LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VIII
QUEUE

Disusun Oleh :
NAMA : Muhamad Naufal Ammar
NIM : 103112430036

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Dalam lima tahun terakhir penelitian tentang circular queue dan circular buffer menekankan bahwa penggunaan indeks yang berputar (modulo) memungkinkan pemanfaatan memori tetap secara lebih efisien dibandingkan antrian array linear karena menghilangkan kebutuhan shifting saat dequeue, sehingga menurunkan overhead waktu dan operasi salin data terutama pada sistem waktu-nyata dan embedded. Beberapa studi juga menunjukkan desain FIFO asinkron dan teknik slice/circular buffer meningkatkan efisiensi latensi dan konsumsi daya untuk aplikasi jaringan dan pengolahan sinyal, serta memberi keuntungan pada pemrosesan paralel dan manajemen memori pada router/SoC. Temuan ini konsisten di berbagai laporan konferensi dan studi perbandingan yang menguji buffer tradisional versus buffer melingkar dalam skenario nyata.

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

Guided 1

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

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

//prosedur untuk mebuat queue kosong
void createQueue(Queue &Q);
//fungsi untuk mengecek apakah queue kosong
bool isEmpty(const Queue Q);
//fungsi untuk mengecek apakah queue penuh
bool isFull(const Queue Q);
//prosedur untuk menambah elemen ke dalam queue(enqueue)
void enqueue(Queue &Q, int x);
//fungsi untuk menghapus elemen dari dalam queue(dequeue)
int dequeue(Queue &Q);
//prosedur untuk menampilkan isi queue
void printInfo(Queue Q);
#endif
```

*C:\Program Files\Not...

File Edit Search View Encoding Language
Settings Tools Macro Run Plugins
Window ?

change.log

```
1  Nama : Muhamad Naufal Ammar
2  NIM : 103112430036
3  Kelas 12-IF-03
```

Ln: Windows (CR LF) UTF-8 INS

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

using namespace std;

//definisi prosedur dan fungsi queue
void createQueue(Queue &Q){
    Q.head = 0; //set kepala indeks menjadi 0
    Q.tail = 0; //set ekor indeks menjadi 0
    Q.count = 0; //set jumlah elemen menjadi 0
}

//definisi fungsi untuk mengecek apakah queue kosong
bool isEmpty(const Queue Q){
    return (Q.count == 0); //mengembalikan true jika jumlah elemen 0
}

//efinisi fungsi untuk mengecek apakah queue penuh
bool isFull(const Queue Q){
    return (Q.count == MAX_QUEUE); //mengembalikan true jika jumlah elemen sama dengan kapasitas maksimum
}

//definisi prosedur untuk menambah elemen ke dalam queue(enqueue)
void enqueue(Queue &Q, int x){
    if (!isFull(Q)){ //cek apakah queue penuh
        Q.info[Q.tail] = x; //menambahkan elemen pada posisi ekor
        Q.tail = (Q.tail + 1) % MAX_QUEUE; //mengupdate indeks ekor secara melingkar
        Q.count++; //menambah jumlah elemen
    } else {
        cout << "Antrean Penuh" << endl; //pesan jika queue penuh
    }
}
//definisi fungsi untuk menghapus elemen dari dalam queue(dequeue)
int dequeue(Queue &Q){
    if (!isEmpty(Q)){ //cek apakah queue kosong
        int x = Q.info[Q.head]; //mengambil elemen pada posisi kepala
        Q.head = (Q.head + 1) % MAX_QUEUE; //mengupdate indeks kepala secara melingkar
        Q.count--; //mengurangi jumlah elemen
        return x; //mengembalikan elemen yang dihapus
    } else {
        cout << "Antrean Kosong" << endl; //pesan jika queue kosong
        return -1; //mengembalikan nilai -1 sebagai indikasi queue kosong
    }
}
//definisi prosedur untuk menampilkan isi queue
void printInfo(Queue Q){
    cout << "Isi Antrean: ["; // tampilkan awalan
    if (!isEmpty(Q)){ //cek apakah queue kosong
        int i = Q.head; //mulai dari indeks kepala
        int n = 0; //inisialisasi penghitung elemen yang ditampilkan
        while (n < Q.count){ //loop sebanyak jumlah elemen
            cout << Q.info[i] << " "; //tampilkan elemen pada indeks i
            i = (i + 1) % MAX_QUEUE; //update indeks i secara melingkar
            n++; //increment penghitung elemen
        }
        cout <<  "]" << endl; //tampilkan akhiran
    }
}
```

*C:\Program Files\Not...

