

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

Nia Novela Ariandini
2311102057

Dosen

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

Queue bisa disebut juga antrian pada struktur data. Pengertian queue adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung yang disebut sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain. Contoh paling sederhana dapat dilihat pada antrian. Prinsip kerja dari queue adalah prinsip “First In First Out” (FIFO) atau “masuk pertama keluar pertama”. Sedangkan prinsip “masuk terakhir keluar pertama” atau “Last In First Out” (LIFO), digunakan pada tumpukan atau stack.

Pada antrian terdapat satu pintu masuk di salah satu ujung dan satu pintu keluar di ujung lainnya. Maka ada penunjuk yang menunjukkan awal dan akhir. Operasi penting dalam antrian :

1. Add yang berfungsi menambah sebuah elemen ke dalam antrian.
2. Delete yang berfungsi menghapus atau mengeluarkan elemen dari antrian.

Dalam ilmu komputer, antrian banyak digunakan terutama dalam sistem operasi yang memerlukan manajemen sumber daya dan penjadwalan. Contohnya time-sharing computer-system yang bisa dipakai oleh sejumlah orang secara serempak. Sebuah antrian memiliki suatu operasi bernama `add_priority`. Dalam hal ini, antrian tidak lagi menerapkan konsep antrian yang murni, namun berubah menjadi antrian sesuai prioritas tertentu pada elemen, dan elemen yang baru ditambah tidak selalu berada di akhir.

Operasi-Operasi Queue :

1. Create Untuk menciptakan dan menginisialisasi antrian dengan cara membuat Head dan Tail = -1
2. IsEmpty Dipakai untuk memeriksa penuh tidaknya sebuah antrian dengan cara memeriksa nilai tail, jika tail = -1 maka empty. Kita tidak memeriksa head, karena head adalah tanda kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah. Pergerakan pada antrian terjadi dengan penambahan elemen antrian di bagian belakang, yaitu menggunakan nilai tail.
3. IsFull Dipakai untuk mengecek penuh tidaknya antrian dengan cara mengecek nilai tail, jika tail \geq MAX-1 (karena MAX-1 adalah batas elemen array pada C) maka sudah penuh.
4. EnQueue Digunakan untuk penambahan elemen ke dalam antrian, penambahan elemen selalu ditambahkan di elemen paling belakang. Penambahan elemen selalu menggerakkan variabel tail dengan cara increment counter tail terlebih dahulu

5. DeQueue Dipakai untuk menghapus elemen terdepan (head) dari antrian dengan cara menggeser semua elemen antrian ke bagian depan dan mengurangi tail dengan 1 penggeseran yang dilakukan dengan menggunakan pengulangan.
6. Clear Digunakan untuk menghapus elemen-elemen antrian dengan cara membuat tail dan head bernilai -1. Penghapusan elemen-elemen antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesannya menjadi -1 sehingga elemen-elemen antrian tidak lagi terbaca.
7. Tampil Dipakai untuk menampilkan nilai-nilai elemen antrian menggunakan pengulangan dari head hingga tail.

