

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



DISUSUN OLEH :
MUHAMAD NASRULLOH
2311102044

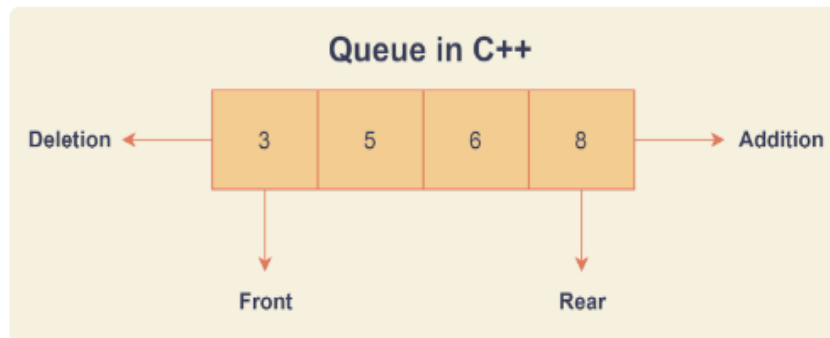
DOSEN :
WAHYU ANDI SAPUTRA, S.PD., M.ENG.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. DASAR TEORI

- Pengertian

Queue dalam C++ adalah kebalikan dari stack, merupakan struktur data yang mengikuti prinsip First In First Out (FIFO), yang berarti elemen yang masuk terlebih dahulu akan keluar terlebih dahulu.



Jadi bisa dikatakan queue adalah antrian (dalam Bahasa Indonesia) yang bila divisualisasikan ibarat di suatu loket pembelian tiket, orang yang datang terlebih dahulu akan diilayani terlebih dahulu.

- Bentuk Operasi

1. Enqueue : menambahkan elemen ke akhir queue
2. Dequeue : menghapus elemen dari depan queue
3. IsEmpty : memeriksa apakah queue kondisi kosong
4. IsFull : memeriksa apakah queue sudah penuh
5. Peek : mendapatkan (mengetahui) nilai elemen terdepan tanpa perlu menghapusnya.
6. Initialize : membuat queue baru tanpa elemen data (kosong)

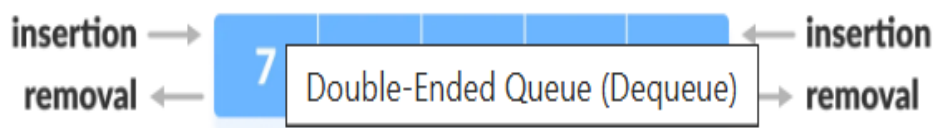
Namun secara umum, queue hanya memiliki 2 operasi utama yaitu enqueue dan dequeue.

- Jenis – Jenis Queue

1. **Simple Queue** : jenis queue yang paling dasar dimana penyisipan item dilakukan di simpul belakang (rear atau tail) dan penghapusan terjadi di simpul depan (front atau head).



2. **Circular Queue** : jenis queue dimana simpul terakhirnya terhubung ke simpul pertama, dikenal juga 'ring buffer' karena semua ujungnya terhubung ke ujung yang lain.
3. **Priority Queue** : jenis queue dimana simpul memiliki beberapa prioritas yang telah ditentukan. Simpul dengan prioritas terbesar akan menjadi yang pertama dihapus dari queue dan penyisipan item akan terjadi sesuai urutan kedatangannya.
4. **Double-Ended Queue (Deque)** : jenis queue yang operasi penyisipan dan penghapusan dapat terjadi di ujung depan dan belakang dari queue.



- Kegunaan Queue

Queue banyak digunakan untuk menangani lalu lintas (traffic) di website, membantu untuk mempertahankan playlist yang ada pada aplikasi media player, digunakan dalam sistem operasi untuk menangani interupsi, membantu dalam melayani permintaan pada satu sumber daya bersama seperti penjadwalan tugas CPU, dan digunakan dalam transfer data asinkronus misal pipeline dan socket.

- Kelebihan dan Kekurangan Queue

Kelebihan queue diantaranya data jumlah besar dapat dikelola secara efisien, operasi seperti penyisipan dan penghapusan dapat dilakukan dengan mudah karena mengikuti aturan masuk pertama keluar pertama (FIFO), cepat untuk komunikasi antar-proses data. Kekurangannya adalah operasi penyisipan dan penghapusan elemen dari tengah cenderung banyak memakan waktu, dan ukuran maksimum queue harus ditentukan sebelumnya.

