

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL VIII
QUEUE**



Disusun Oleh :

NAMA : Dealova Agta Syahlevi

NIM : 103112400124

Dosen

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

C++ adalah bahasa pemrograman yang dikembangkan oleh Bjarne Stroustrup sebagai pengembangan dari bahasa C. Bahasa ini mendukung pemrograman prosedural maupun berorientasi objek, jadi fleksibel digunakan dalam pembuatan aplikasi, game, sampai sistem operasi. Keunggulan C++ ada pada efisiennya yang tinggi serta kemampuannya menjrmbatani pemrograman tingkat rendah dan tinggi.

Dalam perkuliahan C++ sering digunakan karena strukturnya hampir sama dengan bahasa C namun memiliki fitur yang lebih modern. Jadi bisa lebih memudahkan kami mahasiswa memahami dasar-dasar pemrogrman, struktur data, dan algoritma, sekaligus mengenalkan konsep berorientasi objek. Selain itu, bahasa C++ juga membiasakan kami mahasiswa dengan bahasa yang banyak dipakai dalam industri perangkat lunak dan oengembangan sistem.

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

Guided1
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
```

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 ke indeks 0
```

```
    Q.tail = 0; // Set ekor ke indeks 0
```

```
    Q.count = 0; // Set jumlah elemen ke 0
```

```
}
```

```
// Definisi fungsi untuk mengecek apakah queue kosong
```

```
bool isEmpty(Queue Q) {
```

```
    return Q.count == 0; //Kembalikan true jika jumlah elemen adalah 0
```

```
}
```

```
// Definisi fungsi untuk mengecek apakah queue penuh
```

```

bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE; // Kembalikan true jika jumlah elemen
    sama dengan maks
}

// Definisi prosedur untuk menambahkan elemen (enqueue)
void enqueue(Queue &Q, int x) {
    if (!isFull(Q)) { // Jika queue tidak penuh
        Q.info[Q.tail] = x; // Masukkan data (x) ke posisi ekor (tail)
        // Pindahkan ekor secara circular (memutar)
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.count++; //Tambah jumlah elemen
    } 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 data di posisi kepala (head)
        //Pindahkan kepala secara circular (memutar)
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count--; // Kurangi jumlah elemen
        return x; // Kembalikan data yang diambil
    } else { // Jika queue kosong
        cout << "Antrean Kosong!" << endl; //Tampilkan pesan error
        return -1; // Kembalikan nilai -1 sebagai tanda error
    }
}

```

```

// Definisi prosedur untuk menampilkan isi queue
void printInfo(Queue Q) {
    cout << "Isi Queue: [ "; // Tampilkan awalan
    if (!isEmpty(Q)) { // jika tidak kosong
        int i = Q.head; // Mulai dari kepala
        int n = 0; // Penghitung elemen yang sudah dicetak
        while (n < Q.count) { // Ulangi sebanyak jumlah elemen
            cout << Q.info[i] << " "; // Cetak info
            i = (i + 1) % MAX_QUEUE; // Geser 'i' secara circular
            n++; // Tambah penghitung
        }
    }
    cout << "]" << endl; // Tampilkan akhiran
}

```

Main.cpp

```

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

using namespace std; // Menggunakan namespace standar

// Fungsi utama program
int main() {
    Queue Q; // Deklarasikan variabel queue bernama Q

    createQueue(Q); // Panggil prosedur untuk inisialisasi queue
}

```

```
    printInfo(Q); //Tampilkan isi queue (seharusnya kosong)

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

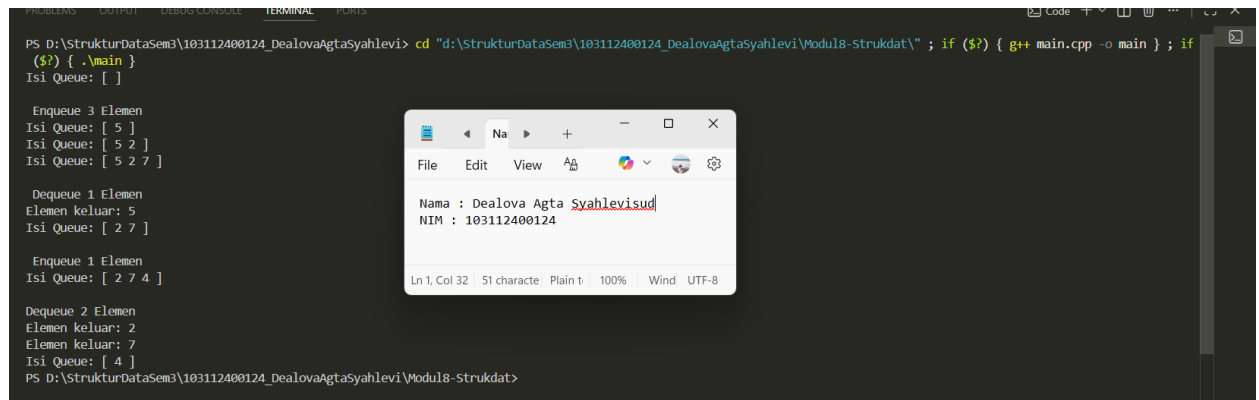
    cout << "\n Dequeue 1 Elemen" << endl;
    // Hapus 1 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // Tampilkan isi queue setelah dequeue

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

    cout << "\nDequeue 2 Elemen" << endl;
    // Hapus 1 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    // Hapus 1 elemen lagi dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // Tampilkan isi queue

    return 0;
}
```