B. Guided

Guided 1

```
#include <iostream>
using namespace std;
const int maksimalQueue = 5; // Maksimal antrian
int front = 0;                // Penanda antrian
int back = 0;                 // Penanda
string queueTeller[5];        // Fungsi pengecekan
bool isFull()
{ // Pengecekan antrian penuh atau tidak
    if (back == maksimalQueue)
    {
        return true; // =1
    }
    else
    {
        return false;
    }
}
bool isEmpty()
{ // Antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
void enqueueAntrian(string data)
{ // Fungsi menambahkan antrian
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        if (isEmpty())
        { // Kondisi ketika queue kosong
```

```

        queueTeller[0] = data;
        front++;
        back++;
    }
    else
    { // Antrianya ada isi
        queueTeller[back] = data;
        back++;
    }
}

void dequeueAntrian()
{ // Fungsi mengurangi antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}

int countQueue()
{ // Fungsi menghitung banyak antrian
    return back;
}

void clearQueue()
{ // Fungsi menghapus semua antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
    }
}

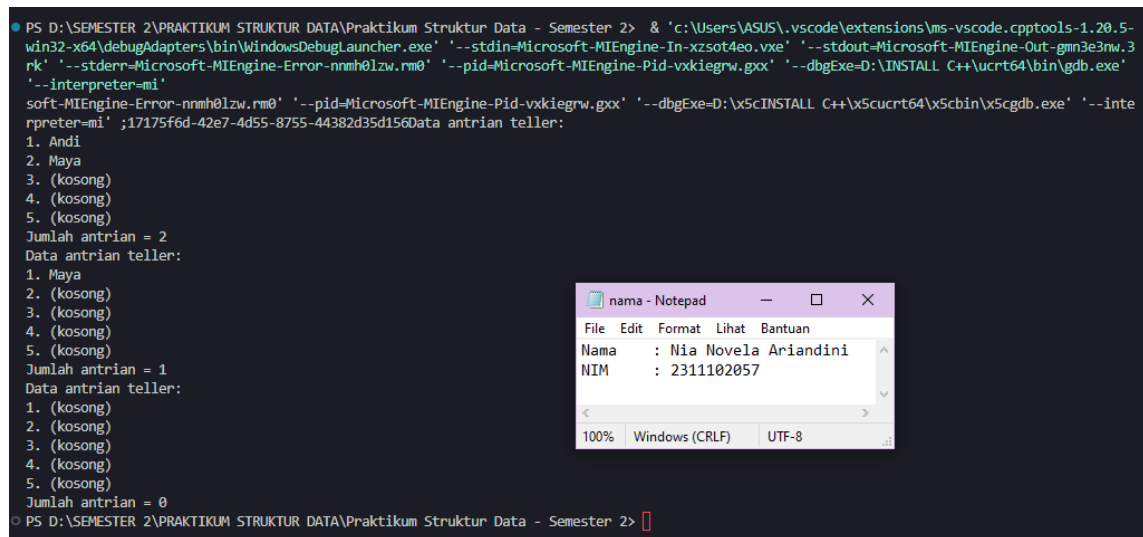
```

```

        back = 0;
        front = 0;
    }
}
void viewQueue()
{ // Fungsi melihat antrian
    cout << "Data antrian teller:" << endl;
    for (int i = 0; i < maksimalQueue; i++)
    {
        if (queueTeller[i] != "")
        {
            cout << i + 1 << ". " << queueTeller[i] << endl;
        }
        else
        {
            cout << i + 1 << ". (kosong)" << endl;
        }
    }
}
int main()
{
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Screenshots Output :



```
PS D:\SEMESTER 2\PRAKTIKUM STRUKTUR DATA\Praктикум Struktur Data - Semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-xzst4eo.vxe' '--stdout=Microsoft-MIEngine-Out-gmn3e3nw.3rk' '--stderr=Microsoft-MIEngine-Error-nnmh0lw.rm0' '--pid=Microsoft-MIEngine-Pid-vxkiegrw.gxx' '--dbgExe=D:\INSTALL C++\ucrt64\bin\gdb.exe' '--interpreter=mi'
soft-MIEngine-Error-nnmh0lw.rm0' '--pid=Microsoft-MIEngine-Pid-vxkiegrw.gxx' '--dbgExe=D:\x5cINSTALL C++\x5cucrt64\x5cbin\x5cgdb.exe' '--inte
rpreter=mi' ;17175f6d-42e7-4d55-8755-44382d35d156Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS D:\SEMESTER 2\PRAKTIKUM STRUKTUR DATA\Praктикум Struktur Data - Semester 2>
```

The screenshot shows a terminal window with the output of a C++ program. The program implements a queue using an array. The output shows the initial state of the queue, the addition of two elements (Andi and Maya), the removal of the first element (Andi), and the final state of the queue (Maya and two empty slots). A Notepad window is also open, displaying the name 'Nia Novela Ariandini' and the NIM '2311102057'.

Deskripsi :

Script di atas mengimplementasikan antrian (queue) menggunakan array statis untuk menyimpan data nama pelanggan dalam antrian. Variabel front dan back digunakan sebagai penanda untuk mengelola posisi depan dan belakang dari antrian. Fungsi isFull() digunakan untuk memeriksa apakah antrian sudah penuh, sedangkan isEmpty() memeriksa apakah antrian kosong. Fungsi tambahData() menambahkan data ke antrian jika belum penuh, dan mengatur front dan back sesuai kondisi. Fungsi kurangAntrian() menghapus data dari depan antrian dan menggeser semua elemen ke kiri. count() menghitung jumlah elemen dalam antrian, dan clearQueue() mengosongkan seluruh antrian. Fungsi viewQueue() menampilkan semua elemen dalam antrian, menunjukkan posisi dan status (kosong atau terisi). Dalam fungsi main(), script menambahkan beberapa data ke antrian menggunakan tambahData(), menampilkan isi antrian dengan viewQueue(), kemudian mengurangi satu data dari antrian dengan kurangAntrian() dan kembali menampilkan isi antrian. Terakhir, seluruh antrian dihapus menggunakan clearQueue(), dan isi antrian ditampilkan kembali untuk memastikan antrian telah kosong. Pendekatan ini sederhana namun efisien untuk manajemen antrian dalam skala kecil dengan batas maksimal yang tetap, cocok untuk kasus penggunaan dasar.

C. Unguided/Tugas

Unguided 1

Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

Source Code :

