0;
        }
      }

      void viewQueue()
      {  // Fungsi melihat antrian
            cout << "Data antrian teller:" << endl;
            for (int i = 0; i < maksimalQueue; i++)
            {
               if (queueTeller[i] != "")
               {
                  cout << i + 1 << ". " << queueTeller[i] << endl;
               }
               else
               {
                  cout << i + 1 << ". (kosong)" << endl;
          }
        }
      }

      int main()
      {
        enqueueAntrian("Andi");
        enqueueAntrian("Maya");
        viewQueue();
        cout << "Jumlah antrian = " << countQueue() << endl;
        dequeueAntrian();
        viewQueue();
        cout << "Jumlah antrian = " << countQueue() << endl;
        clearQueue();
        viewQueue();
        cout << "Jumlah antrian = " << countQueue() << endl;
        return 0;
}
```

## OUTPUT




## DESKRIPSI PROGRAM

Program ini merupakan implementasi sederhana dari struktur data antrian (queue). Antrian ini memiliki batas maksimum lima elemen yang ditentukan oleh variabel `maksimalQueue`. Dua variabel, `front` dan `back`, digunakan untuk menunjukkan posisi depan dan belakang antrian. Elemen-elemen antrian disimpan dalam array `queueTeller`. Terdapat fungsi-fungsi seperti `isFull()` dan `isEmpty()` untuk memeriksa apakah antrian penuh atau kosong. Fungsi `enqueueAntrian()` menambahkan elemen baru ke belakang antrian jika masih ada ruang kosong. Fungsi `dequeueAntrian()` menghapus elemen dari depan antrian jika antrian tidak kosong. Fungsi `countQueue()` mengembalikan jumlah elemen saat ini dalam antrian. Fungsi `clearQueue()` menghapus semua elemen dalam antrian, dan `viewQueue()` mencetak semua elemen beserta nomor urutannya. Program utama menunjukkan penggunaan fungsi-fungsi ini dengan berbagai operasi seperti penambahan elemen, penghapusan elemen, pengosongan antrian, dan penampilan antrian pada setiap langkahnya. Implementasi ini menunjukkan bagaimana struktur data antrian dapat diatur dan dimanipulasi menggunakan array dalam bahasa pemrograman C++.

# BAB IV

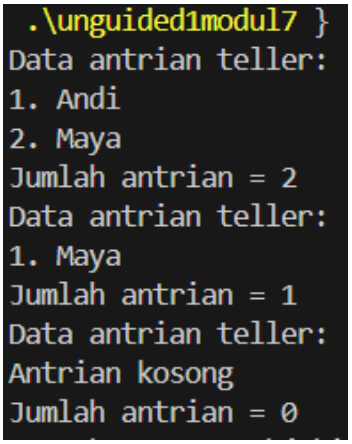
## D. UNGUIDED

### UNGUIDED 1

```
#include <iostream>
using namespace std;

// Definisi struktur Node
struct Node {
    string data;
    Node* next;
};

// Deklarasi variabel untuk front dan back
Node* front = nullptr;
Node* back = nullptr;

// Fungsi untuk mengecek apakah queue penuh (tidak relevan
untuk linked list)
bool isFull() {
    return false; // Linked list tidak terbatas ukuran
}

// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        if (front == nullptr) {
            back = nullptr;
        }
        delete temp;
    }
}

// Fungsi untuk menghitung jumlah elemen dalam queue
int countQueue() {
```

```
            int count = 0;
            Node* temp = front;
            while (temp != nullptr) {
                count++;
                temp = temp->next;
            }
            return count;
        }

        // Fungsi untuk menghapus semua elemen dalam queue
        void clearQueue() {
            while (!isEmpty()) {
                dequeueAntrian();
            }
        }

