LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL QUEUE

Disusun Oleh :

NAMA : Ayu Setyaning Tyas

NIM : 103112430119

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Struktur Data Queue adalah konsep mendasar dalam ilmu komputer yang digunakan untuk menyimpan dan mengelola data dalam urutan tertentu. Ia mengikuti prinsip " First in, First out " (FIFO) , di mana elemen pertama yang ditambahkan ke antrian adalah yang pertama dihapus. Ini digunakan sebagai buffer dalam sistem komputer di mana terdapat ketidaksesuaian kecepatan antara dua perangkat yang berkomunikasi satu sama lain. Misalnya, CPU dan keyboard serta dua perangkat dalam jaringan. Antrean juga digunakan dalam algoritma Sistem Operasi seperti Penjadwalan CPU dan Manajemen Memori, serta banyak algoritma standar seperti Pencarian Luas Pertama Grafik, Penelusuran Urutan Level Pohon.

Queue dapat melakukan implementasi array dengan konsep melingkar atau circular. Circular akan memperlakukan array sebagai buffer melingkar yang berjalan maju dan mundur, saat memasukkan elemen akan ditambahkan ke bagian depan, sedangkan saat menghapus elemen akan mengurangi bagian belakangnya.

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

Guided 1

Code *queue.h*

```
#ifndef QUEUE_H // jika QUEUE_H belum didefinisikan
#define QUEUE_H //

#define MAX_QUEUE 5

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

// fungsi untuk mengecek apakah queue kosong atau tidak
bool isEmpty(queue Q);

// fungsi untuk mengecek apakah queue penuh atau tidak
bool isFull(queue Q);

// prosedur untuk menambahkan elemen ke dalam queue (enqueue)
void enqueue(queue &Q, int x);

//  fungsi untuk menghapus dan mengembalikan elemen dari queue (dequeue)
int dequeue(queue &Q);

// prosedur untuk menampilkan semua isi queue
void printInfo(queue Q);
```

```
#endif
```

Code *queue.cpp*

```
#include "queue.h"
#include <iostream>
using namespace std; // menggunakan namespace standar agar tidak perlu menulis std::

// definisi prosedur untuk membuat queue kosong
void createQueue(queue &Q)
{
   Q.head = 0; // set kepala/head ke indeks 0
   Q.tail = 0; // set ekor/tail ke indeks 0
   Q.count = 0; // set jumlah elemen/count ke 0
}

// definisi fungsi untuk mengecek apakah queue kosong atau tidak
bool isEmpty(queue Q)
{
   return Q.count == 0; // kembalikan true jika jumlah elemen/count adalah 0
}

// definisi fungsi untuk mengecek apakah queue penuh atau tidak
bool isFull(queue Q)
{
   return Q.count == MAX_QUEUE; // kembalikan true jika jumlah elemen/count sama dengan
kapasitas maksimum
}

void enqueue(queue &Q, int x)
{
   if (!isFull(Q)) // jika queue tidak penuh
   {
      Q.info[Q.tail] = x; // tambahkan elemen x di posisi tail
      Q.tail = (Q.tail + 1) % MAX_QUEUE; // masukkan data  (x) ke posisi ekor (tail)
      // pindahkan ekor secara circular (melingkar)
      Q.count++; // tingkatkan jumlah elemen/count
   }
   else // jika queue penuh
   {
      cout << "Antrean penuh!" << endl; // tampilkan pesan error
   }
}

// definisi fungsi untuk menghapus elemen (dequeue)
int dequeue(queue &Q)
{
```