```
#include <iostream>
using namespace std;

// Struktur simpul untuk node dalam linked list
struct Mahasiswa {
    string nama057;
    Mahasiswa* next;
};

// Kelas Queue yang menggunakan linked list
class Queue {
private:
    Mahasiswa* front; // Pointer ke depan antrian
    Mahasiswa* back;  // Pointer ke belakang antrian

public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

    // Fungsi untuk mengecek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Fungsi untuk menambahkan data ke antrian
    void enqueue(string nama057) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama057 = nama057;
        newNode->next = nullptr;

        // Jika antrian kosong, node baru menjadi front dan
back
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            // Jika antrian tidak kosong, tambahkan node baru
            ke belakang dan update back
        }
    }
};
```



```

        back->next = newNode;
        back = newNode;
    }
    cout << "Data berhasil dimasukkan ke dalam antrian" <<
endl;
}

// Fungsi untuk menghapus data dari antrian
void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        // Jika antrian tidak kosong, hapus node pertama
        dan update front
        Mahasiswa* temp = front;
        front = front->next;
        delete temp;
        cout << "Data berhasil dihapus dari antrian" <<
endl;
    }
}

// Fungsi untuk menghitung jumlah data dalam antrian
int countQueue() {
    int count = 0;
    Mahasiswa* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi untuk menghapus semua data dari antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeue();
    }
    cout << "Data berhasil di-reset" << endl;
}

// Fungsi untuk menampilkan data dalam antrian

```

```

void viewQueue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        cout << "Data antrian Mahasiswa : " << endl;
        Mahasiswa* temp = front;
        int pos = 1;
        while (temp != nullptr) {
            cout << pos << ". Nama : " << temp->nama057 <<
endl;

            temp = temp->next;
            pos++;
        }
    }
};

int main() {
    Queue queue;
    int choice;

    do {
        cout << "=== Menu Antrian Mahasiswa Telkom ===" <<
endl;
        cout << "1. Masukkan data" << endl;
        cout << "2. Hapus satu data" << endl;
        cout << "3. Reset data" << endl;
        cout << "4. Tampil data" << endl;
        cout << "5. Keluar" << endl;
        cout << "=====\n" <<
endl;

        cout << "Masukkan Pilihan Anda : ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string nama;
                cout << "Masukkan Nama Mahasiswa : ";
                cin >> nama;
                queue.enqueue(nama);
                break;
            }

```

```

        case 2: {
            if (queue.isEmpty()) {
                cout << "Antrian kosong" << endl;
            } else {
                queue.dequeue ();
            }
            break;
        }
        case 3: {
            if (queue.isEmpty()) {
                cout << "Antrian kosong" << endl;
            } else {
                queue.clearQueue();
            }
            break;
        }
        case 4: {
            queue.viewQueue();
            break;
        }
        case 5: {
            cout << "Terima kasih telah menggunakan
layanan kami ;)" << endl;
            break;
        }
        default: {
            cout << "Pilihan yang Anda masukkan tidak
valid" << endl;
            break;
        }
    }