B. GUIDED

- Guided I

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];

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 // Antriannya 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

```

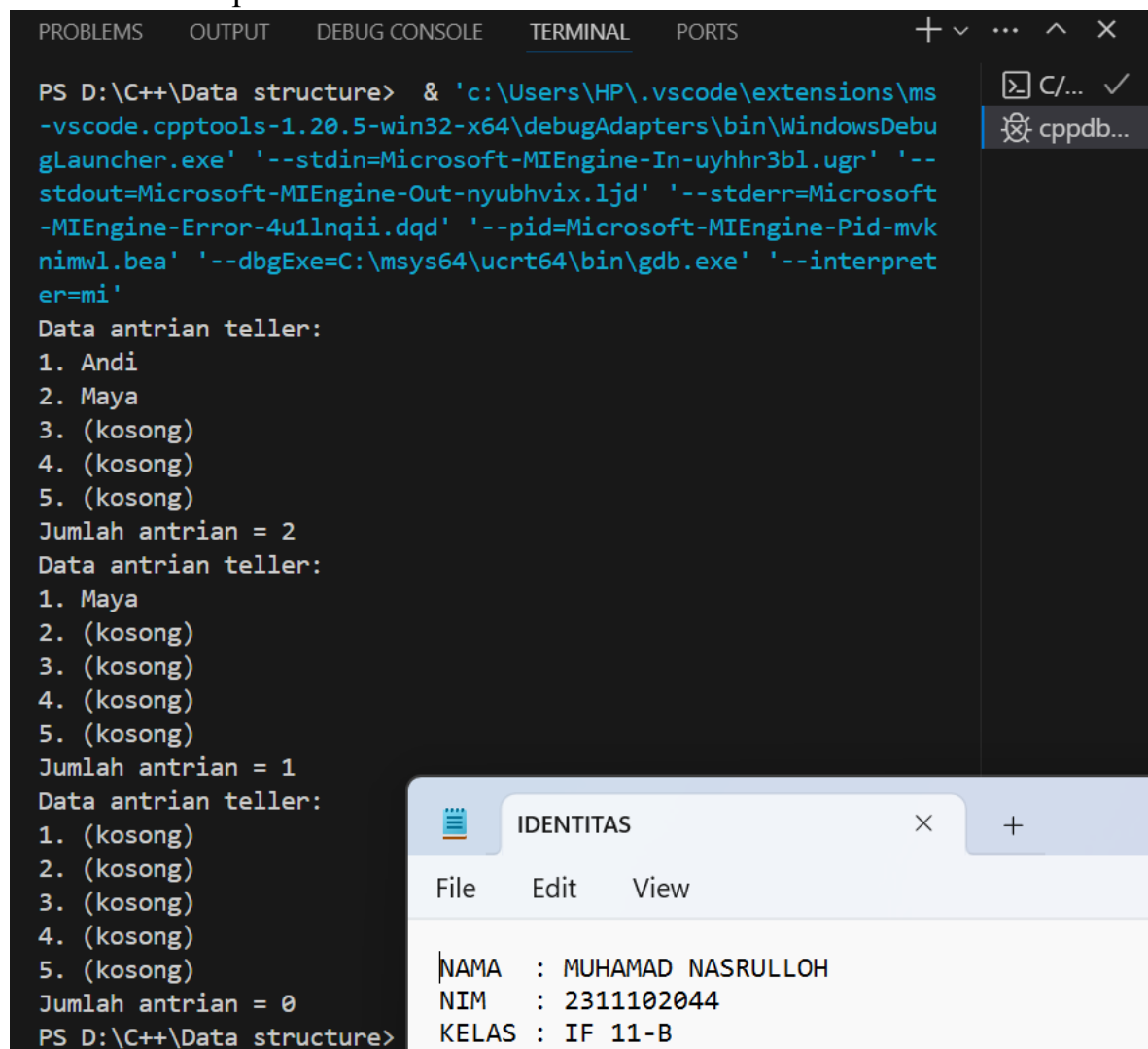
```

{
    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;
}

```

Screenshot Output



The screenshot shows a VS Code interface with a terminal window and a Notepad window. The terminal window displays the output of a C++ program that implements a queue using a static array. The program prompts the user to enter names and checks if the queue is full or empty. The Notepad window, titled 'IDENTITAS', contains student information.