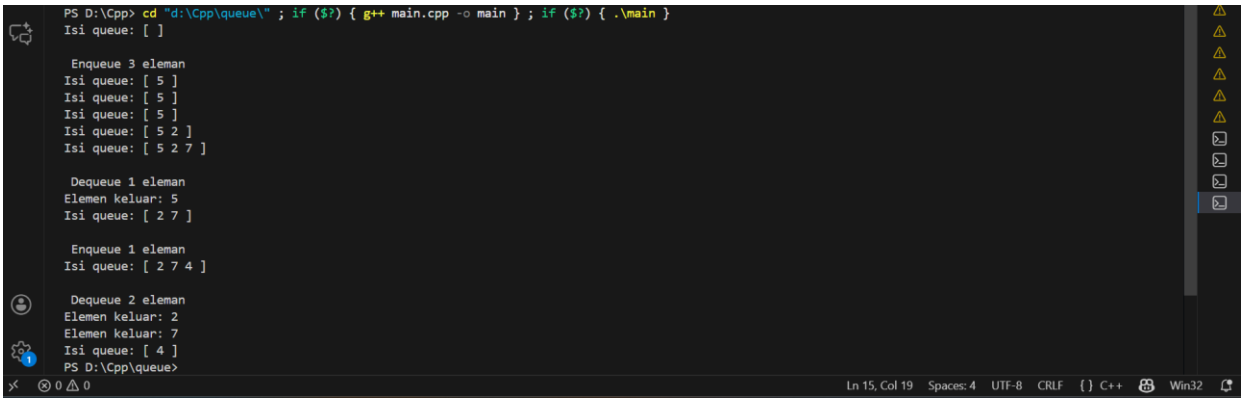
```
    if (!isEmpty(Q)) // jika queue tidak kosong
    {
        int x = Q.info[Q.head]; // ambil elemen di posisi head
        Q.head = (Q.head + 1) % MAX_QUEUE; // pindahkan kepala secara circular (melingkar)
        Q.count--; // kurangi jumlah elemen/count
        return x; // kembalikan elemen yang dihapus
    }
    else // jika queue kosong
    {
        cout << "Antrean kosong!" << endl; // tampilkan pesan error
        return -1; // kembalikan nilai -1 sebagai indikasi error
    }
}

void printInfo(queue Q)
{
    cout << "Isi queue: [ ";
    if (!isEmpty(Q)) // jika queue tidak kosong
    {
        int i = Q.head; // mulai dari posisi head
        int n = 0;
        while (n < Q.count) // ulangi sebanyak jumlah elemen/count
        {
            cout << Q.info[i] << " "; // tampilkan elemen di posisi i
            i = (i + 1) % MAX_QUEUE; // pindahkan i secara circular (melingkar)
            n++; // tambahkan penghitung
        }
    }
    cout << "]" << endl;
}
```

Code *main.cpp*

```
#include <iostream> // menyertakan library untuk input dan output
#include "queue.h" // menyertakan file header queue kita
#include "queue.cpp"

using namespace std;

int main()
{
    queue Q; // deklarasi variabel Q bertipe queue

    createQueue(Q); // inisialisasi
    printInfo(Q); // tampilkan isi queue

    cout << "\n Enqueue 3 eleman" << endl;
```

```
    enqueue(Q, 5);
    printInfo(Q);
    enqueue(Q, 2);
    printInfo(Q);
    enqueue(Q, 7);
    printInfo(Q);

    cout << "\n Dequeue 1 eleman" << endl;
    // hapus 1 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // tampilkan isi queue setelah dequeue

    cout << "\n Enqueue 1 eleman" << endl;
    enqueue(Q, 4);
    printInfo(Q);

    cout << "\n Dequeue 2 eleman" << endl;
    // hapus 2 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    // hapus 2 elemen lagi dan tampilkan isi queue setelah dequeue
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // tampilkan isi queue

    return 0;
}
```

Screenshoot Output



Deskripsi:

Program ini menggunakan mekanisme circular atau berputar dengan atribut head, tail, count, dan array info untuk dijadikan tempat penyimpanan data. Program akan meminta memasukkan isi dengan 3 element yang berurut. Setiap dilakukan enqueue program akan menampilkan isi queue yang terus di tambah sampai ke 3 element. Setelah itu elemen pada baris pertama akan di keluarkan menggunakan fungsi dequeue. Lalu enqueue bisa digunakan untuk menambahkan data atau elemen, program kembali mengeluarkan 2 elemen dengan dequeue hingga menyisakan 1 elemen saja.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

Code *queue.h*

```
#ifndef QUEUE2_H
#define QUEUE2_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
```

Code *queue.cpp*

```
#include <iostream>
#include "queue2.h"
using namespace std;

void createQueue(queue &Q) {
   Q.head = -1;
   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)) {
      cout << "Queue penuh!\n";
      return;
   }

   if (isEmpty(Q)) {
      Q.head = 0;
      Q.tail = 0;
   } else {
      Q.tail++;
   }

   Q.info[Q.tail] = x;
   Q.count++;
}

int dequeue(queue &Q) {
   if (isEmpty(Q)) {
      cout << "Queue kosong!\n";
      return -1;
   }

   int 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--;
   }

   Q.count--;
   return x;
}

void printInfo(queue Q)
{
   cout << Q.head << " - " << Q.tail << "   | ";

   if (isEmpty(Q)) {
      cout << "empty queue\n";
      return;
```

```
    }

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

Code *main.cpp*

```
#include <iostream>
#include "queue2.h"
using namespace std;

int main()
{
    cout << "Hello World\n";
    cout << "-----------------------------\n";
    cout << "H - T   |\tQueue info\n";
    cout << "-----------------------------\n";

    queue Q;
    createQueue(Q);

    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

Deskripsi:

Program ini memiliki head yang selalu berada di indeks 0, sementara tail terus bergerak maju setiap dilakukan operasi enqueue. Ketika di dequeue elemen indeks 0 akan di hapus dan semua elemen lainnya akan di geser ke kiri.

