# LAPORAN PRAKTIKUM
# STRUKTUR DATA

## MODUL VIII
## QUEUE


**Disusun Oleh :**
NAMA : ABYAN RAHMAN AL FARIZ
NIM : 103112430021

**Dosen**
FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA**
**FAKULTAS INFORMATIKA**
**TELKOM UNIVERSITY PURWOKERTO**

**2025**

A. Dasar Teori

Queue atau antrian adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung disebut dengan sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain (disebut dengan sisi depan atau front).

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

Guided 1 (queue.cpp)

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

using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = 0;
    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.info[Q.tail] = x;
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.count++;
    } else {
        cout << "Antrian 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 << "Antrian kosong!" << endl;
        return -1;
    }
}

void printInfo(Queue Q) {
    cout << "Isi Queue: [ ";
    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 << "]" << endl;
}
```

Guided 1 (queue.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);
```

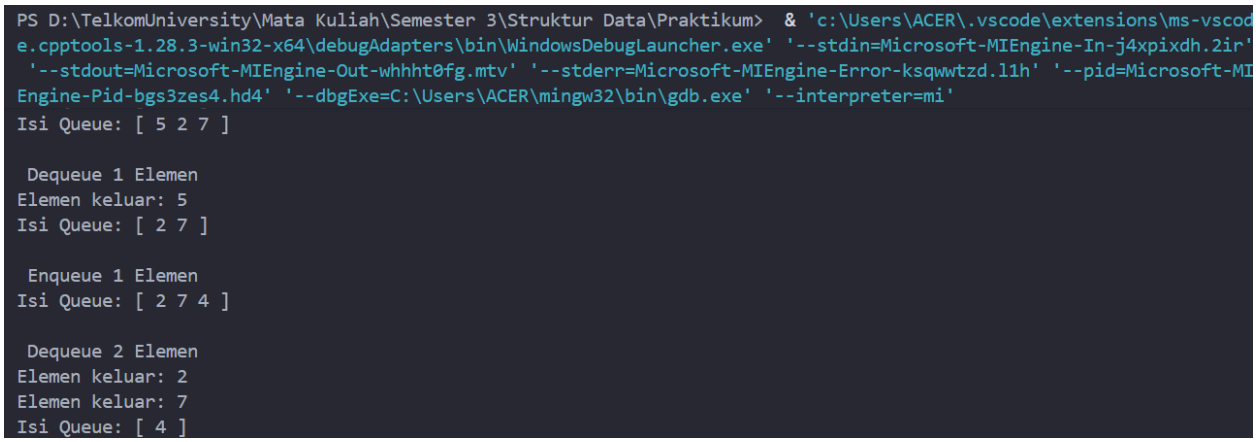
```
bool isEmpty(Queue Q);

bool isFull(Queue Q);

void enqueue(Queue &Q, int x);

int dequeue(Queue &Q);

void printInfo(Queue Q);

#endif
```

Guided 1 (main.cpp)

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

using namespace std;

int main()
{
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout << "\n Enqueue 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 Enqueue 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;
}
```

Screenshots Output:




Deskripsi:

Program ini mengimplementasikan struktur data Queue menggunakan teknik Circular Queue, yaitu mekanisme antrian yang memanfaatkan array secara melingkar (circular) sehingga ruang penyimpanan dapat digunakan kembali secara optimal tanpa perlu menggeser elemen.

Struktur queue disimpan dalam tipe data Queue yang memiliki tiga komponen penting:

- info[] sebagai array penyimpanan elemen,
- head sebagai penanda posisi elemen yang akan dikeluarkan,
- tail sebagai posisi untuk menambahkan elemen baru,
- count untuk menghitung banyaknya elemen dalam antrian.

Proses inisialisasi dilakukan melalui fungsi createQueue(), yang mengatur

semua indeks dan penghitung elemen ke nilai awal. Proses menambah elemen (enqueue) dilakukan dengan meletakkan data pada posisi tail, kemudian tail digeser secara melingkar menggunakan operasi modulo. Sebaliknya, proses menghapus elemen (dequeue) dilakukan dengan mengambil data pada posisi head, lalu head juga digeser ke indeks berikutnya menggunakan modulo.

Fungsi isEmpty() digunakan untuk mengecek apakah antrian kosong, sedangkan isFull() mengecek apakah antrian telah penuh. Untuk menampilkan isi antrian, fungsi printInfo() mencetak setiap elemen dari head hingga jumlah elemen yang ada saat ini.

Pada file main.cpp, program melakukan serangkaian operasi seperti menambah beberapa elemen, menghapus elemen tertentu, serta menampilkan kondisi antrian setelah setiap perubahan. Dengan pendekatan circular queue, bahkan ketika indeks head dan tail mencapai batas array, antrian tetap dapat digunakan kembali tanpa reset manual, sehingga pemanfaatan ruang menjadi lebih efisien.

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

Unguided 1 (queue.cpp)

