

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL VIII

QUEUE



Disusun Oleh :

Nama : Besthian Guido Rafael Simbolon
NIM : 103112430258

Dosen

FAHRUDIN MUKTI WIBOWO

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

A. Dasar Teori

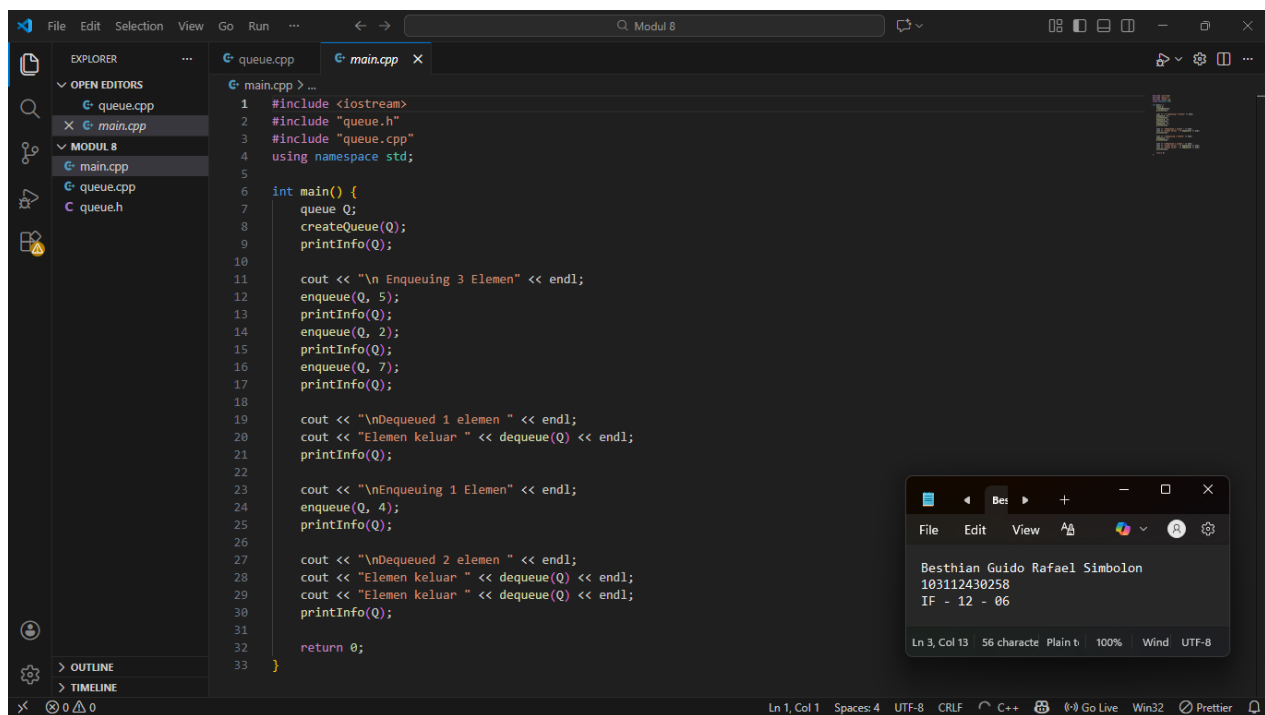
Queue adalah salah satu struktur data linier yang mengatur aliran data berdasarkan urutan kedatangan, sehingga elemen yang lebih dulu dimasukkan akan diproses lebih dulu dibandingkan elemen yang datang belakangan. Pola kerjanya dapat dianalogikan seperti barisan kendaraan di lampu merah, di mana kendaraan paling depan akan bergerak lebih dulu saat lampu hijau menyala.

Dalam penerapannya menggunakan array pada bahasa C++, queue memiliki dua penunjuk posisi, yaitu bagian depan sebagai lokasi pengambilan data dan bagian belakang sebagai tempat penambahan data baru. Penambahan elemen ke dalam queue dilakukan melalui operasi enqueue, sedangkan penghapusan elemen dilakukan melalui operasi dequeue. Karena menggunakan array, ruang penyimpanan queue bersifat terbatas dan tidak dapat bertambah secara dinamis, sehingga pengelolaan indeks harus dilakukan dengan hati-hati agar tidak melebihi kapasitas yang tersedia.

