LAPORAN PRAKTIKUM

- **Identitas Praktikum**

  Nama MK : Struktur Data
  Kode MK : CCK2AAB4
  Bobot SKS : 4 SKS
  Tempat : L-Program, Gedung DC, lantai 3
  Hari, tanggal : Selasa, 12 November 2024
  Jam : 12:30-15:30 WIB
  Topik praktikum : Modul-8 QUEUE

- **Identitas Mahasiswa**

  Nama lengkap : Afad Fath Musyarof Halim
  NIM : 2211104030
  Program Studi : S-1 Software Engineering

- **Hasil Praktikum**

8. QUEUE

8.1 Pengertian

QUEUE Adalah struktur data yang berbentuk seperti antrian dimana urutan masuk pertama akan di akses pertama kali dan urutan terakhir akan di akhir terakhir

8.2 Operasi

8.2.1 Insert (Enqueue)

Menambahkan data di urutan terakhir antrian

```
void enqueue(string value) {
    if (isFull()) {
        cout << "Antrian penuh\n";
        return;
    }
    if (isEmpty()) {
        front = 0;
    }
    rear = (rear + 1) % MAX;
    items[rear] = value;
    cout << "Tambah antrian: " << value << endl;
}
```

8.2.2 Delete (Dequeue)

Menghapus data dari urutan pertama

```
string dequeue() {
    if (isEmpty()) {
        cout << "Antrian penuh\n";
        return "";
    }
    string data = items[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % MAX;
    }
    return data;
}
```

8.2.3 Display

Menampilkan isi antrian

```
void display() const {
    if (isEmpty()) {
        cout << "Antrian penuh\n";
        return;
    }
    cout << "Daftar Antrian: ";
    int i = front;
    while (true) {
        cout << items[i] << " ";
        if (i == rear) break;
        i = (i + 1) % MAX;
    }
    cout << endl;
}
```

8.2.4 isFull

mengecek apakah sudah penuh

```
bool isFull() const { return (rear + 1) % MAX == front; }
```

8.2.5 isEmpty

mengecek apakah masih kosong

```
bool isEmpty() const { return front == -1; }
```

8.3 Latihan

8.3.1 Buatlah ADT Queue

```
Type infotype: integer
Type Queue:  <
     info : array [5] of infotype {index array dalam C++
     dimulai dari 0}
     head, tail : integer
>
prosedur CreateQueue (in/out Q: Queue)
fungsi isEmptyQueue (Q: Queue) → boolean
fungsi isFullQueue (Q: Queue) → boolean
prosedur enqueue (in/out Q: Queue, in x: infotype)
fungsi dequeue (in/out Q: Queue) → infotype
prosedur printInfo (in Q: Queue)
```

Buatlah implementasi ADT *Queue* pada *file* "*queue*.cpp" dengan menerapkan mekanisme *queue* Alternatif 1 (*head* diam, *tail* bergerak).

```
1  int main() {
2      cout << "Hello World" << endl;
3      Queue Q;
4      createQueue(Q);
5
6      cout<<"----------------------"<<endl;
7      cout<<" H - T \t | Queue info"<<endl;
8      cout<<"----------------------"<<endl;
9      printInfo(Q);
10     enqueue(Q,5); printInfo(Q);
11     enqueue(Q,2); printInfo(Q);
12     enqueue(Q,7); printInfo(Q);
13     dequeue(Q); printInfo(Q);
14     enqueue(Q,4); printInfo(Q);
15     dequeue(Q); printInfo(Q);
16     dequeue(Q); printInfo(Q);
17
18     return 0;
19 }
```

Gambar 8-18 *Main Queue*

```
Hello world!
----------------------
 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 - 0   | 7
 0 - 1   | 7 4
 0 - 0   | 4
 -1 - -1 | empty queue
```

Gambar 8-17 *Output Queue*

- Code
    - Queue.h

```h
#ifndef QUEUE_H
#define QUEUE_H

const int MAX_SIZE = 5;
typedef int infotype;

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

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

#endif // QUEUE_H
```