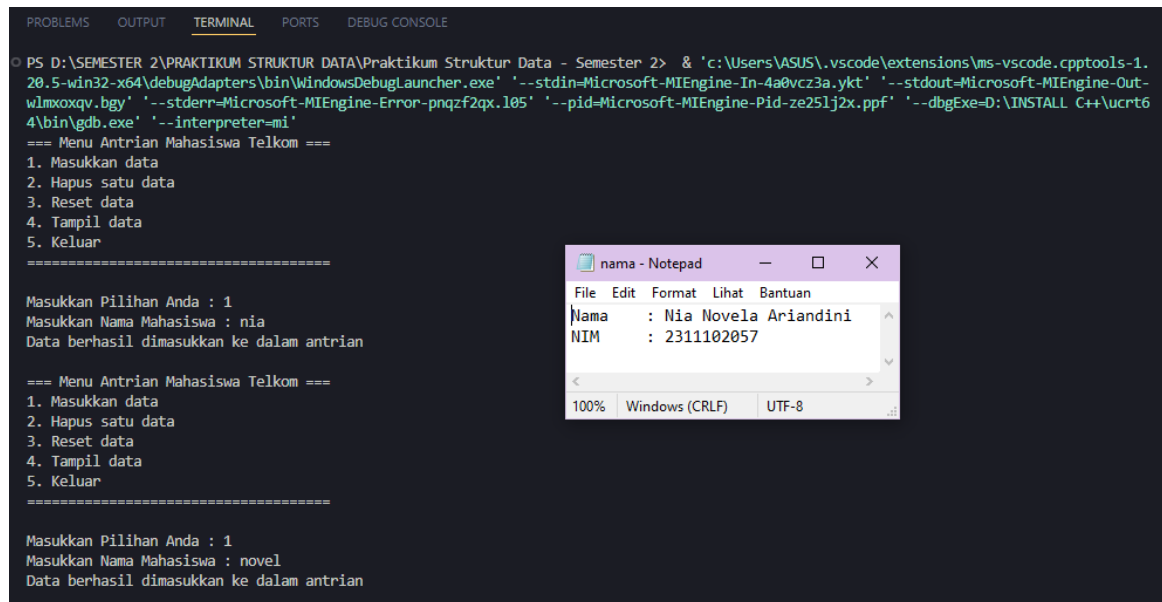
    cout << endl;

    } while (choice != 5);

    return 0;
}

```

Screenshots Output :



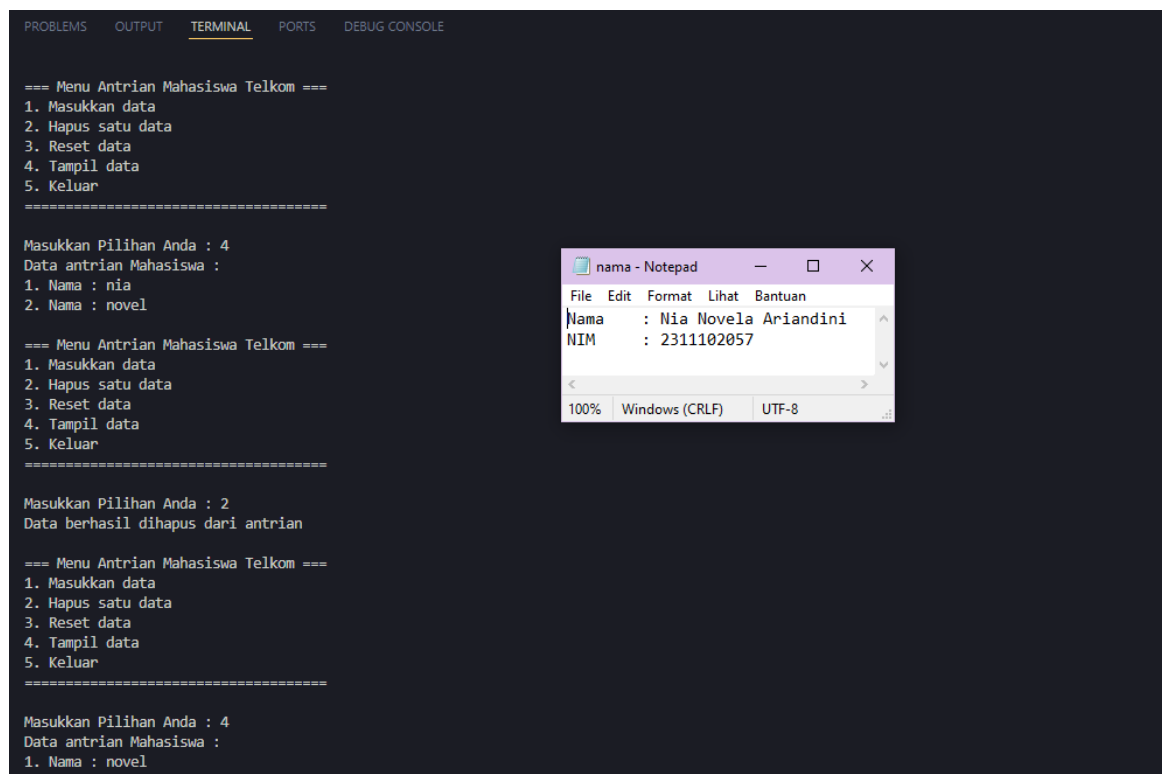
```
PS D:\SEMESTER 2\PRAKTIKUM STRUKTUR DATA\Praktikum Struktur Data - Semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-4a0vcz3a.ykt' '--stdout=Microsoft-MIEngine-Out-wlmoqxv.bgy' '--stderr=Microsoft-MIEngine-Error-pnqzf2qx.l05' '--pid=Microsoft-MIEngine-Pid-ze25lj2x.ppf' '--dbgExe=D:\INSTALL C++\ucrt64\bin\gdb.exe' '--interpreter=mi'
=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 1
Masukkan Nama Mahasiswa : nia
Data berhasil dimasukkan ke dalam antrian