B. Guided

Guided 1

Main.cpp



```
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 << "\n Enqueuing 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 }
```

Queue.cpp

The screenshot shows the Visual Studio Code editor with the file `queue.cpp` open. The code implements a queue with the following functions:

- `createQueue(queue &Q)`: Initializes the queue with `Q.head = 0`, `Q.tail = 0`, and `Q.count = 0`.
- `isEmpty(queue Q)`: Returns `Q.count == 0`.
- `isFull(queue Q)`: Returns `Q.count == MAX_QUEUE`.
- `enqueue(queue &Q, int x)`: Adds an element `x` to the queue if it is not full. It updates `Q.info[Q.tail] = x`, `Q.tail = (Q.tail + 1) % MAX_QUEUE`, and `Q.count++`. If the queue is full, it prints "Antrean Full".
- `dequeue(queue &Q)`: Removes an element from the queue if it is not empty. It updates `Q.head = (Q.head + 1) % MAX_QUEUE` and `Q.count--`.
- `printInfo(queue Q)`: Prints the queue's state.

The status bar at the bottom indicates the cursor is at line 48, column 36, with 4 spaces, UTF-8 encoding, and C++ language.

Queue.h

The screenshot shows the Visual Studio Code editor with the file `queue.h` open. The header file defines the queue structure and declares the functions:

```
#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
```

The status bar at the bottom indicates the cursor is at line 1, column 1, with 4 spaces, UTF-8 encoding, and C++ language.

Screenshots Output



```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Dequeued 1 elemen
Elemen keluar 5
Isi Queue: [ 2 7 ]

Enqueuing 1 Elemen
Isi Queue: [ 2 7 4 ]

Dequeued 2 elemen
Elemen keluar 2
Elemen keluar 7
Isi Queue: [ 4 ]
PS C:\StrukDat\Modul 8>
```

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

Ln 3, Col 13 56 character Plain b 100% Wind UTF-8

Deskripsi:

Program ini merupakan contoh penerapan **circular queue** yang memanfaatkan sebuah variabel penghitung bernama *count* untuk mengelola data di dalam antrean. Tidak seperti queue biasa yang bisa berhenti saat penunjuk mencapai ujung array, antrean ini dirancang agar penunjuk **head** dan **tail** dapat berputar kembali ke awal array dengan bantuan operasi modulus (%), sehingga ruang kosong yang sudah ditinggalkan dapat digunakan kembali. Variabel *count* berperan penting karena menyimpan jumlah elemen yang sedang ada di dalam queue, nilainya akan bertambah saat proses *enqueue* dan berkurang saat proses *dequeue*. Dengan cara ini, program dapat dengan mudah mengetahui apakah antrean dalam keadaan penuh atau kosong tanpa perlu perhitungan tambahan yang rumit. Pada file *main.cpp*, dilakukan simulasi sederhana dengan memasukkan beberapa angka, menghapus satu elemen, lalu menambahkan elemen baru dan mengeluarkan sisanya, yang menunjukkan bahwa mekanisme antrean melingkar berjalan dengan baik dan konsisten.

C. Unguided

Unguided 1

Main.cpp

The screenshot shows the Visual Studio Code editor with the file `main.cpp` open. The code implements a simple queue using an array. The `main` function demonstrates enqueue and dequeue operations. A user profile overlay is visible in the bottom right corner.

```
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 }
```

User Profile Overlay:

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 13 56 caractere Plain t 100% Wind UTF-8
```

Queue.cpp

The screenshot shows the Visual Studio Code editor with the file `queue.cpp` open. It contains the implementation of the queue functions: `createQueue`, `isEmptyQueue`, `isFullQueue`, `enqueue`, and `dequeue`. A user profile overlay is visible in the bottom right corner.