Unguided 2

Code *queue.h*

**tidak ada perubahan pada code*

Code *queue.cpp*

- hanya ada perubahan para void *dequeue*

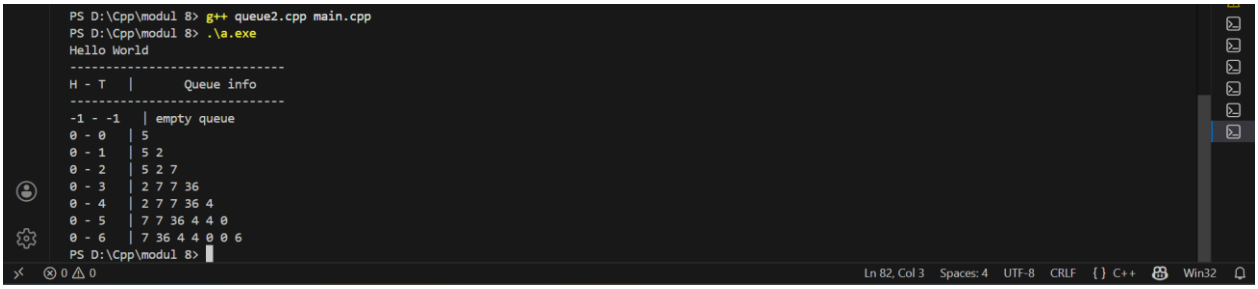
**Sama dengan code sebelumnya hanya saja ada sedikit perubahan dari Q.head--; menjadi :*

```
Q.head++;
```

Code *main.cpp*

**tidak ada perubahan pada code*

Screenshots Output



Deskripsi:

Program ini memiliki kode yang tidak jauh beda dari program yang pertama tetapi memiliki pembeda yakni head ikut bergerak Bersama tail sehingga keduanya akan bergerak maju setiap kali operasi enqueue dan dequeue di jalankan. Tidak ada proses perpindahan lain karena semua indeks bergerak kea rah kanan.

Unguided 3

Code *queue.h*

**tidak ada perubahan pada code*

Code *queue.cpp*

- *enqueue*

**Sama dengan code sebelumnya hanya saja ada sedikit perubahan dari Q.tail++; menjadi :*

```
Q.tail = (Q.tail + 1) % MAX_QUEUE
```

- *dequeue*

Sama dengan code sebelumnya hanya saja ada sedikit perubahan dari Q.head--;
menjadi :

```
Q.head = (Q.head + 1) % MAX_QUEUE;
```

- *printInfo*

Sama dengan code sebelumnya hanya saja ada sedikit perubahan dari

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

menjadi :

```
int i = Q.head;
for (int c = 0; c < Q.count; c++) {
   cout << Q.info[i] << " ";
   i = (i + 1) % MAX_QUEUE;
}
```

Screenshoot Output

Deskripsi

Program ini juga kurang lebih sama dengan yang pertama penbedanya terrletak pada pergerakan head dan tail yang bergerak memutar atau bisa di sebut konsep circular. Dengan car aini ruang array dapat dimanfaatkan sepenuhnya tanpa ada false full.

D. Kesimpulan

Implementasi ADT Queue dengan cara antrian melingkar sudah berhasil dilakukan dan berfungsi sesuai dengan prinsip FIFO, di mana elemen pertama yang masuk adalah yang pertama keluar. Dengan memanfaatkan pergerakan head dan tail secara melingkar, program dapat memakai seluruh kapasitas array tanpa perlu melakukan pergeseran, sehingga proses enqueue dan dequeue jadi lebih cepat dan efisien.

Secara umum, antrian adalah struktur data yang penting dan sering digunakan sebagai area penyimpanan sementara. Antrian ini ada di banyak aplikasi seperti penjadwalan CPU, pengelolaan memori, pencarian BFS, dan komunikasi antar perangkat. Dari percobaan ini, bisa disimpulkan bahwa antrian melingkar memberikan penggunaan memori yang lebih baik dan merupakan solusi yang efektif untuk mengatur data dalam antrian.

E. Referensi

https://www.geeksforgeeks.org/dsa/queue-data-structure/
https://www.geeksforgeeks.org/dsa/introduction-and-array-implementation-of-queue/