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 1
Masukkan Nama Mahasiswa : novel
Data berhasil dimasukkan ke dalam antrian
```

```
nama - Notepad
File Edit Format Lihat Bantuan
Nama : Nia Novela Ariandini
NIM : 2311102057
100% Windows (CRLF) UTF-8
```



```
=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 4
Data antrian Mahasiswa :
1. Nama : nia
2. Nama : novel

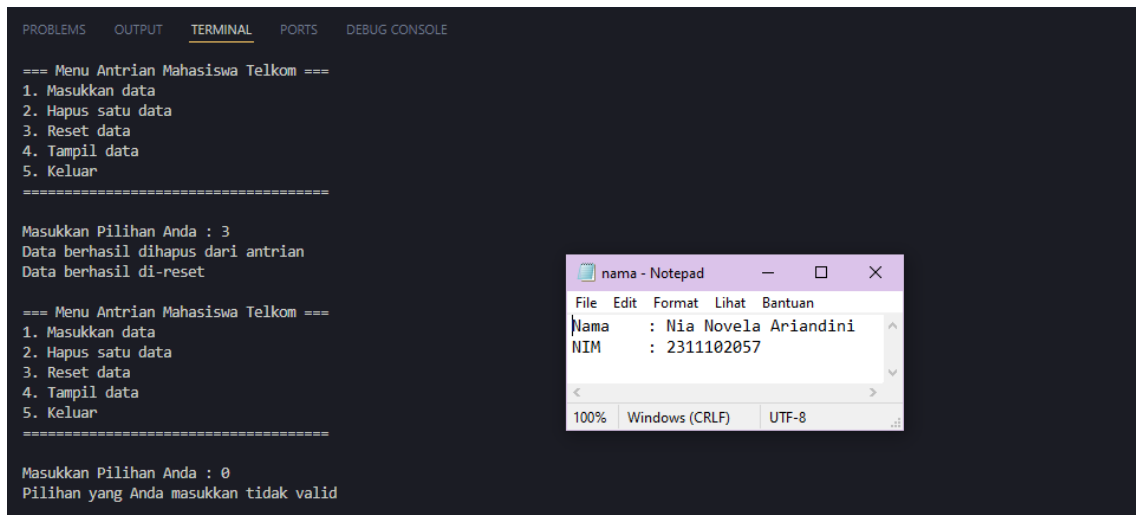
=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 2
Data berhasil dihapus dari antrian

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 4
Data antrian Mahasiswa :
1. Nama : novel
```

```
nama - Notepad
File Edit Format Lihat Bantuan
Nama : Nia Novela Ariandini
NIM : 2311102057
100% Windows (CRLF) UTF-8
```



```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 3
Data berhasil dihapus dari antrian
Data berhasil di-reset

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 0
Pilihan yang Anda masukkan tidak valid
```

nama - Notepad

File	Edit	Format	Lihat	Bantuan
Nama	:	Nia Novela Ariandini		
NIM	:	2311102057		

100% Windows (CRLF) UTF-8

Deskripsi :

Script di atas mengimplementasikan sebuah kelas Queue untuk mengelola antrian (queue) mahasiswa menggunakan linked list. Struktur Mahasiswa mendefinisikan node dalam linked list, yang terdiri dari dua atribut: nama140 dan nim140, serta pointer next untuk menunjuk ke node berikutnya dalam antrian. Kelas Queue memiliki dua pointer, front untuk menunjuk ke node pertama dalam antrian, dan back untuk menunjuk ke node terakhir. Berikut adalah penjelasan singkat tentang fungsi-fungsi yang ada di kelas Queue:

1. isEmpty(): Mengecek apakah antrian kosong dengan mengembalikan nilai true jika front adalah nullptr.
2. enqueue(): Menambahkan node baru ke belakang antrian. Jika antrian kosong, node baru menjadi front dan back. Jika tidak, node baru ditambahkan di belakang dan back diperbarui.
3. dequeue(): Menghapus node dari depan antrian. Jika antrian kosong, menampilkan pesan bahwa antrian kosong. Jika tidak, node pertama dihapus dan front diperbarui ke node berikutnya.
4. countQueue(): Menghitung dan mengembalikan jumlah node dalam antrian dengan menghitung node dari front hingga nullptr.
5. clearQueue(): Menghapus semua node dalam antrian dengan menggunakan fungsi dequeue secara berulang hingga antrian kosong.
6. viewQueue(): Menampilkan semua data dalam antrian. Jika antrian kosong, menampilkan pesan bahwa antrian kosong. Jika tidak, menampilkan nama dan NIM dari setiap node dalam antrian.

Kelas ini menyediakan fungsi dasar untuk mengelola antrian mahasiswa dengan linked list, termasuk penambahan, penghapusan, penghitungan, pengosongan, dan penampilan data dalam antrian.

Unguided 2

Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

Source Code :