```
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 }
33
34 infotype dequeue(Queue &Q) {
35     if (isEmptyQueue(Q)) {
36         cout << "Queue kosong! Tidak bisa menghapus" << endl;
37         return 0;
38     }
39
40     infotype x = Q.info[Q.head];
41     Q.head++;
42     return x;
43 }
```

User Profile Overlay:

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 13 56 caractere Plain t 100% Wind UTF-8
```

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
```

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

Ln 3, Col 13 56 character Plain t 100% Wind UTF-8

Screenshot Output

```
PS C:\StrukDat\Modul 8> cd "C:\StrukDat\Modul 8\unguided1\" ; if ($?) { g++ main.cpp -o main } ; if ($?) {
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 C:\StrukDat\Modul 8\unguided1>
```

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

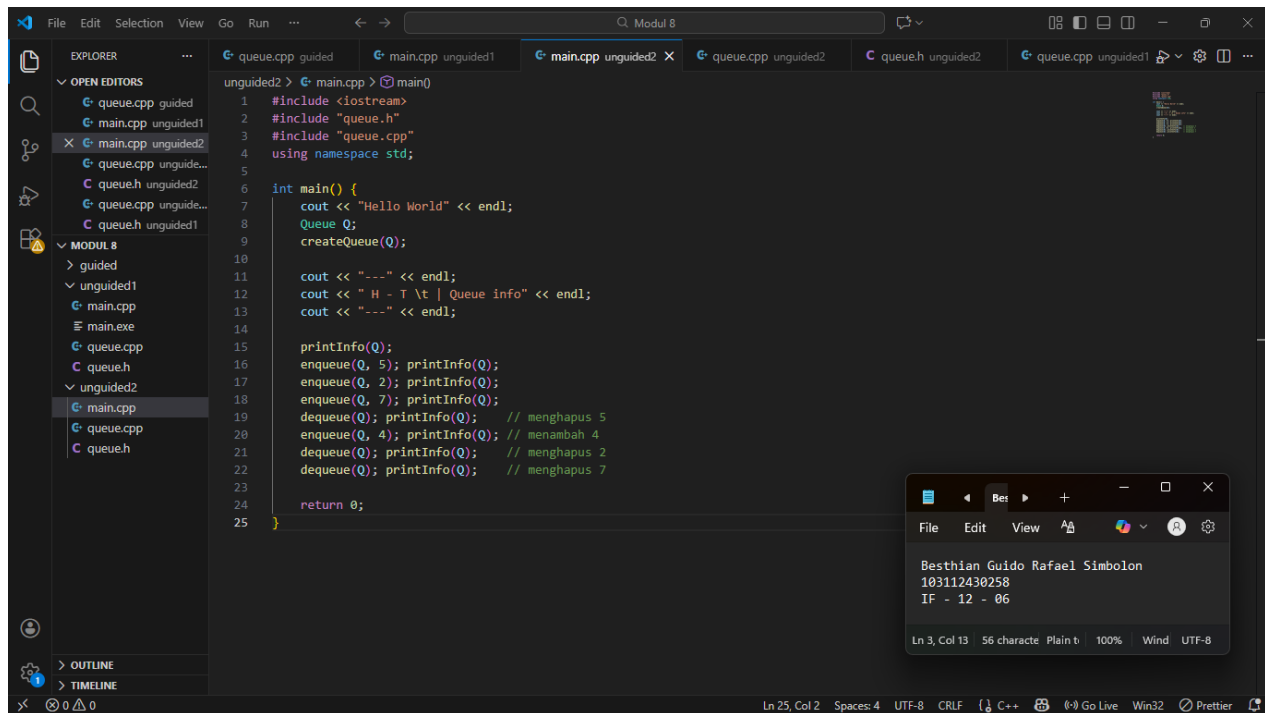
Ln 3, Col 13 56 character Plain t 100% Wind UTF-8

Deskripsi:

Pada program ini, posisi elemen terdepan antrean selalu berada di indeks pertama (indeks 0). Jika dianalogikan dengan antrean di kasir, saat orang paling depan selesai dilayani dan keluar dari antrean, orang-orang di belakangnya harus maju ke depan satu per satu. Hal yang sama terjadi di program: ketika data dikeluarkan dari antrean (dequeue), sistem harus memindahkan semua data yang tersisa agar tetap berurutan mulai dari indeks awal. Proses pemindahan ini dilakukan menggunakan perulangan, sehingga membutuhkan waktu dan kerja tambahan dari komputer. Meskipun konsepnya sederhana dan mudah dipahami, cara ini kurang efisien jika jumlah data dalam antrean cukup banyak..

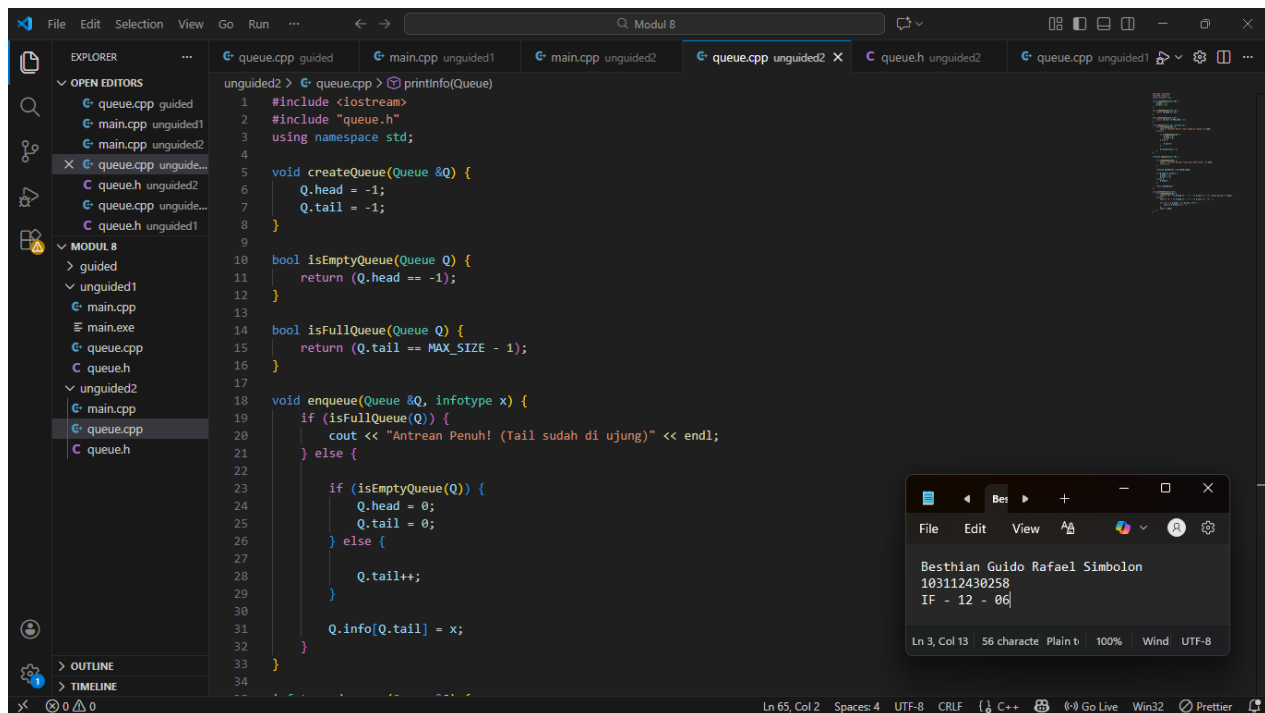
Unguide 2

Main.cpp



