

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL VII**

**QUEUE**



**Disusun Oleh :**  
NAMA : Muhammad Fachri Auravyano Saka  
NIM : 103112430180

**Dosen**  
FAHRUDIN MUKTI WIBOWO

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

## A. Dasar Teori

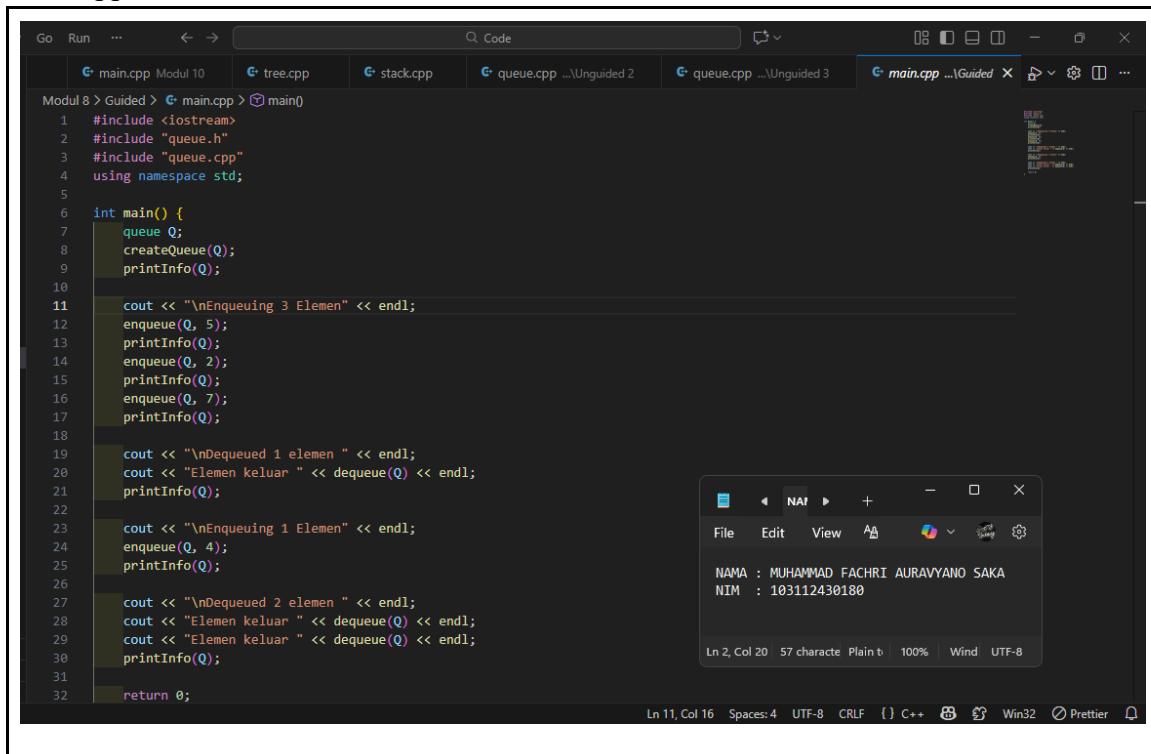
Queue atau antrean adalah struktur data yang bekerja berdasarkan prinsip FIFO (First In, First Out). Konsep ini sangat mirip dengan antrean di dunia nyata, misalnya saat mengantre membeli tiket bioskop: orang yang datang paling awal (masuk pertama) akan dilayani dan keluar paling awal juga. Sebaliknya, orang yang datang terakhir harus menunggu di posisi paling belakang.

Dalam pemrograman (khususnya C++ menggunakan Array), Queue memiliki dua pintu utama: Head (depan) untuk data keluar, dan Tail (belakang) untuk data masuk. Operasi memasukkan data disebut Enqueue, sedangkan mengeluarkan data disebut Dequeue. Tantangan utama menggunakan Array untuk Queue adalah ukuran memori yang tetap.

