LAPORAN

MODUL 4

SINGLY LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh :

NAMA :

Devi Nurliana

NIM :

103112400144

Dosen :

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. DASAR TEORI

Queue adalah bentuk struktur data linear yang bekerja berdasarkan prinsip FIFO, yaitu data yang masuk lebih dulu akan keluar lebih dulu. Konsep queue bisa dibayangkan seperti antrean di kehidupan sehari-hari, seperti antrean di loket atau kasir. Di dalam queue terdapat dua bagian utama, yaitu head yang menunjukkan data yang paling depan dan tail yang menunjukkan data yang paling belakang. Operasi utama dalam queue terdiri dari enqueue, yaitu menambahkan data ke bagian belakang (tail), dan dequeue, yaitu mengambil data dari bagian depan (head). Struktur data queue dapat dibuat menggunakan array atau linked list.

B. GUIDED

1. Guide 1
   a. Source Code
   - singlylist.h

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

struct  Queue
{
   int info[MAX_QUEUE];
   int head;
   int tail;
   int count;
};

void createQueue(Queue &Q);

bool isEmpty(Queue Q);

bool isFull(Queue Q);

void enqueue(Queue &Q, int x);

int dequeue(Queue &Q);

void printInfo(Queue Q);

#endif
```

   - queue.cpp

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

using namespace std;

void createQueue(Queue &Q) {
```

```
    Q.head = 0;
    Q.tail = -1;
    Q.count = 0;
}

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE;
}

void enqueue(Queue &Q, int x) {
    if (!isFull(Q)) {
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.info[Q.tail] = x;
        Q.count++;
    } else {
        cout << "Antrean Penuh! " << endl;
    }
}

int dequeue(Queue &Q) {
    if (!isEmpty(Q)) {
        int x = Q.info[Q.head];
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count--;
        return x;
    } else {
        cout << "Antrean Kosong! " << endl;
        return -1;
    }
}

void printInfo(Queue Q) {
    cout << "Isi Antrean: [";
    if (!isEmpty(Q)) {
        int i = Q.head;
        int n = 0;
        while (n < Q.count) {
            cout << Q.info[i];
            i = (i + 1) % MAX_QUEUE;
            n++;
        }
    cout << "]";
    } else {
        cout << "Antrean Kosong!";
    }
}
```

- main.cpp

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

using namespace std;

int main() {
   Queue Q;

   createQueue(Q);
   printInfo(Q);

   cout <<"\n enqueu 3 elemen"<<endl;
   enqueue(Q, 5);
   printInfo(Q);
   enqueue(Q, 2);
   printInfo(Q);
   enqueue(Q, 7);
   printInfo(Q);

   cout <<"\n Dequeue 1 elemen"<<endl;

   cout<< "Elemen keluar: " << dequeue(Q) << endl;
   printInfo(Q);

   cout <<"\n enqueu 1 elemen"<<endl;
   enqueue(Q, 4);
   printInfo(Q);

   cout <<"\n Dequeue 2 elemen"<<endl;
   cout<< "Elemen keluar: " << dequeue(Q) << endl;
   cout << "Elemen keluar: " << dequeue(Q) << endl;
   printInfo(Q);

   return 0;

}
```

b. Screenshot Output

```
Isi Antrean: [Antrean Kosong!
 enqueu 3 elemen
Isi Antrean: [5]Isi Antrean: [52]Isi Antrean: [527]
 Dequeue 1 elemen
Elemen keluar: 5
Isi Antrean: [27]
 enqueu 1 elemen
Isi Antrean: [274]
 Dequeue 2 elemen
Elemen keluar: 2
Elemen keluar: 7
Isi Antrean: [4]
```

c. Deskripsi

Program ini berfungsi untuk mengimplementasikan struktur data **queue (antrean)** menggunakan array statis dengan konsep **FIFO (First In First Out)**. Program menyediakan operasi dasar untuk menambah, menghapus, dan menampilkan data dalam antrean.

- Pendefinisian struktur dan ADT
  Program mendefinisikan struktur Queue yang berisi array penyimpan data, penunjuk head dan tail, serta variabel count. Selain itu, dideklarasikan fungsi-fungsi dasar untuk mengelola antrean.
- Implementasi fungsi
  Program mengimplementasikan fungsi pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan data (enqueue), penghapusan data (dequeue), serta penampilan isi antrean.
- Proses utama
  Program membuat antrean kosong, kemudian melakukan beberapa operasi enqueue dan dequeue. Setiap perubahan ditampilkan untuk melihat kondisi antrean.
- Selesai
  Setelah seluruh operasi dijalankan, program berakhir dan mengembalikan nilai 0.

C. UNGUIDED

1. Alternatif 1
   a. Source Code
   - queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;
```

```
struct Queue {
    infotype info[MAX_QUEUE];
    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 "queue.h"
#include <iostream>
using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = -1;
}

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

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

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

infotype dequeue(Queue &Q) {
    infotype x = Q.info[0];
    for (int i = 0; i < Q.tail; i++) {
        Q.info[i] = Q.info[i + 1];
    }
    Q.tail--;
    return x;
}

void printInfo(Queue Q) {
```

```
    cout << Q.head << " - " << Q.tail << "\t | ";
    for (int i = 0; i <= Q.tail; i++)
        cout << Q.info[i] << " ";
    cout << endl;
}
```