```
#include "queue.h"

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 == 4); // array size 5 (index 0-4)
}

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

```
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmptyQueue(Q)) {
        Q.head = 0;
        Q.tail = 0;
    } else {
        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];

    if (Q.head == Q.tail) {
        // kembali jadi kosong
        Q.head = -1;
        Q.tail = -1;
    } else {
        // geser elemen ke kiri (ALTERNATIF 1)
        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" << endl;
        return;
```

```
    }

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

Unguided 1 (queue.h)

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED

#include <iostream>
using namespace std;

typedef int infotype;

struct Queue {
    infotype info[5];
    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
```

Unguided 1 (main.cpp)

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

```cpp
#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);
    enqueue(Q, 4); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    dequeue(Q);   printInfo(Q);

    return 0;
}
```

Screenshots Output:

```
PS D:\TelkomUniversity\Mata Kuliah\Semester 3\Struktur Data\Praktikum>  & 'c:\Users\ACER\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-jars3xvn.izs' '--stdout=Microsoft-MIEngine-Out-at4mxuwc.kh0' '--stderr=Microsoft-MIEngine-Error-cttshd3m.u1r' '--pid=Microsoft-MIEngine-Pid-paajz3xl.n5l' '--dbgExe=C:\Users\ACER\mingw32\bin\gdb.exe' '--interpreter=mi'
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 - 2 | 2 7 4
0 - 1 | 7 4
0 - 0 | 4
```

Deskripsi:

Program ini merupakan implementasi ADT Queue menggunakan Array dengan mekanisme head diam, tail bergerak sebagaimana yang dicontohkan pada modul. Queue

bekerja dengan prinsip FIFO (First In, First Out), artinya elemen yang pertama masuk akan menjadi elemen pertama yang keluar. Struktur queue didefinisikan dalam file queue.h, berisi array info[5] untuk menyimpan data dan dua variabel penanda, yaitu head untuk posisi elemen paling awal dan tail untuk posisi elemen terakhir.

File queue.cpp berisi implementasi seluruh operasi dasar queue. Fungsi createQueue() menginisialisasi queue dalam keadaan kosong dengan memberi nilai -1 pada head dan tail. Fungsi isEmptyQueue() dan isFullQueue() digunakan untuk memeriksa apakah queue sedang kosong atau penuh. Proses penambahan elemen dilakukan melalui enqueue(), yaitu dengan menempatkan data baru pada indeks tail + 1, sedangkan penghapusan elemen dilakukan menggunakan dequeue(), yang mengambil elemen pada posisi head dan kemudian menggeser semua elemen lainnya ke kiri agar head tetap berada pada indeks 0 (sesuai mekanisme Alternatif 1). Fungsi printInfo() digunakan untuk menampilkan kondisi head, tail, serta isi queue pada setiap langkah program.

Pada file main.cpp, program menguji seluruh operasi queue dengan melakukan serangkaian proses enqueue dan dequeue, kemudian menampilkan hasilnya. Output yang dihasilkan menunjukkan pergerakan nilai tail, perubahan isi array, serta transisi queue dari kondisi kosong, terisi, hingga kosong kembali.

