

**LAPORAN PRAKTIKUM STRUKTUR  
DATA**

**MODUL VIII  
QUEUE**



**Disusun Oleh :**

NAMA : IVAN RAMADHAN

NIM : 103112400186

**Dosen**

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## A. Dasar Teori

C++ adalah pengembangan dari bahasa C yang dibuat oleh Bjarne Stroustrup sekitar tahun 1980-an. C++ disebut bahasa multi-paradigma, artinya bisa dipakai dengan gaya prosedural (pakai fungsi biasa), berorientasi objek (pakai class dan object), atau bahkan gabungan keduanya. C++ punya dasar-dasar seperti variabel, operator percabangan (if, switch), perulangan (for, while), dan bisa memakai class untuk membuat objek.

## B. Guided (berisi screenshot source code & output program disertai penjelasannya)

### Guided 1 queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

const int MAX = 5;
typedef int infotype;

struct Queue {
    infotype info[MAX];
    int head;
    int tail;
};

void createQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
bool isFullQueue(Queue Q);
void enqueue(Queue &Q,
infotype x);
infotype dequeue(Queue &Q);
void printInfo(Queue Q);

#endif
```

### queue.cpp

```
#include <iostream>
#include "queue.h"

using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}
```

```

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return (Q.tail == MAX - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh" << endl;
    } else {
        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        } else {
            Q.tail++;
        }
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
    infotype x = -1;

    if (isEmptyQueue(Q)) {
        cout << "Queue kosong" << endl;
    } else {
        x = Q.info[Q.head];

        for (int i = Q.head; i < Q.tail; i++) {
            Q.info[i] = Q.info[i + 1];
        }

        Q.tail--;

        if (Q.tail < Q.head) {
            createQueue(Q);
        }
    }
    return x;
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";

    if (isEmptyQueue(Q)) {
        cout << "Queue kosong";
    }
}

```

```
} else {  
    for (int i = Q.head; i <= Q.tail; i++) {  
        cout << Q.info[i] << " ";  
    }  
}  
cout << endl;  
}
```

Main.cpp

```
#include <iostream>  
#include "queue.h"  
  
using namespace std;  
  
int main() {  
    cout << "Hello World" << endl;  
  
    Queue Q;  
    createQueue(Q);  
  
    cout << "-----" << endl;  
    cout << " H - T \t | Queue info" << endl;  
    cout << "-----" << endl;  
  
    printInfo(Q);  
  
    enqueue(Q, 5);  
    printInfo(Q);  
  
    enqueue(Q, 2);  
    printInfo(Q);  
  
    enqueue(Q, 7);  
    printInfo(Q);  
  
    dequeue(Q);  
    printInfo(Q);  
  
    enqueue(Q, 4);  
    printInfo(Q);  
  
    dequeue(Q);  
    printInfo(Q);  
}
```

```
    dequeue(Q);  
    printInfo(Q);  
  
    return 0;  
}
```

Screenshots Output:

1.

```
//  
Hello World  
-----  
H - T | Queue info  
-----  
-1 - -1 | Queue kosong  
0 - 0   | 5  
0 - 1   | 5 2  
0 - 2   | 5 2 7  
0 - 1   | 2 7  
0 - 2   | 2 7 4  
0 - 1   | 7 4  
0 - 0   | 4  
PS C:\xampp\htdocs\Modul Alpro\laprakstd> |
```

Guided 2 queue.h

```

#ifndef QUEUE_H
#define QUEUE_H

const int MAX = 5;
typedef int infotype;

struct Queue {
    infotype info[MAX];
    int head;
    int tail;
};

void createQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
bool isFullQueue(Queue Q);
void enqueue(Queue &Q,
infotype x);
infotype dequeue(Queue &Q);
void printInfo(Queue Q);

#endif

```

#### queue.cpp

```

#include <iostream>
#include "queue.h"

using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return (Q.tail == MAX - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh" << endl;
    } else {

```

```

        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        } else {
            Q.tail++;
        }
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue kosong" << endl;
        return -1;
    }

    infotype x = Q.info[Q.head];
    Q.head++;

    if (Q.head > Q.tail) {
        createQueue(Q);
    }
    return x;
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";

    if (isEmptyQueue(Q)) {
        cout << "Queue kosong";
    } else {
        for (int i = Q.head; i <= Q.tail; i++) {
            cout << Q.info[i] << " ";
        }
    }
    cout << endl;
}

```

Main.cpp

```

#include <iostream>
#include "queue.h"

using namespace std;

int main() {

```

```
cout << "Hello World" << endl;

Queue Q;
createQueue(Q);

cout << "-----" << endl;
cout << " H - T \t | Queue info" << endl;
cout << "-----" << endl;

printInfo(Q);

enqueue(Q, 5);
printInfo(Q);

enqueue(Q, 2);
printInfo(Q);

enqueue(Q, 7);
printInfo(Q);

dequeue(Q);
printInfo(Q);

enqueue(Q, 4);
printInfo(Q);

dequeue(Q);
printInfo(Q);

dequeue(Q);
printInfo(Q);

return 0;
}
```

Screenshots Output:

1.