- main.cpp

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

b. Screenshot Output

```
----------------------
 H - T   | Queue info
----------------------
0 - -1   |
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
```

c. Deskripsi

Program ini mengimplementasikan ADT Queue menggunakan array dengan mekanisme Alternatif 1, yaitu head tetap (diam) dan tail bergerak. Pada mekanisme ini, elemen antrean selalu dimulai dari indeks pertama array, sehingga posisi head tidak pernah berubah.

1. Pendefinisian struktur dan ADT
   Program mendefinisikan struktur Queue yang terdiri dari array sebagai tempat penyimpanan data, serta variabel head dan tail sebagai penunjuk posisi elemen. Beberapa operasi dasar seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan data, penghapusan data, dan penampilan isi antrean juga dideklarasikan.

2. Implementasi mekanisme antrean
   Pada mekanisme Alternatif 1, setiap operasi enqueue menambahkan elemen di posisi tail yang bergerak ke kanan. Saat operasi dequeue dilakukan, elemen terdepan dihapus dan seluruh elemen lainnya digeser ke depan agar head tetap berada di posisi awal.

3. Proses utama program
   Program membuat antrean kosong, kemudian melakukan beberapa operasi enqueue dan dequeue. Setelah setiap operasi, isi antrean ditampilkan untuk menunjukkan perubahan data dan pergerakan elemen dalam antrean.

4. Selesai
   Setelah seluruh proses dijalankan, program berakhir dan mengembalikan nilai 0.

2. Alternatif 2
   a. Source Code
      - queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;

struct Queue {
   infotype info[MAX_QUEUE];
   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

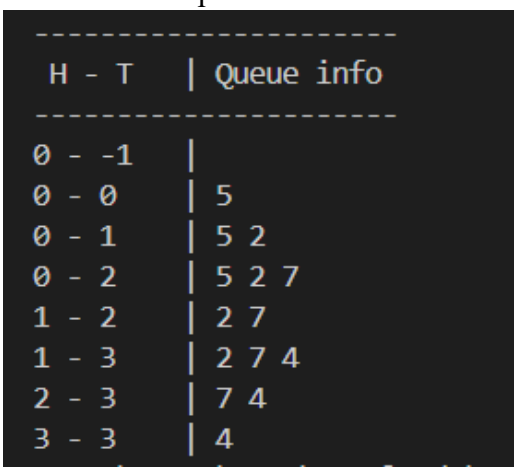
```
#include "queue.h"
#include <iostream>
using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return Q.head > Q.tail;
}

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

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

infotype dequeue(Queue &Q) {
    return Q.info[Q.head++];
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";
    for (int i = Q.head; i <= Q.tail; i++)
        cout << Q.info[i] << " ";
    cout << endl;
}
```

- main.cpp

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

b. Screenshot Output



c. Deskripsi

Program ini mengimplementasikan **ADT Queue menggunakan array** dengan mekanisme **Alternatif 2**, yaitu **head bergerak dan tail bergerak**. Pada mekanisme ini, elemen antrean tidak perlu digeser ketika terjadi penghapusan data.

1. Pendefinisian struktur dan ADT

   Program mendefinisikan struktur Queue yang terdiri dari array sebagai tempat penyimpanan data, serta variabel head dan tail sebagai penunjuk posisi elemen antrean. Selain itu, dideklarasikan operasi dasar untuk mengelola antrean seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan data, penghapusan data, dan penampilan isi antrean.

2. Implementasi mekanisme antrean

   Pada mekanisme Alternatif 2, operasi enqueue dilakukan dengan menambahkan elemen pada posisi tail, sedangkan operasi dequeue dilakukan dengan memindahkan posisi head ke elemen berikutnya. Data yang telah keluar tidak digeser, sehingga proses penghapusan menjadi lebih cepat dibandingkan Alternatif 1.