File Edit Search View Encoding Language
Settings Tools Macro Run Plugins
Window ?

change.log

```
1  Nama : Muhamad Naufal Ammar
2  NIM : 103112430036
3  Kelas 12-IF-03
```

Ln: Windows (CR LF) UTF-8 INS

*C:\Program Files\Not...

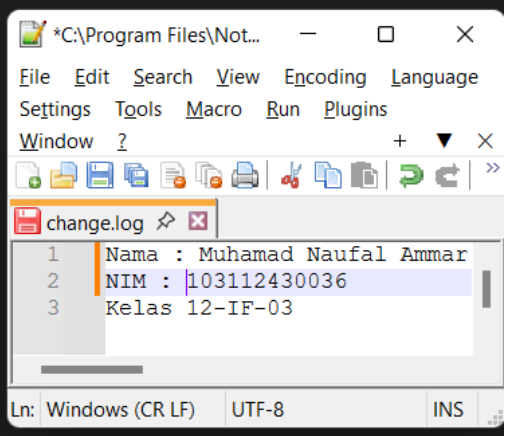
File Edit Search View Encoding Language
Settings Tools Macro Run Plugins
Window ?

change.log

```
1  Nama : Muhamad Naufal Ammar
2  NIM : 103112430036
3  Kelas 12-IF-03
```

Ln: Windows (CR LF) UTF-8 INS

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

using namespace std;

int main () {
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout << "Enqueue 3 Elemen" << endl;
    enqueue(Q, 5);
    printInfo(Q);
    enqueue(Q, 2);
    printInfo(Q);
    enqueue(Q, 9);
    printInfo(Q);

    cout << "Dequeue 1 Elemen" << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

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

    cout << "Dequeue 2 Elemen" << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

    return 0;