```
>>
Hello World
-----
  H - T   | Queue info
-----
-1 - -1   | Queue kosong
0 - 0     | 5
0 - 1     | 5 2
0 - 2     | 5 2 7
1 - 2     | 2 7
1 - 3     | 2 7 4
2 - 3     | 7 4
3 - 3     | 4
PS C:\xampp\htdocs\Modul Alpro\laprakstd>
```

### Guided 3 queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

const int MAX = 5;
typedef int infotype;

struct Queue {
    infotype info[MAX];
    int head;
    int tail;
};

void createQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
bool isFullQueue(Queue Q);
void enqueue(Queue &Q, infotype
x);
infotype dequeue(Queue &Q);
void printInfo(Queue Q);

#endif
```

### queue.cpp

```
#include <iostream>
#include "queue.h"

using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
```

```

    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return (Q.tail == MAX - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh" << endl;
    } else {
        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        } else {
            Q.tail++;
        }
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
    infotype x = -1;

    if (isEmptyQueue(Q)) {
        cout << "Queue kosong" << endl;
    } else {
        x = Q.info[Q.head];

        for (int i = Q.head; i < Q.tail; i++) {
            Q.info[i] = Q.info[i + 1];
        }

        Q.tail--;

        if (Q.tail < Q.head) {
            createQueue(Q);
        }
    }
    return x;
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";
}

```

```

    if (isEmptyQueue(Q)) {
        cout << "Queue kosong";
    } else {
        for (int i = Q.head; i <= Q.tail; i++) {
            cout << Q.info[i] << " ";
        }
    }
    cout << endl;
}

```

Main.cpp

```

#include <iostream>
#include "queue.h"

using namespace std;

int main() {
    cout << "Hello World" << endl;

    Queue Q;
    createQueue(Q);

    cout << "-----" << endl;
    cout << " H - T \t | Queue info" << endl;
    cout << "-----" << endl;

    printInfo(Q);

    enqueue(Q, 5);
    printInfo(Q);

    enqueue(Q, 2);
    printInfo(Q);

    enqueue(Q, 7);
    printInfo(Q);

    dequeue(Q);
    printInfo(Q);

    enqueue(Q, 4);
    printInfo(Q);
}

```

```

    dequeue(Q);
    printInfo(Q);

    dequeue(Q);
    printInfo(Q);

    return 0;
}

```

Screenshots Output:

1.

```

>>
Hello World
-----
H - T | Queue info
-----
-1 - -1 | Queue kosong
0 - 0   | 5
0 - 1   | 5 2
0 - 2   | 5 2 7
1 - 2   | 2 7
1 - 3   | 2 7 4
2 - 3   | 7 4
3 - 3   | 4
PS C:\xampp\htdocs\Modul Alpro\laprakstd>

```

Deskripsi:

Program ini mengimplementasikan Circular Queue menggunakan array dengan kapasitas maksimum 5 elemen. Circular Queue adalah struktur data FIFO (First In First Out) yang memanfaatkan seluruh kapasitas array secara efisien dengan cara head dan tail dapat berputar dari akhir kembali ke awal array.

### C. Kesimpulan

Dari implementasi ini, dapat disimpulkan bahwa Circular Queue lebih efisien dibandingkan queue biasa berbasis array karena memaksimalkan penggunaan ruang dan menjaga operasi enqueue serta dequeue tetap cepat. Circular Queue sangat cocok digunakan pada aplikasi

yang membutuhkan antrian berulang atau terus-menerus, seperti buffer data, penjadwalan tugas, atau manajemen sumber daya. Dengan demikian, penggunaan Circular Queue memungkinkan pemanfaatan memori secara optimal sekaligus mempertahankan performa operasi queue yang tinggi.

#### D. Referensi

- W3Resource. (2020). *C++ String Exercises: Convert digit/number to words*.
- GeeksforGeeks. (2020). *Loops in C++ (for, while, do-while)*.