

LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL 8 QUEUE



Disusun Oleh :

NAMA : Maulana Ananta Piliang

NIM : 103112400156

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

H

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX 5

typedef int infotype;

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

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
```

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 && Q.tail == -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;    if (isEmptyQueue(Q))
{
    cout << "Queue kosong" <<
endl;
    return -1;    } else {
x = Q.info[Q.head];    if
(Q.head == Q.tail) {
    Q.head = -1;
    Q.tail = -1;    } else {        for
(int i = Q.head; i < Q.tail; i++) {
    Q.info[i] = Q.info[i + 1];
    }
    Q.tail--;
    }    }
return x;
}

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

cout << endl;

```

```
}
```

Main

```
#include <iostream>

#include "queue.h" using
namespace std;

int main() {
    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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

-----
H - T    | Queue info
-----
-1 -1 empty queue
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
-----
```

Deskripsi:

Program ini merupakan implementasi queue menggunakan array di mana head selalu berada di indeks awal, sedangkan tail bergerak maju saat enqueue.

Pada operasi dequeue, seluruh elemen digeser ke kiri agar head tetap di posisi awal. Metode ini mudah dipahami, tetapi kurang efisien karena membutuhkan proses penggeseran elemen.

Guided 2

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++;
    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) {
        Q.head = -1;
        Q.tail = -1;
    }

    return x;
}

```

```

}

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

```

Screenshots Output

```

>>
>>
-----
H - T   | Queue info
-----
-1 -1 empty queue
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:\Users\ASUS\Documents\Struktur Data\Laporan\modul8\1>

```

Deskripsi:

Program ini head dan tail sama bergerak sesuai operasi dequeue dan enqueue. Elemen yang dihapus tidak menyebabkan pergeseran array sehingga lebih efisien dibanding 1. Namun, slot kosong didepan tidak dapat digunakan kembali, sehingga kapasitas queue tidak dimanfaatkan secara optimal.

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 + 1) % MAX == Q.head);
}

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.tail + 1) % MAX;
    }
    Q.info[Q.tail] = x;
```

```

    }
}

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

    return -1;
}

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

    if (Q.head == Q.tail) {
        Q.head = -1;
        Q.tail = -1;
    } else {
        Q.head = (Q.head + 1) % MAX;
    }
return x;
}

void printInfo(Queue Q) {    cout <<
Q.head << " " << Q.tail << " ";    if
(isEmptyQueue(Q)) {        cout <<
"empty queue";

    } else {        int i = Q.head;
while (true) {            cout <<
Q.info[i] << " ";            if (i ==
Q.tail) break;            i = (i + 1)
% MAX;

        }
}

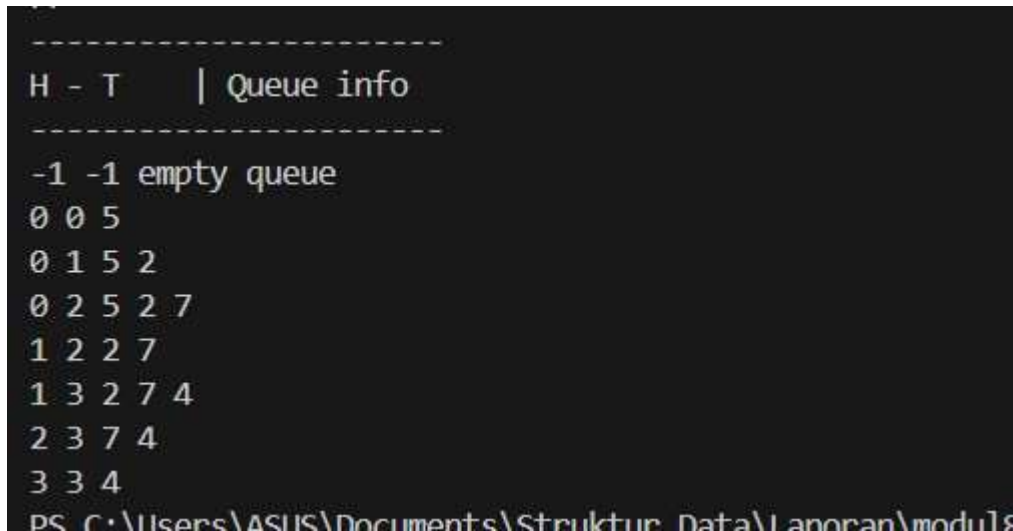
```

```

    }
    cout << endl;
}

```

Screenshots Output



```

-----
H - T   | Queue info
-----
-1 -1 empty queue
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:\Users\ASUS\Documents\Struktur Data\Laporan\modul8

```

Deskripsi:

Program ini menggunakan konsep array berputar (circular), dimana head dan tail bergerak secara melingkar menggunakan operasi modulo. Metode ini memungkinkan pemanfaatan penuh seluruh array, karena slot kosong dapat digunakan kembali.

C. Kesimpulan

Kesimpulan dari keempat program C++ tersebut adalah bahwa masing-masing program mengajarkan konsep dasar pemrograman dan logika perhitungan menggunakan array, pointer, fungsi, serta struktur perulangan. Dari program pertama, dipelajari cara mengolah data mahasiswa, menghitung rata-rata, dan menampilkan hasil dalam format tabel. Program kedua memperkenalkan array dinamis dan penggunaan pointer untuk mengakses serta memanipulasi data. Program ketiga menekankan penggunaan fungsi terpisah (modularisasi) untuk menghitung rata-rata, nilai maksimum, dan minimum dari sekumpulan data. Sementara program keempat mengajarkan konsep perulangan bersarang untuk mencetak pola angka berbentuk segitiga. Secara keseluruhan, keempat program ini membantu memahami logika dasar, manipulasi data, penggunaan fungsi, array, pointer, serta struktur kontrol dalam bahasa pemrograman C++.

D. Referensi

- Alhazmi, A., & Huang, W. (2020). Teaching loops concept through visualization construction. *International Journal of Instruction*. 13(4), 399-114.

- Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2020). Dampak penggunaan bahasa pemrograman terhadap pemahaman konsep loop pada siswa K-12. *Pendidikan dan Teknologi Informasi*, 25(1), 987-1000.
- Stroustrup, B. (1999). Tinjauan umum bahasa pemrograman C++. *ACM SIGPLAN Notices*, 34(4), 7-18.
- Stroustrup, B. (1999). Tinjauan umum bahasa pemrograman C++. *ACM SIGPLAN Notices*, 34(4), 7-18.