```
PS D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms
-vscodetools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebu
gLauncher.exe' '--stdin=Microsoft-MIEngine-In-uyhhr3bl.ugr' '--
stdout=Microsoft-MIEngine-Out-nyubhvix.ljd' '--stderr=Microsoft
-MIEngine-Error-4u1lnqii.dqd' '--pid=Microsoft-MIEngine-Pid-mvk
nimwl.bea' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpret
er=mi'
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
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
PS D:\C++\Data structure>
```

IDENTITAS

File Edit View

NAMA : MUHAMAD NASRULLOH
NIM : 2311102044
KELAS : IF 11-B

Deskripsi

Program diatas implementasi sederhana dari queue menggunakan array statis dalam c++. Program mencakup operasi dasar pada queue seperti penambahan elemen (enqueue), penghapusan elemen (dequeue), pengecekan apakah queue penuh atau kosong lalu menampilkan isi queue tersebut. Secara garis besarnya :

- deklarasi/inisialisasi : maksimalQueue ditetapkan 5 adalah kapasitas maksimal antrian, front dan back adalah penanda elemen depan dan belakang, queueTeller adalah array, yang menyimpan elemen antrian.
- fungsi : isFull untuk mengecek apakah antrian penuh dengan membandingkan nilai 'back' dengan 'maksimalQueue' ; isEmpty untuk mengecek antrian kosong dengan nilai 'back' apakah sama dengan 0 ; enqueueAntrian untuk menambahkan elemen baru ke antrian, jika penuh akan menampilkan pesan 'antrian penuh', jika kosong elemen ditambahkan ke posisi pertama, lalu jika tidak kosong elemen ditambahkan ke posisi 'back' dan nilai 'back' ditingkatkan

; dequeueAntrian untuk menghapus elemen dari antrian, jika kosong menampilkan pesan 'antrian kosong'; countQueue untuk mengembalikan jumlah elemen dalam antrian berdasar nilai 'back'; clearQueue untuk menghapus semua elemen dalam antrian dengan mengosongkan array 'queueTeller' dan mengatur ulang nilai 'front' dan 'back'; viewQueue untuk menampilkan isi antrian, jika posisi dalam array kosong maka ditampilkan sebagai '(kosong)'.

C. UNGUIDED

- Unguided I

Ubahlah penerapan queue pada bagian guided dari array menjadi linked list!

Source Code

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

// Implementasi linked list
struct Node {
    string data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* back;
    int count;

public:
    Queue() {
        front = nullptr;
        back = nullptr;
        count = 0;
    }

    bool isFull() {
        return false;
    }

    bool isEmpty() {
        return front == nullptr;
    }

    void enqueueAntrian(string data) {
        Node* newNode = new Node();
```



```

        newNode->data = data;
        newNode->next = nullptr;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        count++;
    }

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
        count--;
        if (front == nullptr) {
            back = nullptr;
        }
    }
}

int countQueue() {
    return count;
}

void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* current = front;
    int position = 1;
    while (current != nullptr) {
        cout << position << ". " << current->data << endl;
        current = current->next;
        position++;
    }
    if (position == 1) {
        cout << "Antrian kosong" << endl;
    }
}

```

```

    }
}
};

int main() {
    Queue queue;

    queue.enqueueAntrian("Andi");
    queue.enqueueAntrian("Maya");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeueAntrian();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    return 0;
}

```

Screenshot Output

```

PS D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapter-out-1121wytyj.tw0' '--stderr=Microsoft-MIEngine-Error-j5kribi1.y44' '--pid=Microsoft-MIEngine-Pid-colghfe4.lz2' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Antrian kosong
Jumlah antrian = 0
PS D:\C++\Data structure>

```

IDENTITAS

File Edit View

NAMA : MUHAMAD NASRULLOH
 NIM : 2311102044
 KELAS : IF 11-B

Deskripsi

Program diatas adalah kode program dari guided yang telah dimodifikasi mengimplementasikan queue dengan linked list bukan array lagi seperti sebelumnya. Dalam implementasi ini setiap elemen antrian direpresentasikan sebagai node dalam linked list, dimana setiap nodenya memiliki 2 bagian utama : data (string) dan sebuah pointer yang menunjuk ke node berikutnya. Kelas

‘queue’ berisi operasi dasar yang secara umum sama dengan guided : menambahkan (enqueue), menghapus (dequeue), dan beberapa fungsi lain seperti memeriksa antrian penuh atau kosong, menghitung jumlah elemen antrian, mengosongkan antrian dan menampilkan seluruh elemen antrian. Program utama menggunakan bagian dari kelas ‘queue’ tersebut untuk melakukan operasi-operasi lalu menampilkannya. Dengan linked list, sruktur antrian jadi lebih fleksibel dalam ukuran dan pengelolaanya.

- Unguided II

Dari nomor 1 buatlah konsep antrian dengan atribut nama mahasiswa dan NIM!