```
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 }
```

Queue.cpp



```
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, int type x) {
19     if (isFullQueue(Q)) {
20         cout << "Antrean Penuh! (Tail sudah di ujung)" << endl;
21     } else {
22         if (isEmptyQueue(Q)) {
23             Q.head = 0;
24             Q.tail = 0;
25         } else {
26             Q.tail++;
27         }
28         Q.info[Q.tail] = x;
29     }
30 }
31
32 }
```

Queue.h

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'queue.h' file is open, displaying the following code:

```
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
```

A small window titled 'Best' is overlaid on the right side of the editor, displaying the text:

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

ScreenShoot Output :

The screenshot shows the terminal output of the program. The command executed is `cd "c:\StrukDat\Modul 8\unguided2\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { ./main }`. The output is as follows:

```
PS C:\StrukDat\Modul 8\unguided2> cd "c:\StrukDat\Modul 8\unguided2\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { ./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 C:\StrukDat\Modul 8\unguided2>
```

A small window titled 'Best' is overlaid on the right side of the terminal, displaying the same text as in the first screenshot:

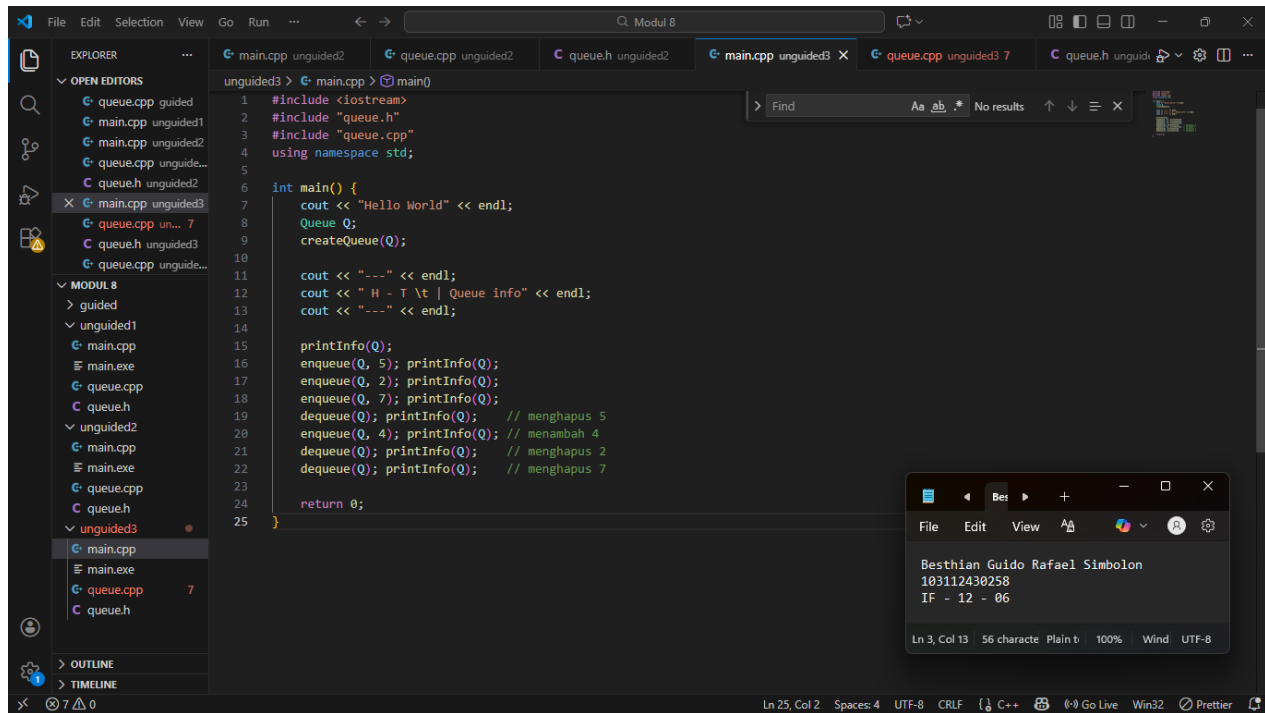
```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

Deskripsi:

Implementasi ini dibuat untuk menghindari proses penggeseran data seperti pada metode sebelumnya, sehingga posisi depan (head) dan belakang (tail) antrian sama-sama bergerak maju. Jika dianalogikan dengan deretan kursi, ketika orang di kursi paling depan pergi, petugas cukup melayani orang di kursi berikutnya tanpa membuat semua orang bergeser. Cara ini membuat proses antrian menjadi lebih cepat dan efisien, namun memiliki kelemahan berupa kondisi *penuh semu*, yaitu saat antrian terlihat penuh karena posisi tail sudah mencapai batas akhir, padahal masih ada ruang kosong di bagian depan yang tidak dapat digunakan kembali kecuali antrian dikosongkan dan diatur ulang.