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

### Guided 1

#### Main.cpp



The screenshot shows a code editor with multiple tabs open. The main tab is 'main.cpp' which contains C++ code for a queue. The code includes includes for `<iostream>`, `queue.h`, and `queue.cpp`. It defines a queue `Q`, performs enqueue operations (with values 5, 2, and 7), and then dequeues elements (printing them out). The output terminal window shows the names and NIM numbers of the students.

```
Modul 8 > Guided > main.cpp > main()
1 #include <iostream>
2 #include "queue.h"
3 #include "queue.cpp"
4 using namespace std;
5
6 int main() {
7     queue Q;
8     createQueue(Q);
9     printInfo(Q);
10
11     cout << "\nEnqueuing 3 Elemen" << endl;
12     enqueue(Q, 5);
13     printInfo(Q);
14     enqueue(Q, 2);
15     printInfo(Q);
16     enqueue(Q, 7);
17     printInfo(Q);
18
19     cout << "\nDequeued 1 elemen " << endl;
20     cout << "Elemen keluar " << dequeue(Q) << endl;
21     printInfo(Q);
22
23     cout << "\nEnqueuing 1 Elemen" << endl;
24     enqueue(Q, 4);
25     printInfo(Q);
26
27     cout << "\nDequeued 2 elemen " << endl;
28     cout << "Elemen keluar " << dequeue(Q) << endl;
29     cout << "Elemen keluar " << dequeue(Q) << endl;
30     printInfo(Q);
31
32     return 0;
33 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

#### Queue.cpp

Code Editor Screenshot:

**queue.h**

```
Modul 8 > Guided > queue.cpp > printInfo(queue)
1 #include "queue.h"
2 #include <iostream>
3
4 using namespace std;
5
6 void createQueue(queue &Q) {
7     Q.head = 0;
8     Q.tail = 0;
9     Q.count = 0;
10 }
11
12 bool isEmpty(queue Q) {
13     return Q.count == 0;
14 }
15
16 bool isFull(queue Q) {
17     return Q.count == MAX_QUEUE;
18 }
19
20 void enqueue(queue &Q, int x) {
21     if (!isFull(Q)) {
22         Q.info[Q.tail] = x;
23         Q.tail = (Q.tail + 1) % MAX_QUEUE;
24         Q.count++;
25     } else {
26         cout << "Antrean Full" << endl;
27     }
28 }
29
30 int dequeue(queue &Q) {
31     if (!isEmpty(Q)) {
32         int x = Q.info[Q.head];
33         Q.head = (Q.head + 1) % MAX_QUEUE;
34         Q.count--;
35         return x;
36     } else {
37         cout << "Antrean Empty" << endl;
38         return -1; // Return a sentinel value indicating the queue is empty
39     }
40 }
41
42 void printInfo(queue Q) {
43     cout << "Isi Queue: ";
44     if (!isEmpty(Q)) {
45         int i = Q.head;
46         int n = 0;
47         while (n < Q.count) {
48             cout << Q.info[i] << " ";
49             i = (i + 1) % MAX_QUEUE;
50             n++;
51         }
52     }
53     cout << "]" << endl;
54 }
```

**queue.cpp**

```
Modul 8 > Guided > queue.cpp > printInfo(queue)
1 #include "queue.h"
2 #include <iostream>
3
4 using namespace std;
5
6 void createQueue(queue &Q) {
7     Q.head = 0;
8     Q.tail = 0;
9     Q.count = 0;
10 }
11
12 bool isEmpty(queue Q) {
13     return Q.count == 0;
14 }
15
16 bool isFull(queue Q) {
17     return Q.count == MAX_QUEUE;
18 }
19
20 void enqueue(queue &Q, int x) {
21     if (!isFull(Q)) {
22         Q.info[Q.tail] = x;
23         Q.tail = (Q.tail + 1) % MAX_QUEUE;
24         Q.count++;
25     } else {
26         cout << "Antrean Full" << endl;
27     }
28 }
29
30 int dequeue(queue &Q) {
31     if (!isEmpty(Q)) {
32         int x = Q.info[Q.head];
33         Q.head = (Q.head + 1) % MAX_QUEUE;
34         Q.count--;
35         return x;
36     } else {
37         cout << "Antrean Empty" << endl;
38         return -1; // Return a sentinel value indicating the queue is empty
39     }
40 }
41
42 void printInfo(queue Q) {
43     cout << "Isi Queue: ";
44     if (!isEmpty(Q)) {
45         int i = Q.head;
46         int n = 0;
47         while (n < Q.count) {
48             cout << Q.info[i] << " ";
49             i = (i + 1) % MAX_QUEUE;
50             n++;
51         }
52     }
53     cout << "]" << endl;
54 }
```

Terminal Output:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

Ln 49, Col 37 Spaces:4 UTF-8 CRLF {} C++ Win32 Prettier

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

Ln 49, Col 37 Spaces:4 UTF-8 CRLF {} C++ Win32 Prettier

Queue.h

The screenshot shows a code editor interface with multiple tabs. The active tab is `queue.h`, which contains the following C code:

```
Modul 8 > Guided > C queue.h ...
1 #ifndef QUEUE_H
2 #define QUEUE_H
3
4 #define MAX_QUEUE 5
5
6 struct queue
7 {
8     int info[MAX_QUEUE];
9     int head;
10    int tail;
11    int count;
12 };
13
14 void createQueue(queue &Q);
15
16 bool isEmpty(queue Q);
17
18 bool isFull(queue Q);
19
20 void enqueue(queue &Q, int x);
21
22 int dequeue(queue &Q);
23
24 void printInfo(queue Q);
25
26#endif
```

To the right of the code editor is a terminal window showing the following output:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

At the bottom of the terminal window, it says "Ln 2, Col 20 57 character Plain t 100% Wind UTF-8".

## Screenshots Output

The screenshot shows a terminal window with the following command and output:

```
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Guided" ; if ($?) { g++ main.cpp -o main } ; if ($?) { \main }
```

The output shows the execution of the program, which performs the following operations:

- Isi Queue: []
- Enqueuing 3 Elemen
- Isi Queue: [5]
- Isi Queue: [52]
- Isi Queue: [527]
- Dequeued 1 elemen
- Elemen keluar 5
- Isi Queue: [27]
- Enqueuing 1 Elemen
- Isi Queue: [274]
- Dequeued 2 elemen
- Elemen keluar 2
- Elemen keluar 7
- Isi Queue: [4]

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Guided>

To the right of the terminal window is a file viewer window showing the following text:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

At the bottom of the file viewer window, it says "Ln 2, Col 20 57 character Plain t 100% Wind UTF-8".

Deskripsi:

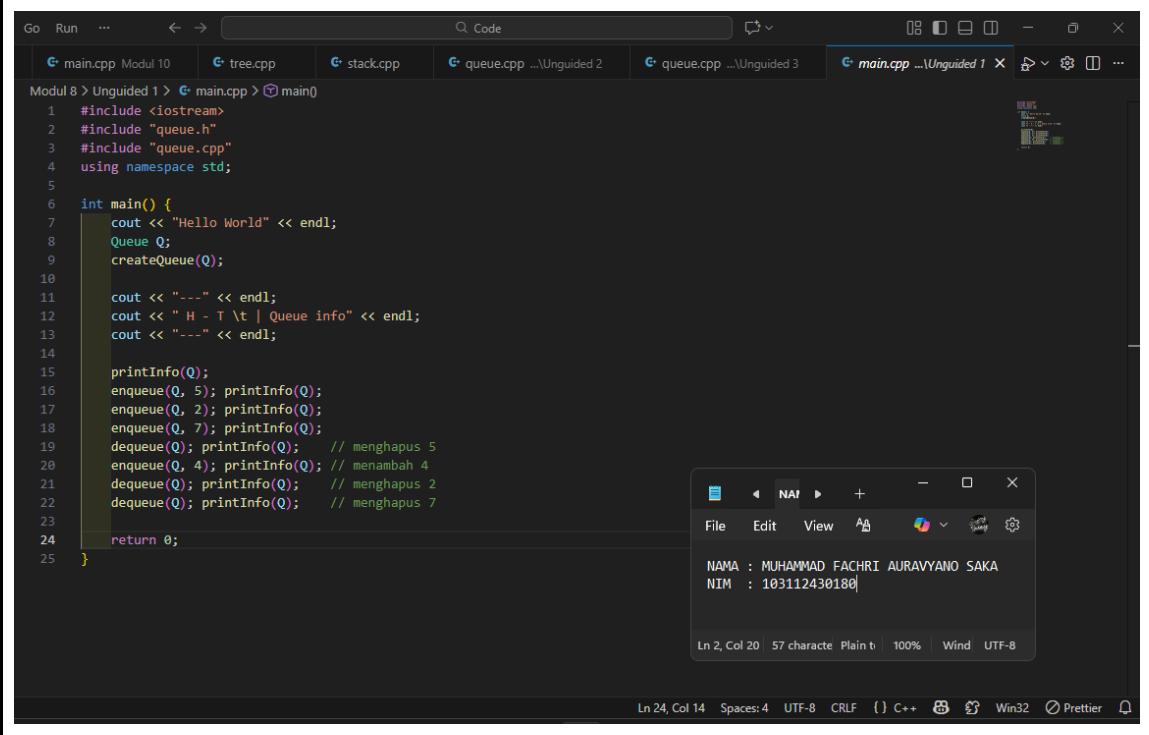
Program ini mengimplementasikan Circular Queue (Antrean Melingkar) menggunakan bantuan variabel penghitung (count). Berbeda dengan antrean biasa yang bisa "mentok" di ujung, program ini menggunakan operasi matematika modulus (%) pada bagian head dan tail. Artinya, jika antrean sudah sampai di posisi terakhir array, penunjuk akan otomatis berputar kembali ke posisi awal (indeks 0) untuk mengisi ruang kosong. Keunikan kode ini terletak pada variabel count, yang berfungsi mencatat jumlah data

secara langsung setiap kali ada yang masuk (enqueue) atau keluar (dequeue). Dengan adanya count, komputer dapat dengan sangat mudah membedakan kondisi antrean penuh atau kosong tanpa perlu menghitung jarak antara depan dan belakang secara manual. File main.cpp kemudian menjalankan simulasi sederhana: memasukkan angka (5, 2, 7), mengeluarkan satu angka, memasukkan angka baru (4), dan mengeluarkan sisanya untuk membuktikan bahwa antrean berjalan lancar secara memutar.

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

#### Unguided 1

##### Main.cpp



The screenshot shows a code editor interface with multiple tabs. The active tab is 'main.cpp' which contains the following C++ code:

```
1 #include <iostream>
2 #include "queue.h"
3 #include "queue.cpp"
4 using namespace std;
5
6 int main() {
7     cout << "Hello World" << endl;
8     Queue Q;
9     createQueue(Q);
10
11    cout << "---" << endl;
12    cout << " H - T \t | Queue info" << endl;
13    cout << "---" << endl;
14
15    printInfo(Q);
16    enqueue(Q, 5); printInfo(Q);
17    enqueue(Q, 2); printInfo(Q);
18    enqueue(Q, 7); printInfo(Q);
19    dequeue(Q); printInfo(Q); // menghapus 5
20    enqueue(Q, 4); printInfo(Q); // menambah 4
21    dequeue(Q); printInfo(Q); // menghapus 2
22    dequeue(Q); printInfo(Q); // menghapus 7
23
24    return 0;
25 }
```

To the right of the code editor, there is a terminal window showing the output of the program:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

At the bottom of the terminal window, it says 'Ln 2, Col 20 | 57 character | Plain text | 100% | Wind | UTF-8'.

At the bottom of the code editor window, it says 'Ln 24, Col 14 | Spaces: 4 | UTF-8 | CRLF | {} C++ | Win32 | Prettier'.

##### queue.cpp

Modul 8 > Unguided 1 > queue.cpp > printInfo(Queue)

```
1 #include <iostream>
2 #include "queue.h"
3 using namespace std;
4
5 void createQueue(Queue &Q) {
6     Q.head = -1;
7     Q.tail = -1;
8 }
9
10 bool isEmptyQueue(Queue Q) {
11     return (Q.head == -1 && Q.tail == -1);
12 }
13
14 bool isFullQueue(Queue Q) {
15     return (Q.tail == MAX_SIZE - 1);
16 }
17
18 void enqueue(Queue &Q, infotype x) {
19     if (isFullQueue(Q)) {
20         cout << "Queue penuh! Tidak bisa menambahkan " << x << endl;
21         return;
22     }
23
24     if (isEmptyQueue(Q)) {
25         Q.head = 0;
26         Q.tail = 0;
27     } else {
28         Q.tail++;
29     }
30
31     Q.info[Q.tail] = x;
32 }
```

File Edit View AA File Edit View AA

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8

Ln 68, Col 2 | Spaces:4 UTF-8 CRLF () C++ Win32 Prettier

Modul 8 > Unguided 1 > queue.cpp > printInfo(Queue)