        // Fungsi untuk menampilkan elemen dalam queue
        void viewQueue() {
            cout << "Data antrian teller:" << endl;
            Node* temp = front;
            int index = 1;
            while (temp != nullptr) {
                cout << index << ". " << temp->data << endl;
                temp = temp->next;
                index++;
            }
            if (index == 1) {
                cout << "Antrian kosong" << endl;
            }
        }

        int main() {
            enqueueAntrian("Andi");
            enqueueAntrian("Maya");
            viewQueue();
            cout << "Jumlah antrian = " << countQueue() << endl;
            dequeueAntrian();
            viewQueue();
            cout << "Jumlah antrian = " << countQueue() << endl;
            clearQueue();
            viewQueue();
            cout << "Jumlah antrian = " << countQueue() << endl;
            return 0;
}
```

**OUTPUT**




**DESKRIPSI PROGRAM**

Program ini menggunakan struktur data antrian (queue) yang diterapkan menggunakan linked list dalam bahasa C++. Dalam implementasinya, antrian direpresentasikan menggunakan struktur `Node` yang terdiri dari dua bagian: `data` untuk menyimpan nilai elemen dan `next` untuk menunjukkan node berikutnya dalam antrian. Variabel `front` dan `back` digunakan untuk menunjukkan node pertama (front) dan terakhir (back) dalam antrian. Pada awalnya, kedua variabel ini diinisialisasi dengan `nullptr` untuk menunjukkan bahwa antrian kosong. Fungsi `isEmpty()` digunakan untuk memeriksa apakah antrian tersebut kosong. Fungsi `enqueueAntrian()` digunakan untuk menambahkan elemen baru ke bagian belakang antrian dengan membuat node baru dan mengatur ulang `back` jika antrian tidak kosong, atau menginisialisasi `front` dan `back` jika antrian kosong. Fungsi `dequeueAntrian()` menghapus elemen dari depan antrian dengan menggeser `front` ke node berikutnya dan menghapus node yang lama. Fungsi `countQueue()` menghitung jumlah elemen dalam antrian dengan mengiterasi melalui linked list. Fungsi `clearQueue()` mengosongkan seluruh antrian dengan mengulang pemanggilan `dequeueAntrian()` sampai antrian kosong. Fungsi `viewQueue()` mencetak semua elemen dalam antrian beserta nomor urutannya, atau menampilkan pesan "Antrian kosong" jika tidak ada elemen dalam antrian. Program utama menunjukkan contoh penggunaan fungsi-fungsi ini dengan menambah, menghapus, dan menampilkan elemen-elemen antrian, serta mengosongkan antrian dan menampilkan jumlah elemen setiap kali operasi dilakukan.

**UNGUIDE 2**

```
#include <iostream>
using namespace std;

// Definisi struktur Node dengan atribut nama mahasiswa dan
NIM mahasiswa
struct Node {
    string nama;
    string nim;
    Node* next;
};

// Deklarasi variabel untuk front dan back
Node* front = nullptr;
Node* back = nullptr;

// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {
    return front == nullptr;
}

// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string nama, string nim) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
    cout << "Enqueued: Nama: " << nama << ", NIM: " << nim <<
endl;
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        if (front == nullptr) {
            back = nullptr;
        }
        cout << "Dequeued: Nama: " << temp->nama << ", NIM: "
<< temp->nim << endl;
        delete temp;
    }
}

// Fungsi untuk menghitung jumlah elemen dalam queue
int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
```

```
        return count;
    }

    // Fungsi untuk menghapus semua elemen dalam queue
    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
    }

    // Fungsi untuk menampilkan elemen dalam queue
    void viewQueue() {
        cout << "Data antrian mahasiswa = " << endl;
        Node* temp = front;
        int index = 1;
        while (temp != nullptr) {
            cout << index << ". Nama: " << temp->nama << ", NIM: "
    << temp->nim << endl;
            temp = temp->next;
            index++;
        }
        if (index == 1) {
            cout << "Antrian kosong" << endl;
        }
    }

    int main() {
        enqueueAntrian("Ahmadan", "2311102038");
        enqueueAntrian("syaridin", "2311102038");
        viewQueue();
        cout << "Jumlah antrian : " << countQueue() << endl;