```
#include <iostream>
using namespace std;

// Struktur simpul untuk node dalam linked list
struct Mahasiswa {
    string nama057;
    string nim057;
    Mahasiswa* next;
};

// Kelas Queue yang menggunakan linked list
class Queue {
private:
    Mahasiswa* front; // Pointer ke depan antrian
    Mahasiswa* back;  // Pointer ke belakang antrian

public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

    // Fungsi untuk mengecek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Fungsi untuk menambahkan data ke antrian
    void enqueue(string nama057_, string nim057_) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama057 = nama057_;
```

```

        newNode->nim057 = nim057_;
        newNode->next = nullptr;

        // Jika antrian kosong, node baru menjadi front dan
back
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            // Jika antrian tidak kosong, tambahkan node baru
ke belakang dan update back
            back->next = newNode;
            back = newNode;
        }
        cout << "Data berhasil dimasukkan ke dalam antrian" <<
endl;
    }

    // Fungsi untuk menghapus data dari antrian
void dequeue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        // Jika antrian tidak kosong, hapus node pertama
dan update front
        Mahasiswa* temp = front;
        front = front->next;
        delete temp;
        cout << "Data berhasil dihapus dari antrian" <<
endl;
    }
}

    // Fungsi untuk menghitung jumlah data dalam antrian
int countQueue() {
    int count = 0;
    Mahasiswa* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

```

```

}

// Fungsi untuk menghapus semua data dari antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeue();
    }
    cout << "Data berhasil di-reset" << endl;
}

// Fungsi untuk menampilkan data dalam antrian
void viewQueue() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        cout << "Data antrian Mahasiswa : " << endl;
        Mahasiswa* temp = front;
        int pos = 1;
        while (temp != nullptr) {
            cout << pos << ". Nama : " << temp->nama057 <<
" || NIM : " << temp->nim057 << endl;
            temp = temp->next;
            pos++;
        }
    }
}

};

int main() {
    Queue queue;
    int choice;

    do {
        cout << "=== Menu Antrian Mahasiswa Telkom ===" <<
endl;

        cout << "1. Masukkan data" << endl;
        cout << "2. Hapus satu data" << endl;
        cout << "3. Reset data" << endl;
        cout << "4. Tampil data" << endl;
        cout << "5. Keluar" << endl;
        cout << "=====\n" <<
endl;
    } while (choice != 5);
}

```



```

cout << "Masukkan Pilihan Anda : ";
cin >> choice;

switch (choice) {
    case 1: {
        string nama, nim;
        cout << "Masukkan Nama Mahasiswa : ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM Mahasiswa : ";
        cin >> nim;
        queue.enqueue(nama, nim);
        break;
    }
    case 2: {
        if (queue.isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            queue.dequeue ();
        }
        break;
    }
    case 3: {
        if (queue.isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            queue.clearQueue();
        }
        break;
    }
    case 4: {
        queue.viewQueue();
        break;
    }
    case 5: {
        cout << "Terima kasih telah menggunakan
layanan kami ;)" << endl;
        break;
    }
    default: {
        cout << "Pilihan yang Anda masukkan tidak
valid" << endl;
    }
}

```