Unguided 2 (queue.cpp)

```
#include "queue.h"

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 == 4);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmptyQueue(Q)) {
        Q.head = 0;
        Q.tail = 0;
    } else {
        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];

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

    return x;
}

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

    if (isEmptyQueue(Q)) {
        cout << "empty queue" << endl;
        return;
```

```
    }

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

Unguided 2 (queue.h)

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED

#include <iostream>
using namespace std;

typedef int infotype;

struct Queue {
    infotype info[5];
    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
```

Unguided 2 (main.cpp)

```
#include <iostream>
#include "queue.h"
#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);
    enqueue(Q, 4); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    dequeue(Q);   printInfo(Q);

    return 0;
}
```

Screenshots Output:

```
PS D:\TelkomUniversity\Mata Kuliah\Semester 3\Struktur Data\Praktikum>  & 'c:\Users\ACER\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ukabjtyd.wn5' '--stdout=Microsoft-MIEngine-Out-x0ahdf1d.mlc' '--stderr=Microsoft-MIEngine-Error-hmmh3elh.nxc' '--pid=Microsoft-MIEngine-Pid-bn0skc1d.3it' '--dbgExe=C:\Users\ACER\mingw32\bin\gdb.exe' '--interpreter=mi'
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
1 - 3 | 2 7 4
2 - 3 | 7 4
3 - 3 | 4
```

Deskripsi:

Program ini merupakan implementasi ADT Queue menggunakan array dengan mekanisme Alternatif 2, yaitu model di mana baik head maupun tail bergerak mengikuti proses enqueue dan dequeue. Struktur queue didefinisikan dalam file queue.h, dengan array info[5] sebagai tempat penyimpanan data, serta dua penanda posisi yaitu head untuk elemen paling depan dan tail untuk elemen paling belakang.

Pada file queue.cpp, fungsi createQueue() menginisialisasi queue dalam keadaan kosong dengan memberi nilai awal -1 pada head dan tail. Fungsi isEmptyQueue() dan isFullQueue() digunakan untuk memeriksa apakah queue sedang kosong atau sudah penuh. Proses penambahan elemen dilakukan melalui enqueue(), yaitu dengan menempatkan data pada posisi tail + 1. Jika queue sebelumnya kosong, baik head maupun tail diatur ke indeks 0. Sebaliknya, fungsi dequeue() menghapus elemen di posisi head dan mengembalikannya. Setelah penghapusan, head akan bergerak maju ke indeks berikutnya. Jika setelah operasi head dan tail menjadi sama, queue kembali menjadi kosong dengan mengatur keduanya menjadi -1. Tidak ada proses pergeseran elemen array, sehingga operasi menjadi lebih efisien dibanding Alternatif 1.

Fungsi printInfo() menampilkan nilai indeks head dan tail serta elemen-elemen queue sesuai rentang head–tail. Pada file main.cpp, serangkaian operasi enqueue dan dequeue dijalankan untuk menunjukkan perubahan kondisi queue dalam setiap tahapan.

Unguided 3 (queue.cpp)

```
#include "queue.h"

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 + 1) % 5 == Q.head);
}

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

```
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmptyQueue(Q)) {
        Q.head = 0;
        Q.tail = 0;
    } else {
        Q.tail = (Q.tail + 1) % 5;
    }

    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) % 5;
    }

    return x;
}

void printInfo(Queue Q) {
    if (isEmptyQueue(Q)) {
        cout << "-1 - -1 | empty queue" << endl;
        return;
    }

    cout << Q.head << " - " << Q.tail << " | ";