}
```



Screenshots Output

```
PS C:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided> cd "c:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Guided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Isi Antrean: [Enqueue 3 Elemen
Isi Antrean: [5 ]
Isi Antrean: [5 2 ]
Isi Antrean: [5 2 9 ]
Dequeue 1 Elemen
Elemen keluar: 5
Isi Antrean: [2 9 ]
Enqueue 1 Elemen
Isi Antrean: [2 9 4 ]
Dequeue 2 Elemen
Elemen keluar: 2
Isi Antrean: [9 4 ]
```

Deskripsi:

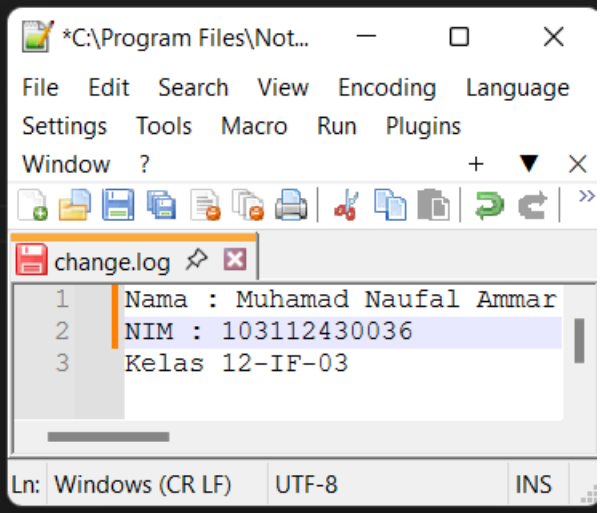
Implementasi struktur data circular queue menggunakan array statis dengan pendekatan indeks melingkar untuk memaksimalkan penggunaan ruang penyimpanan. Pada awalnya, fungsi createQueue menginisialisasi posisi head, tail, dan jumlah elemen (count) menjadi nol sebagai kondisi antrian kosong. Operasi dasar seperti isEmpty dan isFull