```

        break;
    }
}

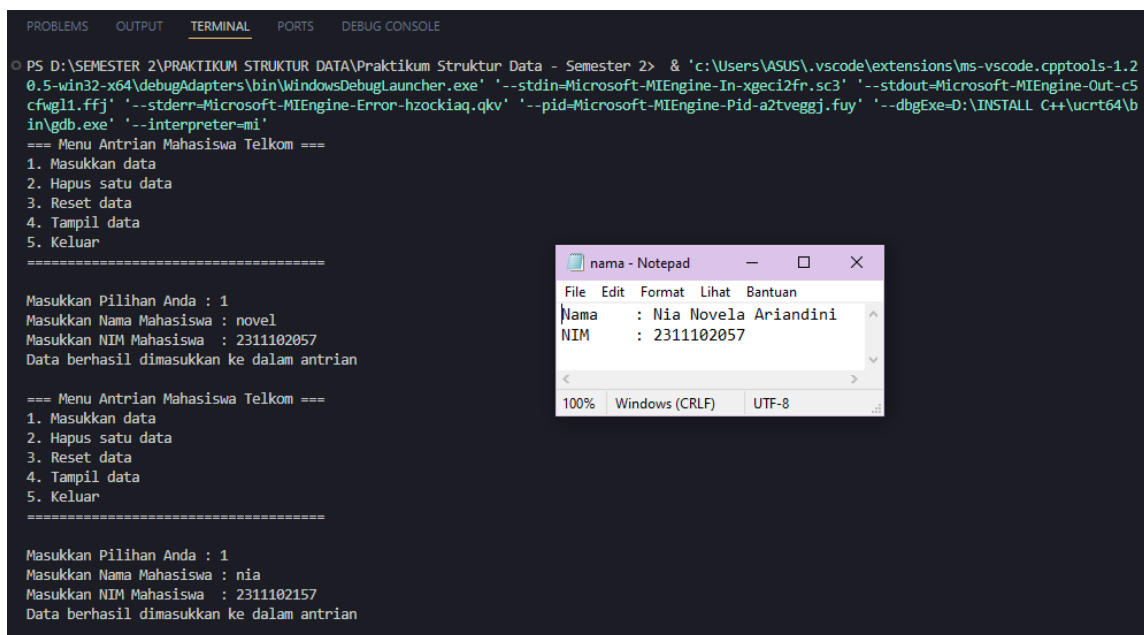
cout << endl;

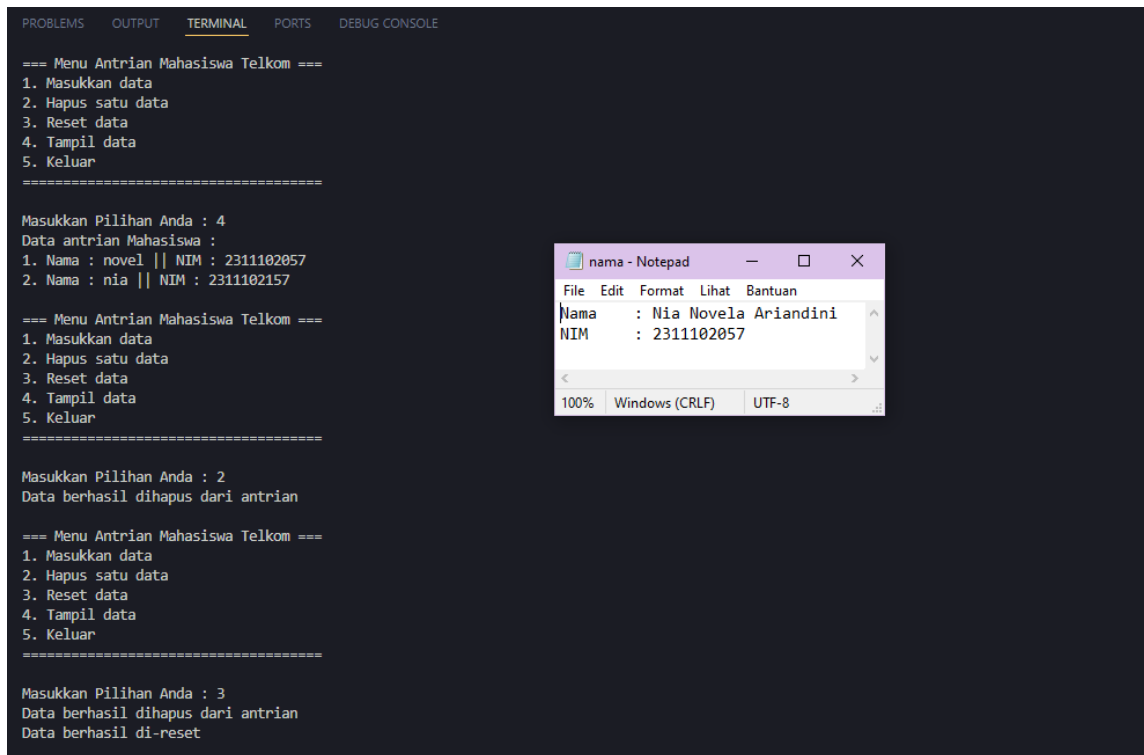
} while (choice != 5);

return 0;
}

```

Screenshots Output :