    int i = Q.head;
    while (true) {
        cout << Q.info[i] << " ";
```

```
        if (i == Q.tail) break;
        i = (i + 1) % 5;
    }
    cout << endl;
}
```

Unguided 3 (queue.h)

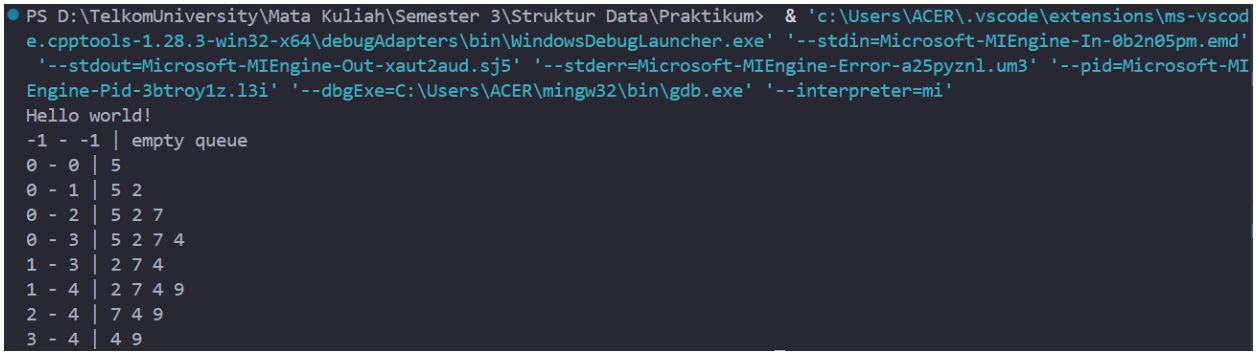
```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED

#include <iostream>
using namespace std;

typedef int infotype;

struct Queue {
    infotype info[5];
    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
```

Unguided 3 (main.cpp)

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

int main() {
    cout << "Hello world!" << endl;
```

```
    Queue Q;
    createQueue(Q);

    printInfo(Q);
    enqueue(Q, 5); printInfo(Q);
    enqueue(Q, 2); printInfo(Q);
    enqueue(Q, 7); printInfo(Q);
    enqueue(Q, 4); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    enqueue(Q, 9); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    dequeue(Q);   printInfo(Q);

    return 0;
}
```

Screenshots Output:




Deskripsi:

Program ini mengimplementasikan ADT Queue dengan mekanisme Circular Queue, yaitu model antrian yang memungkinkan head dan tail bergerak secara melingkar menggunakan operasi modulo. Dengan mekanisme ini, ketika tail mencapai akhir array, ia akan kembali ke indeks 0 selama posisi tersebut belum terisi. Hal ini membuat penggunaan array lebih efisien dibanding Alternatif 1 dan 2 karena tidak ada proses shifting elemen, dan ruang kosong di depan tetap bisa dipakai.

Struktur queue didefinisikan dalam file queue.h, dengan info[5] sebagai array penyimpan data dan dua indeks head serta tail yang mengontrol posisi antrian. Pada queue.cpp, fungsi createQueue() menginisialisasi queue dengan head = -1 dan tail = -1 sebagai tanda bahwa queue kosong. Fungsi isEmptyQueue() mengecek apakah queue

kosong, sedangkan isFullQueue() memeriksa apakah seluruh slot array sudah terisi menggunakan rumus circular (tail + 1) % 5 == head.

Operasi enqueue() menambahkan elemen baru ke posisi tail. Jika queue masih kosong, head dan tail sama-sama diatur ke 0. Jika tail mencapai batas array, ia akan bergerak ke depan menggunakan modulo. Sementara itu, fungsi dequeue() menghapus elemen pada posisi head dan memajukan head juga menggunakan modulo. Jika setelah penghapusan head bertemu tail, queue kembali dianggap kosong.

Fungsi printInfo() digunakan untuk menampilkan seluruh isi queue sesuai urutan FIFO, dengan pergerakan indeks circular. Pada file main.cpp, program menjalankan serangkaian operasi enqueue dan dequeue untuk menunjukkan bagaimana queue dapat bergerak melingkar dan tetap memanfaatkan ruang secara penuh.

## D. Kesimpulan

Queue adalah struktur data linier yang menerapkan konsep FIFO (First In, First Out), di mana elemen yang pertama masuk akan menjadi elemen yang pertama keluar. Pada implementasinya menggunakan array, terdapat beberapa pendekatan untuk mengatur pergerakan head dan tail agar operasi enqueue dan dequeue dapat berjalan dengan benar. Secara keseluruhan, konsep Queue menekankan pengelolaan data secara terurut dan terstruktur, sedangkan Circular Queue memberikan solusi paling optimal dalam pemanfaatan array sebagai media penyimpanan antrian.

## E. Referensi

**https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/**

**https://www.nblognlife.com/2014/05/c-konsep-queue.html**

**https://www.softwareseni.co.id/blog/queue-adalah-pengertian-tipe-dan-contoh-implementasi**