digunakan untuk memeriksa kondisi antrian sebelum dilakukan proses enqueue atau dequeue. Prosedur enqueue menambahkan elemen baru ke posisi tail, kemudian menggeser indeksnya secara melingkar menggunakan operasi modulo agar tidak keluar batas array, sekaligus menambah jumlah elemen. Sebaliknya, fungsi dequeue mengambil elemen dari posisi head, menggeser indeksnya secara melingkar, dan mengurangi jumlah elemen. Untuk menampilkan isi antrian, printInfo mengiterasi elemen mulai dari head hingga jumlah elemen yang tersisa. Konsep indeks melingkar ini membuat antrian dapat berjalan efisien tanpa perlu melakukan shifting data seperti pada queue linear.

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

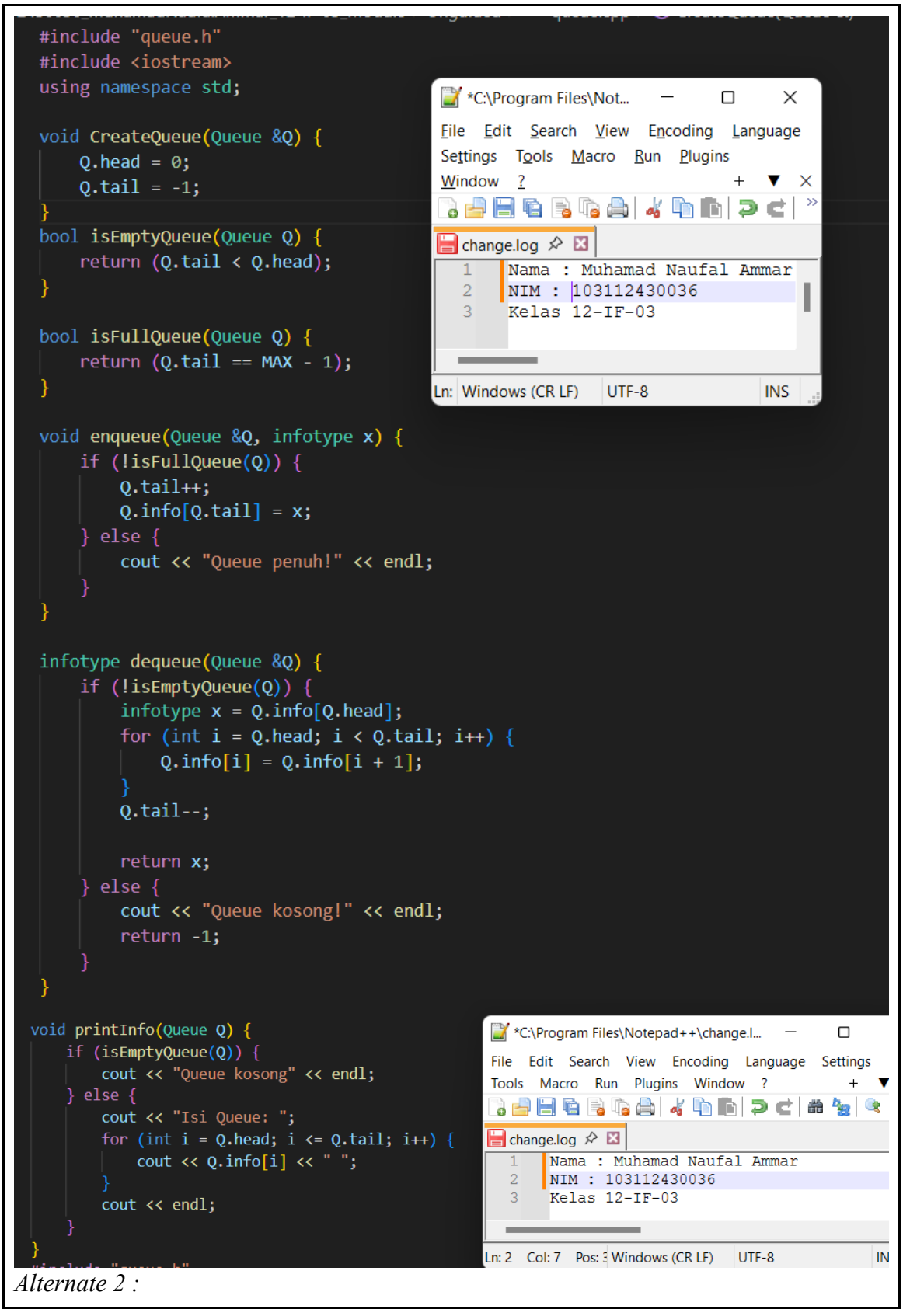
Unguided 1

Queue.h

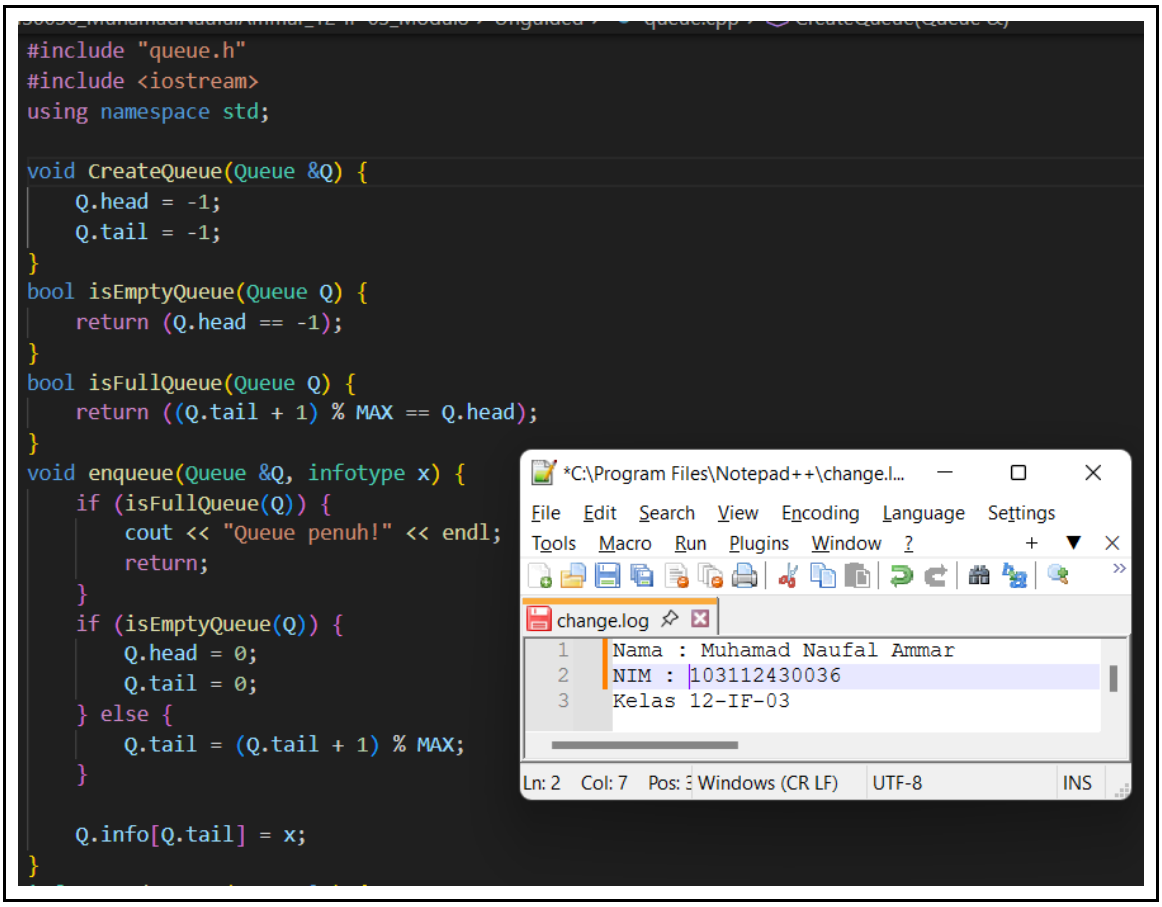
```h
#ifndef QUEUE_H
#define QUEUE_H