```
34 infotype dequeue(Queue &Q) {
35     if (isEmptyQueue(Q)) {
36         cout << "Queue kosong! Tidak bisa menghapus elemen" << endl;
37         return -1;
38     }
39
40     infotype x = Q.info[Q.head];
41
42     // Jika hanya ada 1 elemen
43     if (Q.head == Q.tail) {
44         Q.head = -1;
45         Q.tail = -1;
46     } else {
47         // Geser semua elemen ke kiri
48         for (int i = 0; i < Q.tail; i++) {
49             Q.info[i] = Q.info[i + 1];
50         }
51         Q.tail--;
52         // Head tetap di 0 (tidak berubah)
53     }
54
55     return x;
56 }
57
58 void printInfo(Queue Q) {
59     if (isEmptyQueue(Q)) {
60         cout << Q.head << " - " << Q.tail << " \t | empty queue" << endl;
61     } else {
62         cout << Q.head << " - " << Q.tail << " \t | ";
63         for (int i = Q.head; i <= Q.tail; i++) {
64             cout << Q.info[i] << " ";
65         }
66         cout << endl;
67     }
68 }
```

File Edit View AA File Edit View AA

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8

Ln 68, Col 2 | Spaces:4 UTF-8 CRLF () C++ Win32 Prettier

57
58 void printInfo(Queue Q) {
59 if (isEmptyQueue(Q)) {
60 cout << Q.head << " - " << Q.tail << " \t | empty queue" << endl;
61 } else {
62 cout << Q.head << " - " << Q.tail << " \t | ";
63 for (int i = Q.head; i <= Q.tail; i++) {
64 cout << Q.info[i] << " ";
65 }
66 cout << endl;
67 }
68 }

File Edit View AA File Edit View AA

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8

Ln 68, Col 2 | Spaces:4 UTF-8 CRLF () C++ Win32 Prettier

queue.h

The screenshot shows a code editor interface with multiple tabs. The active tab is `queue.h`, which contains the following C++ code:

```
Modul 8 > Unguided 1 > C queue.h > ...
1 #ifndef QUEUE_H
2 #define QUEUE_H
3
4 const int MAX_SIZE = 5;
5
6 typedef int infotype;
7
8 struct Queue {
9     infotype info[MAX_SIZE];
10    int head;
11    int tail;
12 };
13
14 // Prototype functions
15 void createQueue(Queue &Q);
16 bool isEmptyQueue(Queue Q);
17 bool isFullQueue(Queue Q);
18 void enqueue(Queue &Q, infotype x);
19 infotype dequeue(Queue &Q);
20 void printInfo(Queue Q);
21
22 #endif
```

To the right of the code editor is a terminal window displaying the following output:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

## Screenshot Output

The screenshot shows a terminal window with the following command and output:

```
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 2> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 1\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { ./main }
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
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 1> []
```

To the right of the terminal window is a small text box containing the student's information:

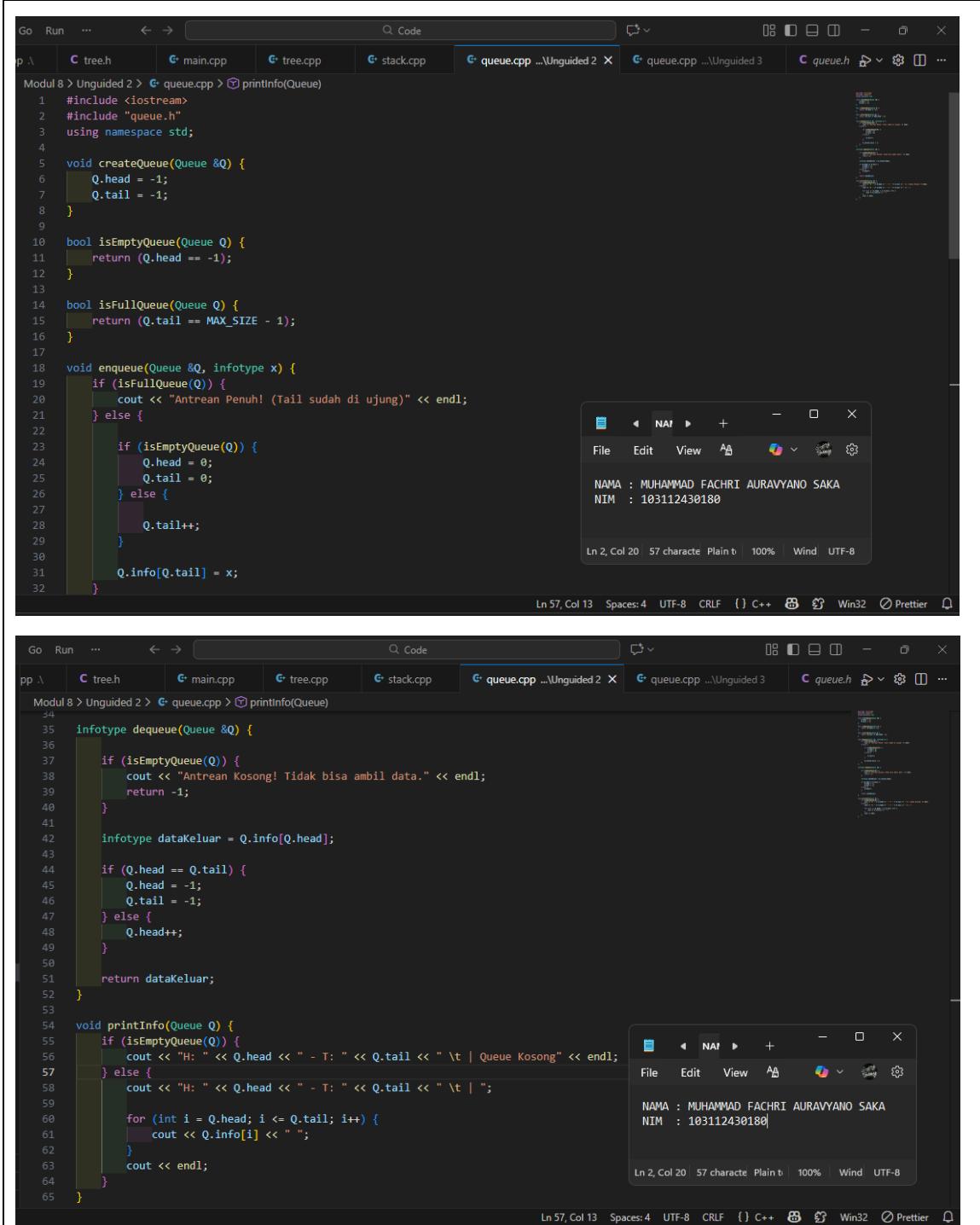
```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

## Deskripsi:

Pada implementasi ini, posisi Head (depan) selalu dikunci di indeks ke-0. Bayangkan antrean orang di kasir; ketika orang paling depan selesai dilayani dan pergi, semua orang di belakangnya harus melangkah maju satu langkah mengisi tempat yang kosong. Dalam bahasa pemrograman, ini berarti setiap kali kita melakukan dequeue (mengeluarkan data),

komputer harus bekerja ekstra untuk menggeser semua data yang tersisa dari indeks belakang ke indeks depannya menggunakan perulangan (looping). Cara ini mudah dibayangkan, tapi kurang efisien karena membebani komputer untuk terus-menerus menggeser data.

## Unguided 2



The screenshot shows a code editor interface with two windows. The main window displays the following C++ code for a queue implementation:

```
Modul 8 > Unguided 2 > queue.cpp > printInfo(Queue)
1 #include <iostream>
2 #include "queue.h"
3 using namespace std;
4
5 void createQueue(Queue &Q) {
6     Q.head = -1;
7     Q.tail = -1;
8 }
9
10 bool isEmptyQueue(Queue Q) {
11     return (Q.head == -1);
12 }
13
14 bool isFullQueue(Queue Q) {
15     return (Q.tail == MAX_SIZE - 1);
16 }
17
18 void enqueue(Queue &Q, infotype x) {
19     if (isFullQueue(Q)) {
20         cout << "Antrean Penuh! (Tail sudah di ujung)" << endl;
21     } else {
22
23         if (isEmptyQueue(Q)) {
24             Q.head = 0;
25             Q.tail = 0;
26         } else {
27
28             Q.tail++;
29         }
30
31         Q.info[Q.tail] = x;
32     }
}
```

The second window shows a terminal or console output with the following information:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