3. Proses utama program

   Program membuat antrean kosong, kemudian melakukan beberapa operasi enqueue dan dequeue. Setelah setiap operasi, isi antrean ditampilkan untuk melihat perubahan posisi head, tail, dan isi antrean.

4. Selesai
   Setelah seluruh proses dijalankan, program berakhir dan mengembalikan nilai 0.

3. Alternatif 3
   a. Source Code
      - queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;

struct Queue {
    infotype info[MAX_QUEUE];
    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

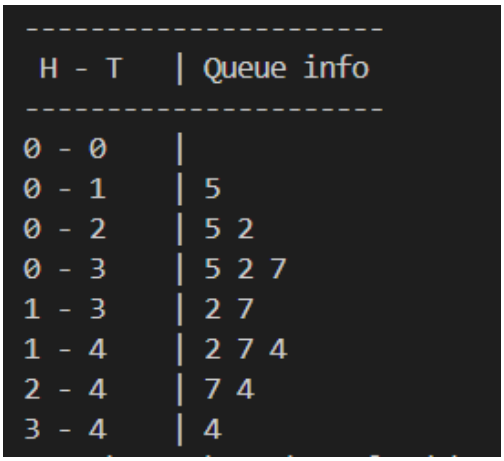
```
#include "queue.h"
#include <iostream>
using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = 0;
}

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

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

void enqueue(Queue &Q, infotype x) {
```

```
    if (!isFullQueue(Q)) {
        Q.info[Q.tail] = x;
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
    }
}

infotype dequeue(Queue &Q) {
    infotype x = Q.info[Q.head];
    Q.head = (Q.head + 1) % MAX_QUEUE;
    return x;
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";
    int i = Q.head;
    while (i != Q.tail) {
        cout << Q.info[i] << " ";
        i = (i + 1) % MAX_QUEUE;
    }
    cout << endl;
}
```

- main.cpp

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

b. Screenshot Output



c. Deskripsi

Program ini mengimplementasikan ADT Queue menggunakan array dengan mekanisme Alternatif 3, yaitu head dan tail bergerak secara berputar (circular queue). Mekanisme ini memungkinkan penggunaan memori secara lebih efisien karena ruang kosong yang sudah dilewati dapat digunakan kembali.

1. Pendefinisian struktur dan ADT

   Program mendefinisikan struktur Queue yang terdiri dari array sebagai tempat penyimpanan data, serta variabel head dan tail sebagai penunjuk posisi elemen antrean. Selain itu, dideklarasikan operasi dasar seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan data, penghapusan data, dan penampilan isi antrean.

2. Implementasi mekanisme antrean

   Pada mekanisme Alternatif 3, operasi enqueue dan dequeue dilakukan dengan memindahkan posisi head dan tail secara berputar menggunakan operasi modulo. Dengan cara ini, antrean dapat memanfaatkan kembali indeks array yang kosong tanpa perlu menggeser data.

3. Proses utama program

   Program membuat antrean kosong, kemudian melakukan beberapa operasi enqueue dan dequeue. Setelah setiap operasi, isi antrean ditampilkan untuk menunjukkan pergerakan head, tail, dan kondisi antrean.

4. Selesai

   Setelah seluruh proses dijalankan, program berakhir dan mengembalikan nilai 0.

D. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian tiga alternatif mekanisme queue menggunakan array, dapat disimpulkan bahwa setiap alternatif memiliki cara kerja dan karakteristik yang berbeda. Pada Alternatif 1, head tetap dan tail bergerak, sehingga proses dequeue memerlukan pergeseran seluruh elemen. Cara ini sederhana namun kurang efisien karena membutuhkan waktu lebih lama saat penghapusan data.

Pada Alternatif 2, baik head maupun tail bergerak tanpa adanya pergeseran data. Mekanisme ini lebih efisien dibandingkan Alternatif 1, tetapi memiliki kelemahan karena ruang array yang sudah dilewati tidak dapat digunakan kembali sehingga antrean dapat terlihat penuh meskipun masih terdapat ruang kosong.

Alternatif 3 merupakan mekanisme yang paling efisien karena menggunakan konsep circular queue, di mana head dan tail bergerak secara berputar. Dengan cara ini, seluruh ruang array dapat dimanfaatkan secara optimal tanpa perlu menggeser data. Oleh karena itu, Alternatif 3 lebih direkomendasikan untuk penggunaan queue yang membutuhkan efisiensi waktu dan memori.

E. REFERNSI

- Modul Praktikum Struktur Data. *Modul 8: Queue*. Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas _____, Tahun _____.
- Kadir, A. (2012). *Dasar Pemrograman C++*. Yogyakarta: Andi Publisher.
- Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++*. Pearson Education.