Screenshots Output



```
PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevisud> cd "d:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevisud\Modul8-StrukturData" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Isi Queue: [ ]

Enqueue 3 Elemen
Isi Queue: [ 5 ]
Isi Queue: [ 5 2 ]
Isi Queue: [ 5 2 7 ]

Dequeue 1 Elemen
Elemen keluar: 5
Isi Queue: [ 2 7 ]

Enqueue 1 Elemen
Isi Queue: [ 2 7 4 ]

Dequeue 2 Elemen
Elemen keluar: 2
Elemen keluar: 7
Isi Queue: [ 4 ]

PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevisud\Modul8-StrukturData>
```

Deskripsi:

Program ini mengimplementasikan struktur data queue (antrian) berbasis array dengan ukuran maksimum 5 menggunakan teknik circular queue. Program menyediakan fungsi untuk membuat antrian kosong, mengecek apakah antrian penuh/kosong, menambah elemen (enqueue), menghapus elemen (dequeue), serta menampilkan isi antrian. Pada fungsi main, program mendemonstrasikan proses enqueue dan dequeue beberapa kali sambil menampilkan perubahan isi antrian setelah setiap operasi.

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

Unguided1

queue1.h

```
#ifndef QUEUE1_H
#define QUEUE1_H

typedef int infotype;

struct Queue {
    infotype info[5];
    int head, tail;
};

void createQueue(Queue &Q);

bool isEmpty(const Queue &Q);
```

```
bool isFull(const Queue &Q);  
void enqueue(Queue &Q, infotype x);  
infotype dequeue(Queue &Q);  
void printInfo(const Queue &Q);  
  
#endif
```

queue1.cpp

```
#include <iostream>  
#include "queue1.h"  
using namespace std;  
  
void createQueue(Queue &Q){  
    Q.head = -1;  
    Q.tail = -1;  
}  
  
bool isEmpty(const Queue &Q){  
    return Q.tail == -1;  
}  
  
bool isFull(const Queue &Q){  
    return Q.tail == 4;  
}  
  
// HEAD DIAM, TAIL BERGERAK  
void enqueue(Queue &Q, infotype x){  
    if(isFull(Q)) return;
```



```

if(isEmpty(Q)){
    Q.head = Q.tail = 0;
} else {
    Q.tail++;
}
Q.info[Q.tail] = x;
}

```

// DEQUEUE MENGGESER DATA KE KIRI

```

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

    infotype x = Q.info[Q.head];

    // geser semua elemen ke kiri
    for(int i = 0; i < Q.tail; i++){
        Q.info[i] = Q.info[i+1];
    }

    Q.tail--;

    if(Q.tail < 0){
        Q.head = Q.tail = -1;
    }

    return x;
}

```

// PRINT SAMA PERSIS DENGAN GAMBAR

```

void printInfo(const Queue &Q){

```

```

    if(isEmpty(Q)){
        cout << "-1 - -1 | empty queue\n";
        return;
    }

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

    for(int i = 0; i <= Q.tail; i++){
        cout << Q.info[i];
        if(i < Q.tail) cout << " ";
    }
    cout << "\n";
}