const int MAX = 5;
typedef int infotype;


typedef struct {
infotype info[MAX];
int head;
int tail;
} Queue;


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
```

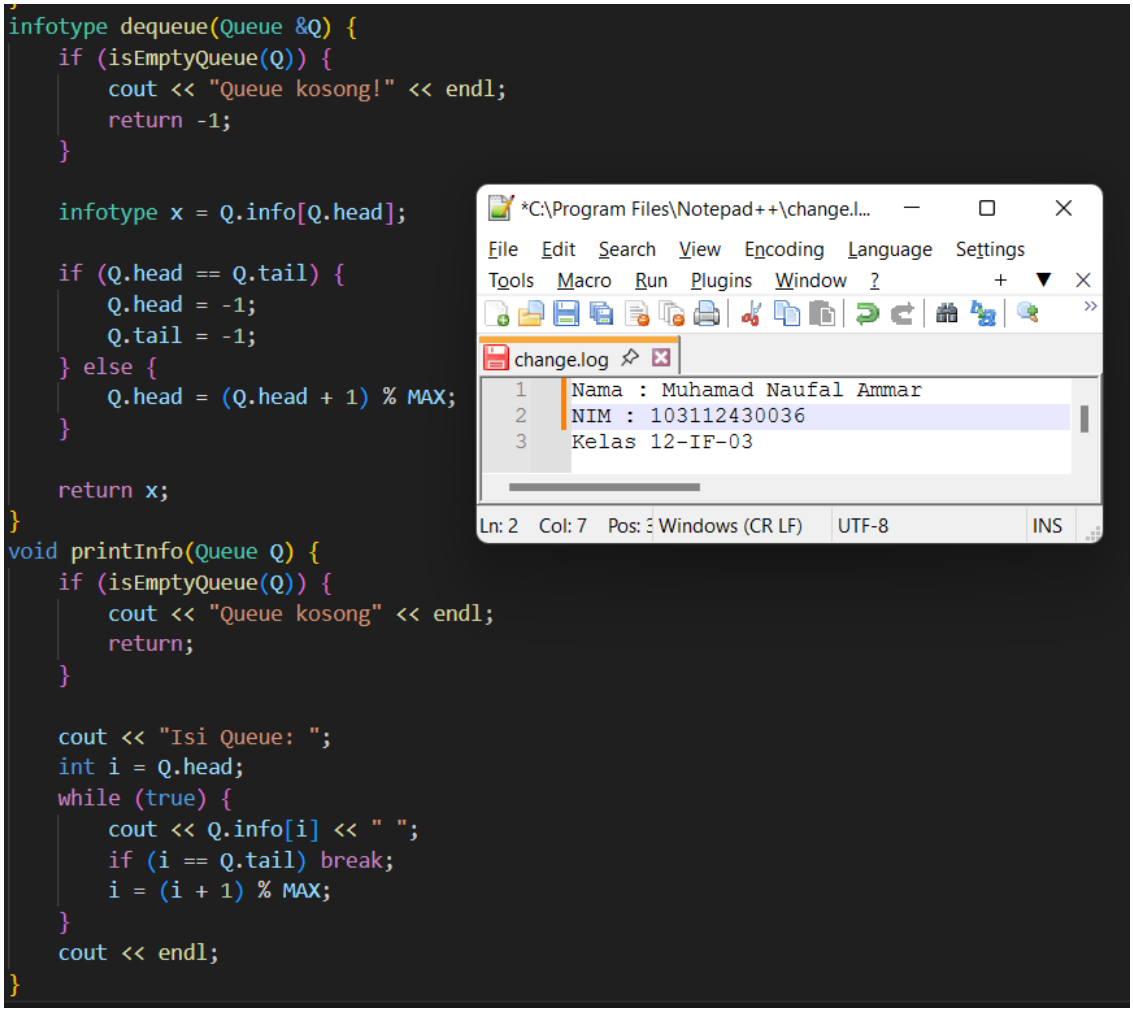


Queue.cpp
Alternate 1;


#include "queue.h"
#include <iostream>
using namespace std;
void CreateQueue(Queue &Q) {
Q.head = 0;
Q.tail = -1;
}
bool isEmptyQueue(Queue Q) {
return (Q.tail < Q.head);
}
bool isFullQueue(Queue Q) {
return (Q.tail == MAX - 1);
}
void enqueue(Queue &Q, infotype x) {
if (!isFullQueue(Q)) {
Q.tail++;
Q.info[Q.tail] = x;
} else {
cout << "Queue penuh!" << endl;
}
}
infotype dequeue(Queue &Q) {
if (!isEmptyQueue(Q)) {
infotype x = Q.info[Q.head];
for (int i = Q.head; i < Q.tail; i++) {
Q.info[i] = Q.info[i + 1];
}
Q.tail--;
return x;
} else {
cout << "Queue kosong!" << endl;
return -1;
}
}
void printInfo(Queue Q) {
if (isEmptyQueue(Q)) {
cout << "Queue kosong" << endl;
} else {
cout << "Isi Queue: ";
for (int i = Q.head; i <= Q.tail; i++) {
cout << Q.info[i] << " ";
}
cout << endl;
}
}
*C:\Program Files\Not...