```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 4
Data antrian Mahasiswa :
1. Nama : novel || NIM : 2311102057
2. Nama : nia || NIM : 2311102157

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 2
Data berhasil dihapus dari antrian

=== Menu Antrian Mahasiswa Telkom ===
1. Masukkan data
2. Hapus satu data
3. Reset data
4. Tampil data
5. Keluar
=====

Masukkan Pilihan Anda : 3
Data berhasil dihapus dari antrian
Data berhasil di-reset
```

Deskripsi :

Script di atas mengimplementasikan antrian (queue) menggunakan linked list untuk menyimpan data mahasiswa berupa nama dan NIM. Struktur Mahasiswa digunakan untuk mendefinisikan node dalam linked list yang memiliki dua atribut: nama140 dan nim140, serta pointer next untuk menunjuk ke node berikutnya. Kelas Queue memiliki metode untuk memeriksa apakah antrian kosong (isEmpty), menambahkan data ke antrian (enqueue), menghapus data dari antrian (dequeue), menghitung jumlah data dalam antrian (countQueue), menghapus semua data dalam antrian (clearQueue), dan menampilkan data dalam antrian (viewQueue). Dalam metode enqueue, node baru ditambahkan ke belakang antrian, dan dalam metode dequeue, node pertama dihapus dari antrian. Pada fungsi main(), terdapat menu interaktif yang memungkinkan pengguna untuk memilih beberapa operasi antrian: menambahkan data, menghapus satu data, menghapus semua data, menampilkan data, dan keluar dari program. Menu ini ditampilkan dalam loop do-while sehingga pengguna dapat melakukan beberapa operasi hingga memilih untuk keluar (pilihan 5). Setiap pilihan menu memiliki penanganan yang sesuai dalam switch-case statement. Pengguna dapat memasukkan nama dan NIM mahasiswa untuk menambahkannya ke antrian, melihat antrian saat ini, menghapus data dari antrian, atau mereset seluruh antrian. Program ini berguna untuk mensimulasikan pengelolaan antrian mahasiswa di sebuah layanan seperti di kampus.

D. Kesimpulan

Setelah mempelajari materi tentang Queue dan penerapannya menggunakan Linked List dan Array, kita memahami bahwa Queue adalah struktur data yang mengikuti prinsip FIFO (First-In-First-Out). Ini berarti elemen yang pertama kali masuk akan menjadi yang pertama kali keluar. Queue dapat diimplementasikan baik dengan Linked List maupun Array. Linked List menawarkan fleksibilitas dalam penambahan dan penghapusan elemen, sedangkan Array memungkinkan akses langsung ke elemen-elemen dalam antrian. Implementasi Queue menggunakan Linked List membutuhkan alokasi memori dinamis untuk setiap elemen yang ditambahkan atau dihapus, sehingga memungkinkan antrian tumbuh atau menyusut sesuai kebutuhan. Sebaliknya, implementasi Queue menggunakan Array memiliki batasan ukuran tetap, sehingga antrian memiliki kapasitas maksimum yang sudah ditentukan sebelumnya. Jika antrian penuh, elemen baru tidak bisa ditambahkan. Operasi dasar pada Queue mencakup enqueue (menambahkan elemen ke dalam antrian), dequeue (menghapus elemen dari antrian), isEmpty (memeriksa apakah antrian kosong), countQueue (menghitung jumlah elemen dalam antrian), dan viewQueue (menampilkan elemen-elemen dalam antrian). Queue sangat berguna dalam situasi yang membutuhkan pemrosesan data secara berurutan sesuai waktu kedatangan, seperti simulasi antrian pelanggan atau penjadwalan tugas. Pemilihan antara implementasi Queue dengan Linked List atau Array bergantung pada kebutuhan dan karakteristik spesifik masalah yang dihadapi. Linked List lebih fleksibel dalam hal ukuran yang bisa berubah, sementara Array lebih efisien dalam hal akses elemen. Dengan memahami konsep dasar Queue serta penerapannya menggunakan Linked List dan Array, kita dapat menggunakannya untuk memecahkan berbagai masalah yang melibatkan pengolahan data berdasarkan urutan waktu kedatangan.

E. Referensi

- [1] IEEE Communications Surveys & Tutorials (Volume: 15, Issue: 3, Third Quarter 2013)
- [2] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In SIGCOMM, 2013.
- [2] Asisten Praktikum. “Modul VI Queue”, Learning Management System, 2024.