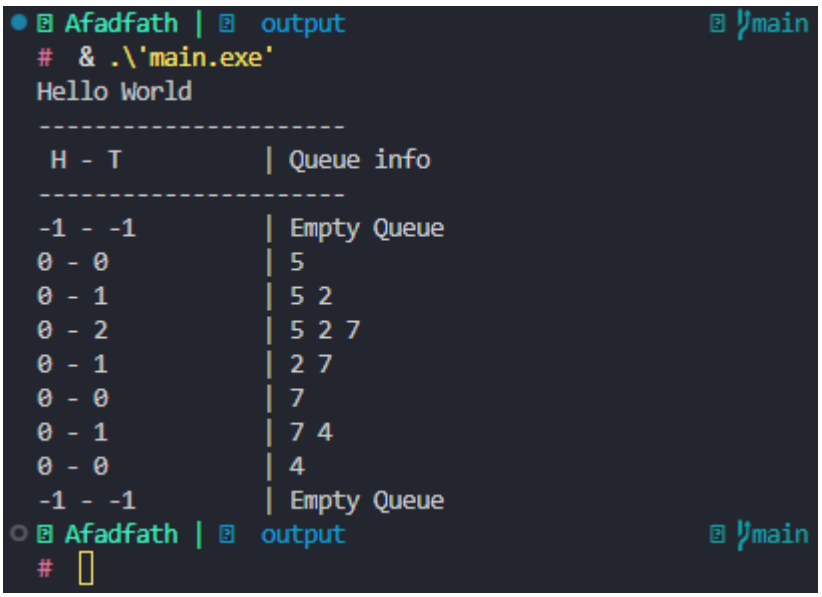
- Queue.cpp

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

using namespace std;

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

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

bool isFullQueue(const Queue &Q) {
    return (Q.tail + 1) % MAX_SIZE == Q.head;
}

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

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Empty Queue" << endl;
        return -1;
    }
    infotype 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) {
        Q.head = -1;
        Q.tail = -1;
    }
    return x;
}

void printInfo(const Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "-1 - -1  \t| Empty Queue" << endl;
        return;
    }

    int i = Q.head;
    cout << Q.head << " - " << Q.tail << "    \t| ";

    while (true) {
        cout << Q.info[i] << " ";
        if (i == Q.tail) break;
        i = (i + 1) % MAX_SIZE;
    }
    cout << endl;
}
```

- Main.cpp

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

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);
    dequeue(Q); printInfo(Q);

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

    dequeue(Q); printInfo(Q);
    dequeue(Q); printInfo(Q);

    return 0;
}
```

- Output



8.3.2 Buatlah implementasi ADT Queue pada file "queue.cpp" dengan menerapkan mekanisme queue Alternatif 2 (head bergerak, tail bergerak).

- Code
    - Peubahan pada Queue.cpp

```
void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Full Queue" << endl;
        return;
    }
    if (isEmptyQueue(Q)) {
        Q.head = 0;
    }
    Q.tail = (Q.tail + 1) % MAX_SIZE;
    Q.info[Q.tail] = x;
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Empty Queue" << 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_SIZE;
    }
    return x;
}
```

- Output

```
Afadfath | output                                  main
#  & .\'main.exe'
Hello World
---------------------
 H - T          | Queue info
---------------------
-1 - -1         | Empty Queue
0 - 0           | 5
0 - 1           | 5 2
0 - 2           | 5 2 7
1 - 2           | 2 7
2 - 2           | 7
2 - 3           | 7 4
3 - 3           | 4
-1 - -1         | Empty Queue
Afadfath | output                                  main
#
```

8.3.3 Buatlah implementasi ADT Queue pada file "queue.cpp" dengan menerapkan mekanisme queue Alternatif 3 (head dan tail berputar).

- Code
    - Perubahan pada Queue.cpp

```
void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Full Queue" << endl;
        return;
    }
    if (isEmptyQueue(Q)) {
        Q.head = 0;
    }
    Q.tail = (Q.tail + 1) % MAX_SIZE;
    Q.info[Q.tail] = x;
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Empty Queue" << 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_SIZE;
    }
    return x;
}
```

- Output

```
Afadfath | output                                  main
# & .\'main.exe'
Hello World
---------------------
 H - T          | Queue info
---------------------
-1 - -1         | Empty Queue
0 - 0           | 5
0 - 1           | 5 2
0 - 2           | 5 2 7
1 - 2           | 2 7
2 - 2           | 7
2 - 3           | 7 4
3 - 3           | 4
-1 - -1         | Empty Queue
Afadfath | output                                  main
#
```