```

Main1.cpp

```

#include <iostream>

#include "queue1.h"
#include "queue1.cpp"
using namespace std;

int main(){
    cout << "Hello world!\n";
    cout << "-----\n";
    cout << "H - T : Queue 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);

        dequeue(Q); printInfo(Q);

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

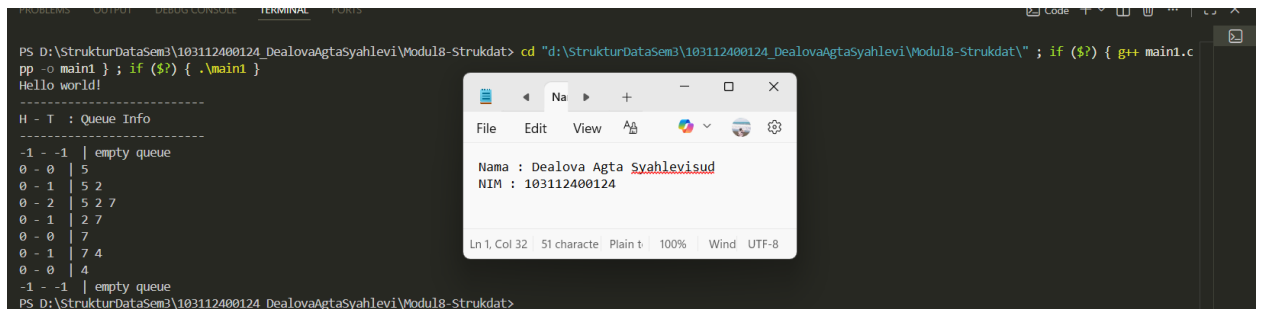
        dequeue(Q); printInfo(Q);

        dequeue(Q); printInfo(Q);

        return 0;
    }

```

Screenshots Output



```

PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat> cd "d:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat\" ; if ($?) { g++ main1.c
pp -o main1 } ; if ($?) { .\main1 }
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
PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat>

```

Deskripsi:

Program ini mengimplementasikan queue sederhana dengan ukuran tetap 5, di mana elemen ditambahkan dari belakang (tail) dan elemen yang dihapus menggeser seluruh isi array ke kiri. Queue dianggap kosong ketika head dan tail bernilai -1, serta penuh ketika tail mencapai indeks 4. Program menyediakan operasi membuat queue, mengecek kosong/penuh, menambah elemen (enqueue), menghapus elemen (dequeue), dan menampilkan isi queue beserta posisi head dan tail. Pada fungsi main, program menampilkan proses perubahan antrian setelah beberapa operasi enqueue dan dequeue secara berurutan.

Unguided2

queue2.h

```

#ifndef QUEUE2_H
#define QUEUE2_H

typedef int infotype;

struct Queue {
    infotype info[5];
    int head, tail;
};

void createQueue(Queue &Q);
bool isEmpty(const Queue &Q);
bool isFull(const Queue &Q);
void enqueue(Queue &Q, infotype x);
infotype dequeue(Queue &Q);
void printInfo(const Queue &Q);

#endif

```

Queue2.cpp

```

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

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

```

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

```
bool isFull(const Queue &Q){  
    return Q.head == 4; // karena head bergerak  
}
```

```
void enqueue(Queue &Q, infotype x){  
    if(isFull(Q)) return;  
    if(isEmpty(Q)){  
        Q.head = 0;  
        Q.info[0] = x;  
    } else {  
        Q.head++;  
        Q.info[Q.head] = x;  
    }  
}
```

```
infotype dequeue(Queue &Q){  
    if(isEmpty(Q)) return -1;  
  
    infotype x = Q.info[Q.tail];  
    Q.tail++;  
  
    if(Q.tail > Q.head){  
        Q.head = -1;  
        Q.tail = 0;  
    }  
}
```

```

        return x;
    }

    void printInfo(const Queue &Q){
        if(isEmpty(Q)){
            cout << "-1 - 0 | empty queue\n";
            return;
        }

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

```

Main2.cpp

```

#include <iostream>

#include "queue2.h"

#include "queue2.cpp"

using namespace std;

int main(){

    cout << "Hello world!\n";

    cout << "-----\n";

    cout << "H - T : Queue 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);

dequeue(Q); printInfo(Q);

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

dequeue(Q); printInfo(Q);

dequeue(Q); printInfo(Q);

return 0;

}

```

Screenshots Output

The screenshot shows a terminal window with the output of a C++ program. The output displays the state of a queue after several enqueue and dequeue operations. The queue is represented as an array of 5 elements, with the head pointer moving from -1 to 0, 1, 2, 3, and back to -1. The output shows the queue becoming empty after the final dequeue operation.

```

PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat> cd "d:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat\" ; if ($?) { g++ main2.c
pp -o main2 } ; if ($?) { .\main2 }
Hello world!

-----
H - T : Queue Info
-----
-1 - 0 | empty queue
0 - 0 | 5
1 - 0 | 5 2
2 - 0 | 5 2 7
2 - 1 | 2 7
2 - 2 | 7
3 - 2 | 7 4
3 - 3 | 4
-1 - 0 | empty queue

```

Overlaid on the terminal is a Notepad window with the following text:

```

Nama : Dealova Agta Syahlevisud
NIM : 103112400124

```

Deskripsi:

Program ini mengimplementasikan queue linier berukuran 5 di mana tail tetap di posisi 0 dan hanya head yang bergerak untuk menambah elemen baru. Queue dianggap kosong jika head = -1 dan penuh jika head mencapai indeks 4. Operasi enqueue menambah data dengan menggeser head ke kanan, sementara dequeue mengambil elemen dari posisi tail dan menaikkan tail sampai melampaui head, yang kemudian mengatur queue kembali kosong. Program menampilkan setiap perubahan isi queue melalui operasi enqueue dan dequeue secara berurutan.