File Edit Search View Encoding Language
Settings Tools Macro Run Plugins
Window ?
change.log
1 Nama : Muhamad Naufal Ammar
2 NIM : 103112430036
3 Kelas 12-IF-03
Ln: Windows (CR LF) UTF-8 INS
*C:\Program Files\Notepad++\change.l...
File Edit Search View Encoding Language Settings
Tools Macro Run Plugins Window ?
change.log
1 Nama : Muhamad Naufal Ammar
2 NIM : 103112430036
3 Kelas 12-IF-03
Ln: 2 Col: 7 Pos: Windows (CR LF) UTF-8 IN

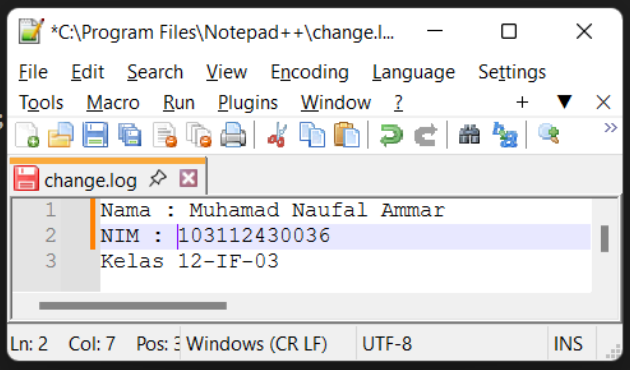
Alternate 2 :


#include "queue.h"
#include <iostream>
using namespace std;
void CreateQueue(Queue &Q) {
Q.head = -1;
Q.tail = -1;
}
bool isEmptyQueue(Queue Q) {
return (Q.head == -1);
}
bool isFullQueue(Queue Q) {
return ((Q.tail + 1) % MAX == 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) % MAX;
}
Q.info[Q.tail] = x;
}
*C:\Program Files\Notepad++\change.l...
File Edit Search View Encoding Language Settings
Tools Macro Run Plugins Window ?
change.log
1 Nama : Muhamad Naufal Ammar
2 NIM : 103112430036
3 Kelas 12-IF-03
Ln: 2 Col: 7 Pos: Windows (CR LF) UTF-8 INS

