LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL I
PENGENALAN CODE BLOCKS

Disusun Oleh :
NAMA : Jauza Rasyiq Hernanta
NIM : **103112430033**

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Queue (dibaca : kyu) merupakan struktur data yang dapat diumpamakan seperti sebuah antrean. Misalkan antrean pada loket pembelian tiket Kereta Api. Orang yang akan mendapatkan pelayanan yang pertama adalah orang pertamakali masuk dalam antrean tersebut dan yang terakhir masuk dia akan mendapatkan layanan yang terakhir pula. Jadi prinsip dasar dalam Queue adalah FIFO (First in Fisrt out), proses yang pertama masuk akan diakses terlebih dahulu. Dalam pengimplementasian struktur Queue dalam C dapat menggunakan tipe data array dan linked list.

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

Guided 1

Main.cpp

main.cpp > main()

```cpp
#include "queue.h"
#include "queue.cpp"
#include <iostream>
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;
}
```

Queue,cpp

queue.cpp > ...

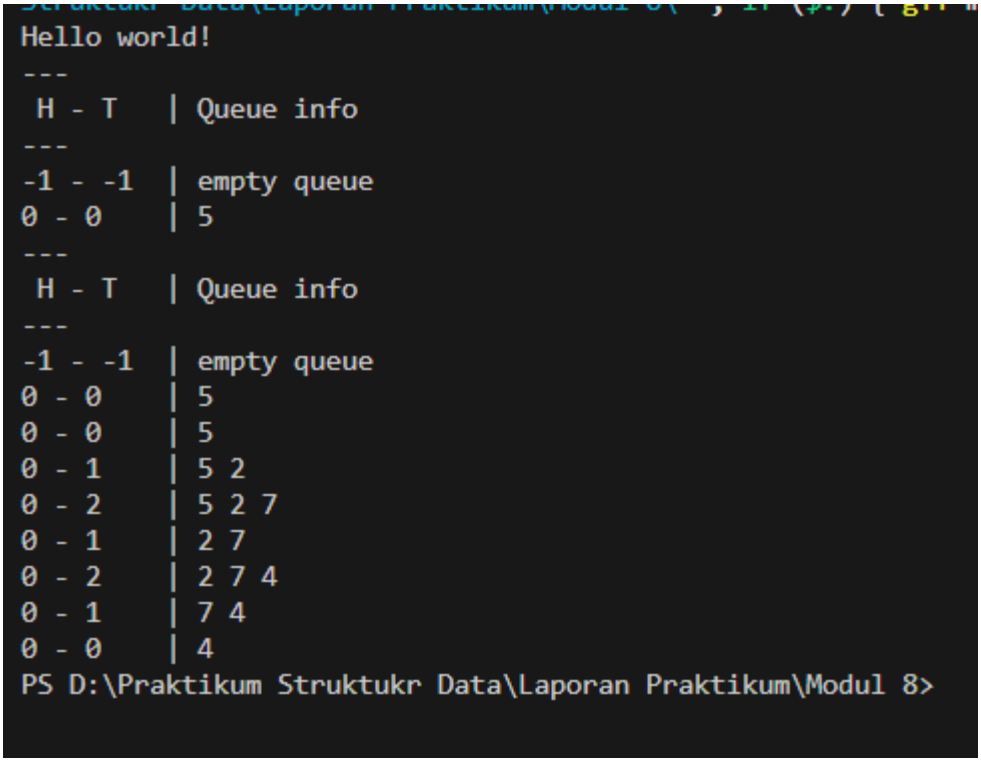
```cpp
#include "queue.h"
#include <iostream>
using namespace std;

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_SIZE - 1;
}

void enqueue(Queue &Q, infotype x) {
    if (Q.head == -1) {
        Q.head = 0;
        Q.tail = 0;
        Q.info[0] = x;
    } else if (!isFullQueue(Q)) {
        Q.tail++;
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        return -1;
    }

    infotype 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--;
    }

```

```
45
46      return x;
47  }
48
49  void printInfo(Queue Q) {
50      if (isEmptyQueue(Q)) {
51          cout << Q.head << " - " << Q.tail << "\t | empty queue" << endl;
52      } else {
53          cout << Q.head << " - " << Q.tail << "\t | ";
54          for (int i = Q.head; i <= Q.tail; i++) {
55              cout << Q.info[i] << " ";
56          }
57          cout << endl;
58      }
59  }
```

Queue.h

```
queue.h > ...
1   #ifndef QUEUE_H
2   #define QUEUE_H
3
4   const int MAX_SIZE = 5;
5
6   typedef int infotype;
7
8   struct Queue {
9       infotype info[MAX_SIZE];
10      int head;
11      int tail;
12  };
13
14  void CreateQueue(Queue &Q);
15  bool isEmptyQueue(Queue Q);
16  bool isFullQueue(Queue Q);
17  void enqueue(Queue &Q, infotype x);
18  infotype dequeue(Queue &Q);
19  void printInfo(Queue Q);
20
21  #endif
```

Screenshots Output


Hello world!

H - T | Queue info

-1 - -1 | empty queue
0 - 0 | 5

H - T | Queue info

-1 - -1 | empty queue
0 - 0 | 5
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 D:\Praktikum Struktukr Data\Laporan Praktikum\Modul 8>