Unguided3

queue3.h

```
#ifndef QUEUE3_H
#define QUEUE3_H

typedef int infotype;

struct Queue {
    infotype info[5];
    int head, tail;
};

void createQueue(Queue &Q);
bool isEmpty(const Queue &Q);
bool isFull(const Queue &Q);
void enqueue(Queue &Q, infotype x);
infotype dequeue(Queue &Q);
void printInfo(const Queue &Q);

#endif
```

Queue3.cpp

```
#include <iostream>
#include "queue3.h"
using namespace std;

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



```
}
```

```
bool isEmpty(const Queue &Q){
```

```
    return Q.head == -1;
```

```
}
```

```
bool isFull(const Queue &Q){
```

```
    return ((Q.tail + 1) % 5) == Q.head;
```

```
}
```

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

```
    if(isFull(Q)) return;
```

```
    if(isEmpty(Q)){
```

```
        Q.head = Q.tail = 0;
```

```
    } else {
```

```
        Q.tail = (Q.tail + 1) % 5;
```

```
    }
```

```
    Q.info[Q.tail] = x;
```

```
}
```

```
infotype dequeue(Queue &Q){
```

```
    if(isEmpty(Q)) return -1;
```

```
    infotype x = Q.info[Q.head];
```

```
    if(Q.head == Q.tail){
```

```
        Q.head = Q.tail = -1;
```

```
    } else {
```

```
        Q.head = (Q.head + 1) % 5;
```

```

    }

    return x;
}

void printInfo(const Queue &Q){
    if(isEmpty(Q)){
        cout << "-1 - -1 | empty queue\n";
        return;
    }

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

    int i = Q.head;
    while(true){
        cout << Q.info[i];
        if(i == Q.tail) break;
        cout << " ";
        i = (i + 1) % 5;
    }
    cout << "\n";
}

```

Main3.cpp

```

#include <iostream>

#include "queue3.h"

#include "queue3.cpp"

using namespace std;

```

```

int main(){

    cout << "Hello world!\n";

    cout << "-----\n";

    cout << "H - T : Queue 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);

    dequeue(Q); printInfo(Q);

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

    dequeue(Q); printInfo(Q);

    dequeue(Q); printInfo(Q);


    return 0;

}

```

Screenshots Output

```

{ .\main3 }
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
PS D:\StrukturDataSem3\103112400124_DealovaAgtaSyahlevi\Modul8-Strukdat>

```

Nama : Dealova Agta Syahlevi
 NIM : 103112400124

Deskripsi:

Program ini mengimplementasikan circular queue berukuran 5 yang menggunakan

pergerakan head dan tail secara melingkar (modulo). Queue dianggap kosong ketika $\text{head} = \text{tail} = -1$, dan penuh ketika $(\text{tail} + 1) \% 5 == \text{head}$. Operasi enqueue menambah elemen dengan memutar posisi tail, sedangkan dequeue menghapus elemen dari head dan mengatur queue kembali kosong jika head dan tail bertemu. Program kemudian menampilkan proses perubahan antrian melalui serangkaian operasi enqueue dan dequeue.

D. Kesimpulan

Kesimpulannya, ketiga program tersebut sama-sama mengimplementasikan konsep struktur data queue, namun dengan tiga pendekatan berbeda: queue linier dengan tail bergerak, queue linier dengan head bergerak, dan circular queue. Masing-masing memiliki cara berbeda dalam menambah dan menghapus elemen, serta dalam menentukan kondisi penuh dan kosong. Program pertama menggeser elemen saat dequeue, program kedua membiarkan tail tetap dan hanya menggerakkan head, sedangkan program ketiga paling efisien karena menggunakan pergerakan melingkar tanpa perlu penggeseran data. Meskipun mekanismenya berbeda, ketiga program menunjukkan cara kerja dasar queue: enqueue memasukkan data, dequeue menghapus data, dan hasilnya ditampilkan setelah setiap operasi.

E. Referensi

GeeksforGeeks. "C++ Programming Language – Introduction." <https://www.geeksforgeeks.org/c-plus-plus/>

Wikipedia. "C++." <https://id.wikipedia.org/wiki/C%2B%2B>