Unguided 3

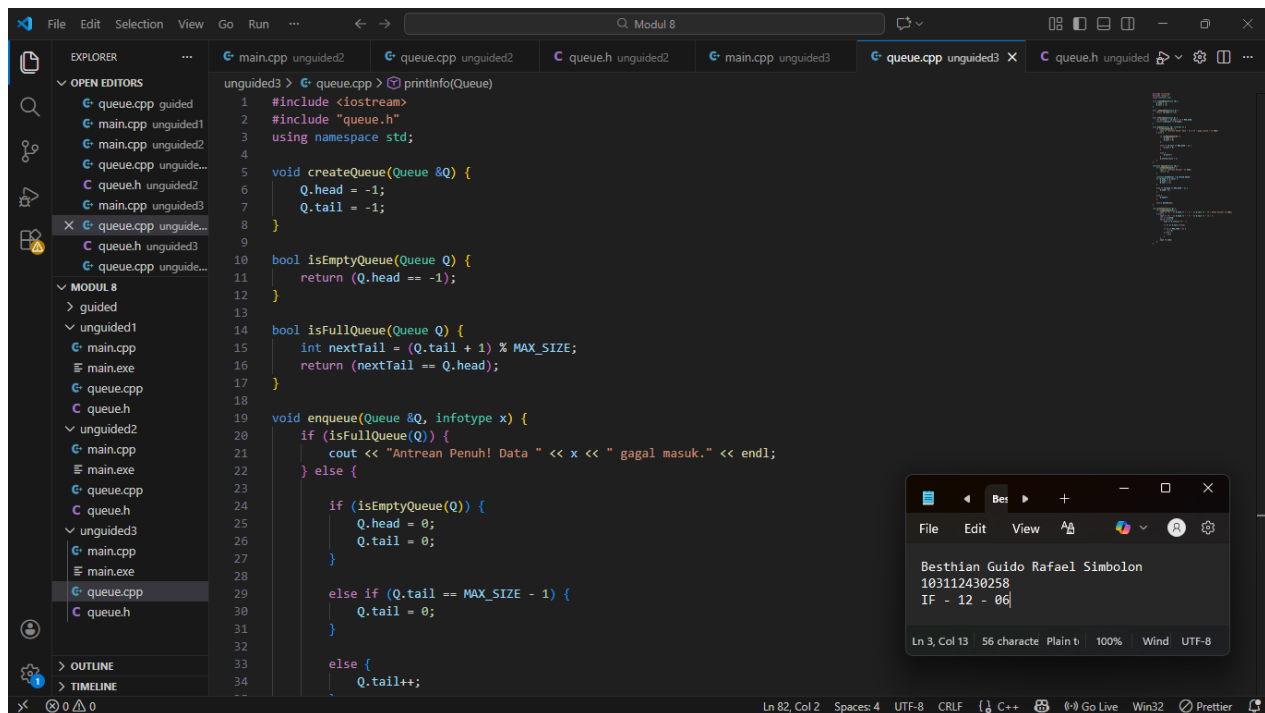
Main.cpp



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'OPEN EDITORS' list shows several files, including 'main.cpp' in 'unguided3'. The main editor window displays the code for 'main.cpp' in 'unguided3'. The code includes headers for `<iostream>`, `"queue.h"`, and `"queue.cpp"`, and uses the `std` namespace. The `main` function prints "Hello World", then "H - T \t | Queue info", and then a series of enqueue and dequeue operations with corresponding print statements. A small window titled 'Best' is open in the bottom right corner, displaying the text 'Besthian Guido Rafael Simbolon', '103112430258', and 'IF - 12 - 06'.

```
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 }
```

Queue.cpp



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'OPEN EDITORS' list shows several files, including 'queue.cpp' in 'unguided3'. The main editor window displays the code for 'queue.cpp' in 'unguided3'. The code includes headers for `<iostream>` and `"queue.h"`, and uses the `std` namespace. It defines a `Queue` struct with `head` and `tail` pointers. Functions `createQueue`, `isEmptyQueue`, `isFullQueue`, `enqueue`, and `dequeue` are implemented. The `enqueue` function checks if the queue is full and handles the wrap-around of the `tail` pointer. A small window titled 'Best' is open in the bottom right corner, displaying the text 'Besthian Guido Rafael Simbolon', '103112430258', and 'IF - 12 - 06'.

```
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, int type x) {
20     if (isFullQueue(Q)) {
21         cout << "Antrean Penuh! Data " << x << " gagal masuk." << endl;
22     } else {
23         if (isEmptyQueue(Q)) {
24             Q.head = 0;
25             Q.tail = 0;
26         }
27         else if (Q.tail == MAX_SIZE - 1) {
28             Q.tail = 0;
29         }
30         else {
31             Q.tail++;
32         }
33     }
34 }
```