        dequeueAntrian();
        viewQueue();
        cout << "Jumlah antrian : " << countQueue() << endl;

        clearQueue();
        viewQueue();
        cout << "Jumlah antrian = " << countQueue() << endl;

        return 0;
}
```

**OUTPUT**

```
 .\unguided2modul7 }
Enqueued: Nama: Ahmadan, NIM: 2311102038
Enqueued: Nama: syaridin, NIM: 2311102038
Data antrian mahasiswa =
1. Nama: Ahmadan, NIM: 2311102038
2. Nama: syaridin, NIM: 2311102038
Jumlah antrian : 2
Dequeued: Nama: Ahmadan, NIM: 2311102038
Data antrian mahasiswa =
1. Nama: syaridin, NIM: 2311102038
Jumlah antrian : 1
Dequeued: Nama: syaridin, NIM: 2311102038
Data antrian mahasiswa =
Antrian kosong
Jumlah antrian = 0
```

## DESKRIPSI PROGRAM

Program ini menggunakan linked list untuk menerapkan struktur data antrian (queue) dalam bahasa C++. Setiap node dalam linked list menyimpan informasi mahasiswa dalam bentuk `nama` dan `NIM`, serta memiliki pointer `next` yang menunjukkan ke node berikutnya dalam antrian. Variabel global `front` dan `back` digunakan untuk mengelola node pertama dan terakhir dalam antrian. Fungsi `isEmpty()` memeriksa apakah antrian kosong dengan memeriksa apakah `front` adalah `nullptr`. Fungsi `enqueueAntrian()` menambahkan node baru ke akhir antrian dengan membuat node baru dan menyesuaikan `back`. Fungsi `dequeueAntrian()` menghapus node pertama dari antrian dengan memindahkan `front` ke node berikutnya dan menghapus node yang lama. Fungsi `countQueue()` menghitung jumlah node dalam antrian dengan mengiterasi dari `front` sampai `back`. Fungsi `clearQueue()` mengosongkan antrian dengan menghapus semua node menggunakan `dequeueAntrian()` sampai antrian kosong. Fungsi `viewQueue()` menampilkan semua node dalam antrian dengan menelusuri dari `front` sampai `back`, atau menampilkan pesan "Antrian kosong" jika antrian kosong. Dalam fungsi `main()`, program menunjukkan penggunaan fungsi-fungsi ini dengan menambahkan beberapa node baru, menampilkan isi antrian, menghitung jumlah node, menghapus node pertama, mengosongkan antrian, dan menampilkan antrian setelah setiap operasi dengan informasi tambahan tentang node yang ditambahkan atau dihapus.

# BAB V

## E. KESIMPULAN

Queue adalah jenis struktur data linear yang mengikuti prinsip First In First Out (FIFO), yang berarti elemen yang pertama kali dimasukkan akan menjadi yang pertama kali keluar. Dalam implementasi menggunakan linked list, setiap node menyimpan data dan pointer yang mengarah ke node berikutnya. Variabel `front` menunjukkan node pertama dalam antrian, sedangkan `back` menunjukkan node terakhir. Fungsi-fungsi seperti `enqueueAntrian()` digunakan untuk menambahkan elemen ke dalam antrian, `dequeueAntrian()` untuk menghapus elemen pertama, `isEmpty()` untuk memeriksa apakah antrian kosong, `countQueue()` untuk menghitung jumlah elemen dalam antrian, `clearQueue()` untuk mengosongkan seluruh antrian, dan `viewQueue()` untuk menampilkan elemen dalam antrian, merupakan operasi dasar yang membentuk queue. Pemahaman yang baik tentang pengaturan urutan masuk dan keluar elemen, serta penggunaan pointer untuk menghubungkan setiap node, sangat penting dalam implementasi dan penggunaan struktur data queue dengan menggunakan linked list.

# BAB VI

## F. DAFTAR PUSTAKA


1. Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.
2. *https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/*