Below the code editor, status bars indicate the line and column numbers (Ln 2, Col 20), character count (57), file type (Plain t), and encoding (UTF-8).

```
Modul 8 > Unguided 2 > queue.cpp > printInfo(Queue)
34
35     infotype dequeue(Queue &Q) {
36
37         if (isEmptyQueue(Q)) {
38             cout << "Antrean Kosong! Tidak bisa ambil data." << endl;
39             return -1;
40         }
41
42         infotype dataKeluar = Q.info[Q.head];
43
44         if (Q.head == Q.tail) {
45             Q.head = -1;
46             Q.tail = -1;
47         } else {
48             Q.head++;
49         }
50
51         return dataKeluar;
52     }
53
54     void printInfo(Queue Q) {
55         if (isEmptyQueue(Q)) {
56             cout << "H: " << Q.head << " - T: " << Q.tail << " \t | Queue Kosong" << endl;
57         } else {
58             cout << "H: " << Q.head << " - T: " << Q.tail << " \t | ";
59
60             for (int i = Q.head; i <= Q.tail; i++) {
61                 cout << Q.info[i] << " ";
62             }
63             cout << endl;
64         }
65     }
}
```

The second window shows a terminal or console output with the following information:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

Below the code editor, status bars indicate the line and column numbers (Ln 2, Col 20), character count (57), file type (Plain t), and encoding (UTF-8).

Screenshot Output

```

n } ; if ($?) { .\main }
Hello World
---
H - T | Queue info
---
H: -1 - T: -1 | Queue Kosong
H: 0 - T: 0 | 5
H: 0 - T: 1 | 5 2
H: 0 - T: 2 | 5 2 7
H: 1 - T: 2 | 2 7
H: 1 - T: 3 | 2 7 4
H: 2 - T: 3 | 7 4
H: 3 - T: 3 | 4
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 2> []

```

Deskripsi:

Implementasi ini mencoba mengatasi kelemahan metode pertama agar komputer tidak perlu repot menggeser data. Di sini, Head (depan) dan Tail (belakang) sama-sama bergerak maju. Bayangkan antrean kursi memanjang; ketika orang di kursi nomor 1 pergi, kasir yang pindah melayani orang di kursi nomor 2. Orang-orang dalam antrean tidak perlu bergeser. Prosesnya jadi sangat cepat. Namun, kelemahannya adalah terjadinya "Penuh Semu". Lama-kelamaan, antrean akan mentok di ujung belakang kursi, padahal kursi-kursi di depan yang sudah ditinggalkan kosong dan tidak bisa dipakai lagi (kecuali antrean di-reset total saat kosong).

### Unguided 3

```

Go Run ... ← → Q Code
main.cpp Modul 10 tree.cpp stack.cpp queue.cpp ... Unguided 2 main.cpp ... Unguided 3 queue.cpp ... Unguided 3
Module 8 > Unguided 3 > queue.cpp > printInfo(Queue)
1 #include <iostream>
2 #include "queue.h"
3 using namespace std;
4
5 void createQueue(Queue &Q) {
6     Q.head = -1;
7     Q.tail = -1;
8 }
9
10 bool isEmptyQueue(Queue Q) {
11     return (Q.head == -1);
12 }
13
14 bool isFullQueue(Queue Q) {
15     int nextTail = (Q.tail + 1) % MAX_SIZE;
16     return (nextTail == Q.head);
17 }
18
19 void enqueue(Queue &Q, infotype x) {
20     if (isFullQueue(Q)) {
21         cout << "Antrean Penuh! Data " << x << " gagal masuk." << endl;
22     } else {
23
24         if (isEmptyQueue(Q)) {
25             Q.head = 0;
26             Q.tail = 0;
27         }
28
29         else if (Q.tail == MAX_SIZE - 1) {
30             Q.tail = 0;
31         }
32     }
}

```

```
Module 8 > Unguided 3 > queue.cpp > printInfo(Queue)
19     void enqueue(Queue &Q, infotype x) {
20     } else {
21         else {
22             Q.tail++;
23         }
24         Q.info[Q.tail] = x;
25     }
26 }
27 infotype dequeue(Queue &Q) {
28     if (isEmptyQueue(Q)) {
29         cout << "Antrian Kosong!" << endl;
30         return -1;
31     }
32
33     infotype dataKeluar = Q.info[Q.head];
34     if (Q.head == Q.tail) {
35         Q.head = -1;
36         Q.tail = -1;
37     }
38
39     else if (Q.head == MAX_SIZE - 1) {
40         Q.head = 0;
41     }
42
43     else {
44         Q.head++;
45     }
46
47     return dataKeluar;
48 }
```

```
Module 8 > Unguided 3 > queue.cpp > printInfo(Queue)
40     infotype dequeue(Queue &Q) {
41
42     else {
43         Q.head++;
44     }
45
46     return dataKeluar;
47 }
48
49 void printInfo(Queue Q) {
50     if (isEmptyQueue(Q)) {
51         cout << "H: " << Q.head << " - T: " << Q.tail << " \t | Queue Kosong" << endl;
52     } else {
53         cout << "H: " << Q.head << " - T: " << Q.tail << " \t | ";
54         int i = Q.head;
55         while (true) {
56             cout << Q.info[i] << " ";
57
58             if (i == Q.tail) break;
59
60             if (i == MAX_SIZE - 1) {
61                 i = 0;
62             } else {
63                 i++;
64             }
65         }
66         cout << endl;
67     }
68 }
```