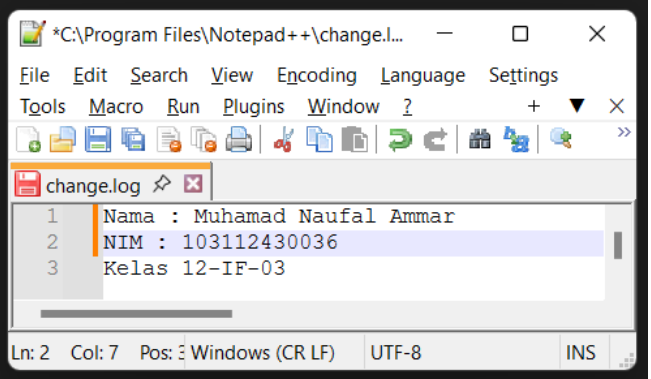

infotype dequeue(Queue &Q) {
if (isEmptyQueue(Q)) {
cout << "Queue kosong!" << endl;
return -1;
}
infotype x = Q.info[Q.head];
if (Q.head == Q.tail) {
Q.head = -1;
Q.tail = -1;
} else {
Q.head = (Q.head + 1) % MAX;
}
return x;
}
void printInfo(Queue Q) {
if (isEmptyQueue(Q)) {
cout << "Queue kosong" << endl;
return;
}
cout << "Isi Queue: ";
int i = Q.head;
while (true) {
cout << Q.info[i] << " ";
if (i == Q.tail) break;
i = (i + 1) % MAX;
}
cout << endl;
}
*C:\Program Files\Notepad++\change.l...
File Edit Search View Encoding Language Settings
Tools Macro Run Plugins Window ?
change.log
1 Nama : Muhamad Naufal Ammar
2 NIM : 103112430036
3 Kelas 12-IF-03
Ln: 2 Col: 7 Pos: Windows (CR LF) UTF-8 INS

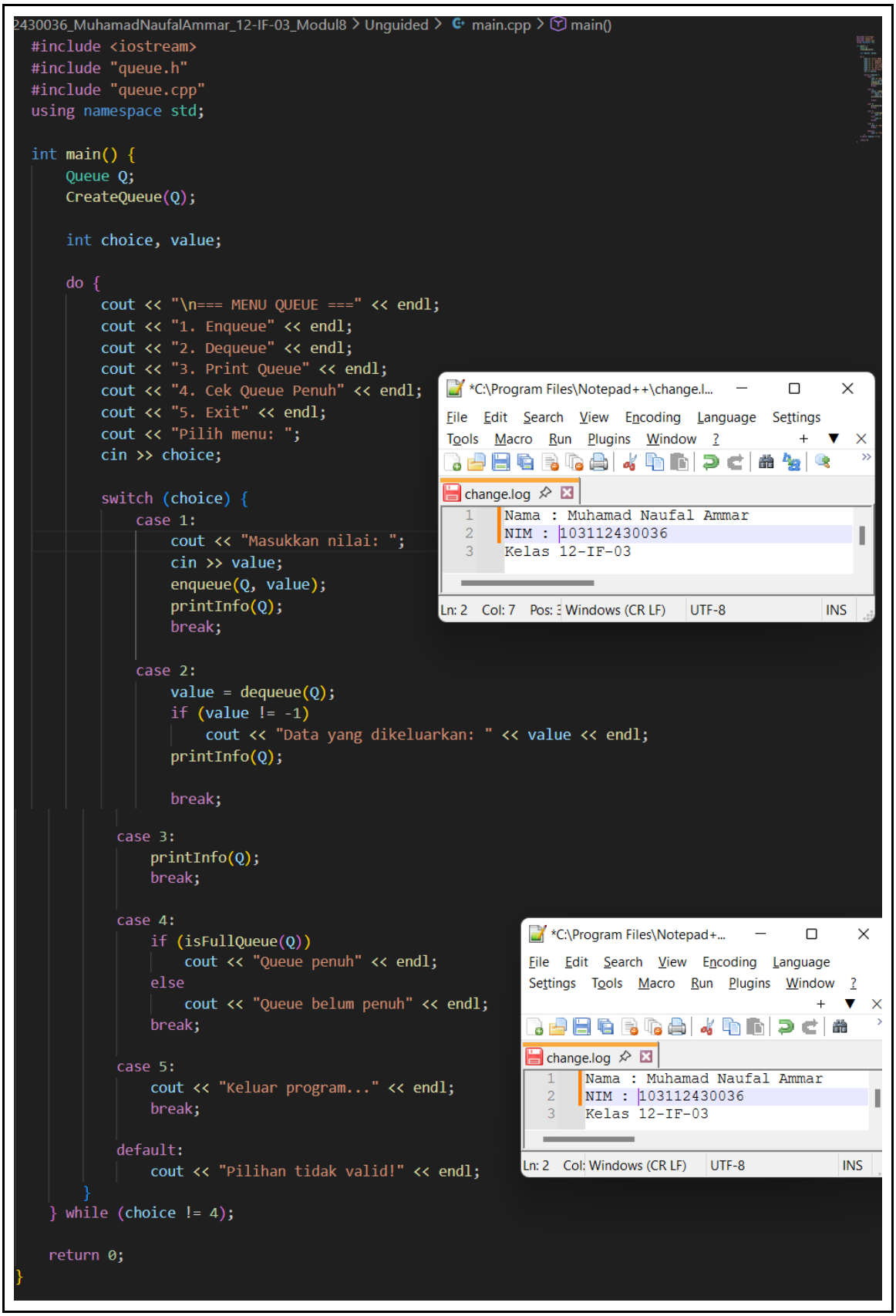
Alternate 3

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

void CreateQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return ((Q.tail + 1) % MAX == 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) % MAX;
    }

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