Queue.h

The screenshot shows the Visual Studio Code editor with a C++ project. The Explorer sidebar on the left shows the file structure with folders for 'guided', 'unguided1', 'unguided2', and 'unguided3'. The 'queue.h' file in the 'unguided3' folder is selected and open in the editor. The code defines a Queue structure and its operations. A small window titled 'Best' is overlaid on the editor, displaying the text: 'Besthian Guido Rafael Simbolon', '103112430258', and 'IF - 12 - 06'.

```
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
```

Screenshot Output

The screenshot shows the terminal window of Visual Studio Code. The command prompt shows the execution of the program: 'PS C:\StrukDat\Modul 8\unguided2> cd "c:\StrukDat\Modul 8\unguided3\" ; if (\$?) { g++ main.cpp -o main }'. The output shows 'Hello World' followed by a table of queue operations. A small window titled 'Best' is overlaid on the terminal, displaying the same text as in the first screenshot: 'Besthian Guido Rafael Simbolon', '103112430258', and 'IF - 12 - 06'.

```
PS C:\StrukDat\Modul 8\unguided2> cd "c:\StrukDat\Modul 8\unguided3\" ; 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 C:\StrukDat\Modul 8\unguided3>
```

Deskripsi:

Metode ini merupakan pendekatan yang paling efisien karena menghubungkan bagian belakang antrean dengan bagian depannya sehingga membentuk antrean melingkar. Konsepnya mirip dengan metode sebelumnya, tetapi ketika posisi tail sudah mencapai indeks terakhir dan masih terdapat ruang kosong di bagian awal array, data baru akan kembali mengisi posisi kosong tersebut. Dengan mekanisme ini, seluruh ruang memori dapat dimanfaatkan secara maksimal tanpa perlu menggeser data, sehingga proses antrean menjadi lebih cepat dan efektif.

D. Kesimpulan

Berdasarkan praktikum Modul 8 mengenai **Queue** yang telah dilaksanakan, dapat ditarik beberapa kesimpulan penting. Praktikum ini menunjukkan bahwa struktur data Queue bekerja sesuai dengan konsep **FIFO (First In First Out)**, di mana elemen yang pertama kali dimasukkan ke dalam antrean akan menjadi elemen pertama yang dikeluarkan.

Selain itu, telah diuji tiga metode implementasi Queue yang memiliki karakteristik berbeda. Metode geser memerlukan proses pemindahan data setiap kali terjadi penghapusan elemen sehingga kurang efisien dari sisi kinerja. Metode pointer maju menawarkan proses yang lebih cepat, namun menyebabkan pemborosan ruang memori karena sebagian indeks array tidak dimanfaatkan meskipun antrean sudah dianggap penuh. Sementara itu, metode

Circular Queue menjadi solusi paling optimal karena memungkinkan penggunaan seluruh kapasitas array secara maksimal dengan menerapkan konsep indeks melingkar menggunakan operasi modulus atau pengaturan ulang indeks.

Penggunaan variabel tambahan berupa `count` juga terbukti mempermudah proses pengecekan kondisi antrean, baik dalam keadaan kosong maupun penuh, tanpa harus membandingkan posisi *head* dan *tail*. Oleh karena itu, dapat disimpulkan bahwa **Circular Queue** merupakan pendekatan yang paling efektif untuk pengelolaan antrean dengan alokasi memori tetap berbasis array.

E. Referensi

- Modul Praktikum Struktur Data. (2023). *Queue dan Variasinya*. Program Studi Teknik Informatika, Fakultas Ilmu Komputer.
- Munir, R. (2010). *Algoritma dan Pemrograman*. Bandung: Informatika Bandung.
- Nugroho, B. (2017). *Dasar Pemrograman dan Struktur Data*. Yogyakarta: Gava Media.
- Suryadi, & Rahayu, S. (2018). *Struktur Data*. Jakarta: Erlangga.