Screenshot Output

```

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 2> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 3">; if ($?) { g++ main.cpp -o main }
Hello World
---
H - T | Queue info
---
H: -1 - T: -1 | Queue Kosong
H: 0 - T: 0 | 5
H: 0 - T: 1 | 5 2
H: 0 - T: 2 | 5 2 7
H: 1 - T: 2 | 2 7
H: 1 - T: 3 | 2 7 4
H: 2 - T: 3 | 7 4
H: 3 - T: 3 | 4
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 8\Unguided 3> []

```

Deskripsi:

Ini adalah metode yang paling cerdik dan efisien. Konsepnya mirip dengan Alternatif 2, tetapi ujung belakang antrean disambungkan kembali ke ujung depan, seolah-olah antrean berbentuk lingkaran. Jika posisi Tail sudah sampai di ujung array (indeks terakhir) tetapi di bagian depan (indeks 0, 1, dst.) masih ada kursi kosong yang ditinggalkan, maka data baru akan berputar dan masuk ke kursi kosong di depan tersebut. Dengan cara ini, tidak ada memori yang terbuang sia-sia dan komputer tidak perlu lelah menggeser-geser data.

#### D. Kesimpulan

Berdasarkan praktikum Modul 8 tentang Queue yang telah dilakukan, kita dapat menyimpulkan beberapa hal penting:

1. Pemahaman Prinsip FIFO: Praktikum ini membuktikan bahwa Queue sukses menerapkan prinsip FIFO. Data yang pertama kali dimasukkan (enqueue) terbukti menjadi data yang pertama kali keluar (dequeue) saat program dijalankan.
2. Perbandingan Metode Implementasi: Kita telah mencoba tiga mekanisme berbeda dan melihat kelebihan serta kekurangannya:
  - Alternatif 1 (Geser): Metode ini boros kinerja karena komputer harus melakukan perulangan (*looping*) untuk menggeser data setiap kali ada pengambilan data.
  - Alternatif 2 (Pointer Maju): Metode ini cepat, tetapi boros tempat. Antrean sering dianggap penuh (*Antrean Penuh/Full*) padahal indeks awal array (0, 1, dst) sebenarnya kosong tak terpakai.
  - Alternatif 3 (Circular/Melingkar): Ini adalah metode yang paling efisien.

Dengan menggunakan logika aritmatika modulus (%) atau reset indeks, kita bisa memanfaatkan seluruh kapasitas array tanpa ada ruang kosong yang terbuang sia-sia.

3. Penggunaan Variabel Count: Pada latihan *Guided*, penggunaan variabel tambahan count sangat membantu logika program. Dengan count, kita bisa mengetahui apakah antrean penuh atau kosong secara instan tanpa harus membandingkan posisi Head dan Tail yang rumit. Secara keseluruhan, Circular Queue adalah teknik terbaik untuk menangani antrean dengan alokasi memori tetap (Array).

#### E. Referensi

- Muñoz, D. F. (2024, June). A C++ library for fast simulation of queues and some experimental results. In *AIP Conference Proceedings* (Vol. 3094, No. 1, p. 110002). AIP Publishing LLC.
- Goponenko, A., & Carroll, S. (2019). A C++ implementation of a lock-free priority queue based on Multi-Dimensional Linked List. *Link:* [https://www.researchgate.net/publication/337020321\\_A\\_C\\_Implementation\\_of\\_a\\_Lock-Free\\_Priority\\_Queue\\_Based\\_on\\_Multi-Dimensional\\_Linked\\_List](https://www.researchgate.net/publication/337020321_A_C_Implementation_of_a_Lock-Free_Priority_Queue_Based_on_Multi-Dimensional_Linked_List).
- Malik, D. S. (2010). *Data structures using C++*. USA.