```cpp
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) % MAX;
    }

    return x;
}

void printInfo(Queue Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue kosong" << endl;
        return;
    }

    cout << "Isi Queue: ";
    int i = Q.head;
    while (true) {
        cout << Q.info[i] << " ";
        if (i == Q.tail) break;
        i = (i + 1) % MAX;
    }
    cout << endl;
}
```





Screenshots Output

Alternate 1 :

```
PS C:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided> cd "c:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 5
Isi Queue: 5

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 2
Isi Queue: 5 2

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 7
Isi Queue: 5 2 7
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 5
Isi Queue: 2 7

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 4
Isi Queue: 2 7 4

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 2
Isi Queue: 7 4
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 7
Isi Queue: 4
```

Alternate 2 :

```
PS C:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided> cd "c:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 5
Isi Queue: 5

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 2
Isi Queue: 5 2

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 7
Isi Queue: 5 2 7
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 5
Isi Queue: 2 7

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 4
Isi Queue: 2 7 4

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 2
Isi Queue: 7 4
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 7
Isi Queue: 4
```

Alternate 3:

```
PS C:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided> cd "c:\Users\user\Documents\modul-amer\103112430036_MuhamadNaufalAmmar_12-IF-03_Modul8\Unguided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 5
Isi Queue: 5

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 2
Isi Queue: 5 2

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 7
Isi Queue: 5 2 7
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 5
Isi Queue: 2 7

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 1
Masukkan nilai: 4
Isi Queue: 2 7 4

=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 2
Isi Queue: 7 4
```

```
=== MENU QUEUE ===
1. Enqueue
2. Dequeue
3. Print Queue
4. Cek Queue Penuh
5. Exit
Pilih menu: 2
Data yang dikeluarkan: 7
Isi Queue: 4
```

Deskripsi:

Implementasi kode pada program queue ini memanfaatkan struktur data antrian berbasis array dengan mekanisme circular queue, di mana posisi head dan tail akan bergerak serta berputar mengikuti indeks menggunakan operasi modulo sehingga ruang penyimpanan dapat dimanfaatkan secara lebih efisien tanpa perlu melakukan shifting elemen seperti pada antrian linear biasa. Fungsi-fungsi seperti enqueue, dequeue, isFullQueue, dan isEmptyQueue bekerja secara terkoordinasi untuk memastikan bahwa proses penambahan dan penghapusan data tetap mengikuti prinsip FIFO (First In First Out) sambil menjaga kondisi antrian tetap valid. Selain itu, fungsi printInfo digunakan untuk menampilkan isi antrian berdasarkan posisi head dan tail saat ini, sehingga mahasiswa dapat memahami bagaimana data berputar mengikuti konsep antrian melingkar. Dengan desain seperti ini, antrian dapat beroperasi lebih optimal dan stabil meskipun menggunakan array berukuran tetap.

.

D. Kesimpulan

Secara praktis, implementasi *circular queue* (head & tail berputar) lebih efisien dan fleksibel untuk aplikasi yang membutuhkan pemrosesan berkelanjutan atau real-time karena mengurangi operasi *copying* dan memaksimalkan ulang pakai slot array; sementara implementasi linear dengan head tetap dan *shifting* mungkin lebih sederhana secara konsep tapi kurang efisien untuk beban kerja nyata, sehingga pemilihan implementasi harus disesuaikan dengan kebutuhan memori, frekuensi operasi enqueue/dequeue, dan batasan perangkat keras.

E. Referensi

Fatourou, P., Giachoudis, N., & Mallis, G. (2024). *Highly-efficient persistent FIFOqueues*.

Nikolaev, R., & Ravindran, B. (2022). *wCQ: A fast wait-free queue with bounded memory usage*. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '22)* (hal. 307–319). Association for Computing Machinery.