Source Code

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

// Implementasi linked list
struct Node {
    string nama;
    string nim;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* back;
    int count;

public:
    Queue() {
        front = nullptr;
        back = nullptr;
        count = 0;
    }

    bool isEmpty() {
        return front == nullptr;
    }

    void enqueueAntrian(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;

        if (isEmpty()) {
```

```

        front = newNode;
        back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
    count++;
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
        count--;
        if (front == nullptr) {
            back = nullptr;
        }
    }
}

int countQueue() {
    return count;
}

void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

void viewQueue() {
    cout << "Jumlah antrian: " << countQueue() << endl;
    cout << "Data antrian mahasiswa:" << endl;
    Node* current = front;
    int position = 1;
    while (current != nullptr) {
        cout << position << ". Nama: " << current->nama << ", NIM: " <<
current->nim << endl;
        current = current->next;
        position++;
    }
    if (position == 1) {
        cout << "Antrian kosong" << endl;
    }
}

```

```

};

int main() {
    Queue queue;
    int choice;
    string nama, nim;

    do {
        cout << "\n";
        cout << "-----\n";
        cout << "\nMenu:\n";
        cout << "1. Tambah Antrian\n";
        cout << "2. Hapus Antrian\n";
        cout << "3. Tampilkan Antrian\n";
        cout << "4. Hapus Semua Antrian\n";
        cout << "5. Exit\n";
        cout << "Pilih: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Masukkan nama mahasiswa: ";
                cin >> nama;
                cout << "Masukkan NIM mahasiswa: ";
                cin >> nim;
                queue.enqueueAntrian(nama, nim);
                break;
            case 2:
                queue.dequeueAntrian();
                break;
            case 3:
                queue.viewQueue();
                break;
            case 4:
                queue.clearQueue();
                cout << "Semua antrian telah dihapus.\n";
                break;
            case 5:
                cout << "Terima kasih.\n";
                break;
            default:
                cout << "Pilihan tidak valid. Silakan masukkan angka 1-
5.\n";
        }
    } while (choice != 5);

    return 0;
}

```

Screenshot Output

```
PS D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-u32bj1lm.dft' '--stdout=Microsoft-MIEngine-Out-030yvbea.hh2' '--stderr=Microsoft-MIEngine-Error-qvavas2k.hnl' '--pid=Microsoft-MIEngine-Pid-t0o55vrn.dbl' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 1
Masukkan nama mahasiswa: Anasrul
Masukkan NIM mahasiswa: 2311102044

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 1
Masukkan nama mahasiswa: Aland
Masukkan NIM mahasiswa: 2311102045

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 1
Masukkan nama mahasiswa: Asshifa
Masukkan NIM mahasiswa: 2311102046

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 2
```

```
-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 3
Jumlah antrian: 2
Data antrian mahasiswa:
1. Nama: Aland, NIM: 2311102045
2. Nama: Asshifa, NIM: 2311102046

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 4
Semua antrian telah dihapus.

-----
Menu:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Hapus Semua Antrian
5. Exit
Pilih: 5
Terima kasih.
PS D:\C++\Data structure>
```

IDENTITAS

File Edit View

NAMA : MUHAMAD NASRULLOH
NIM : 2311102044
KELAS : IF 11-B

Deskripsi

Program diatas adalah kelanjutan dari kode unguided 1 dengan tambahan nama dan NIM mahasiswa, yang mengimplementasikan queue dengan linked list di c++. Dengan sedikit pengubahan melalui program ini user dapat melakukan operasi terhadap antrian mahasiswa, yang terlihat seperti pada pilihan menu : menambahkan mahasiwa dalam antrian, menghapus, menampilkan jumlah dan isi antrian, menghapus semua data mahasiswa dalam antrian, dan exit/keluar program.

D. KESIMPULAN

Queue adalah struktur data yang mengikuti konsep FIFO (First In First Out). Dalam implementasiannya baik dengan array maupun linked list, terdapat kelebihan dan kekurangan yang perlu dipertimbangkan. Implementasi dengan array umumnya lebih sederhana dan butuh ruang memori lebih sedikit, namun ukuran queue statis dan penghapusan elemen depan butuh penggeseran seluruh elemen yang membuat operasi memiliki kompleksitas waktu yang tinggi. Sedangkan dengan linked list, queue bersifat dinamis sehingga tidak ada batasan ukuran dan penghapusan elemen bagian depan lebih cepat, namun setiap nodenya butuh alokasi memori tambahan dan penanganan pointer butuh perhatian khusus untuk menghindari kebocoran memori/salah akses. Jadi pemilihan antara array dan linked list bergantung kebutuhan spesifik. Array cocok untuk queue dengan ukuran tetap dan akses cepat ke elemen, linked list lebih fleksibel untuk queue dengan ukuran dinamis dan operasi penghapusan yang sering dilakukan.

E. REFERENSI

- [1] Vibhor Bhatnagar (2024). Implement std::queue in C++
<https://www.naukri.com/code360/library/implement-std-queue-in-c>.
- [2] Trivusi (2023). Struktur Data Queue: Pengertian, Jenis dan Kegunaanya
<https://www.trivusi.web.id/2022/07/struktur-data-queue.html>.
- [3] Christian Wahyudi. Konsep Queue Dalam Bahasa Pemrograman C++
https://www.academia.edu/18129263/Konsep_QUEUE_Dalam_Bahasa_Pemrograman_C.
- [4] Asisten Praktikum (2024). Modul 7 Queue