LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 8
QUEUE

Disusun Oleh :

NAMA : Faris Walid Awwal Aidi

NIM : 103112430133

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Queue (antrean) adalah salah satu struktur data linear yang fundamental dalam ilmu komputer, yang berfungsi untuk menyimpan sekumpulan elemen dengan aturan akses yang sangat ketat. Struktur ini dianalogikan dengan antrean di dunia nyata, seperti antrean pembelian tiket, di mana elemen yang masuk pertama akan keluar (diproses) pertama pula. Prinsip dasar yang wajib dipatuhi oleh Queue adalah FIFO (First In, First Out). Konsep FIFO ini membatasi operasi pada Queue menjadi dua operasi utama: penambahan elemen (Enqueue atau Insert) yang selalu dilakukan di bagian belakang (Tail), dan penghapusan elemen (Dequeue atau Delete) yang selalu dilakukan di bagian depan (Head). Mengimplementasikan Queue dapat dilakukan menggunakan array statis (circular buffer) atau linked list dinamis.

Dalam implementasi menggunakan linked list, Queue dikelola melalui dua buah penunjuk (pointer), yaitu Head dan Tail. Penunjuk Head merujuk pada elemen pertama (terdepan) dari antrean, yang menjadi lokasi untuk operasi Dequeue. Sementara itu, penunjuk Tail merujuk pada elemen terakhir (terbelakang), yang menjadi lokasi untuk operasi Enqueue. Implementasi linked list ini menawarkan fleksibilitas yang lebih besar karena tidak memiliki batasan ukuran tetap (tidak terbatas pada NMax) dan efisien dalam manajemen memori, di mana alokasi dan dealokasi memori dilakukan secara dinamis seiring dengan bertambah atau berkurangnya elemen dalam antrean. Primitif-primitif lain yang esensial dalam ADT Queue mencakup fungsi untuk memeriksa apakah antrean kosong (IsEmpty) dan fungsi untuk membuat antrean kosong (CreateQueue).

.

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

Guided

- queue.h

```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
```

*Untitl...
File Edit Format View Help
Faris Walid Awwal Aidi
103112430133
Windows (CRLF) UTF-8

- queue.cpp

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

using namespace std;

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

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE;
}

void enqueue(Queue &Q, int x) {
    if (!isFull(Q)) {
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.info[Q.tail] = x;
        Q.count++;
    } else {
        cout << "Antrean Penuh! " << endl;
    }
}
```

*Untitl...
File Edit Format View Help
Faris Walid Awwal Aidi
103112430133
Windows (CRLF) UTF-8

```cpp
int dequeue(Queue &Q) {
    if (!isEmpty(Q)) {
        int x = Q.info[Q.head];
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count--;
        return x;
    } else {
        cout << "Antrean Kosong! " << endl;
        return -1;
    }
}

void printInfo(Queue Q) {
    if (isEmpty(Q)) {
        cout << "Isi Antrean: [Kosong]" << endl;
        return;
    }

    cout << "Isi Antrean: [";

    int i = Q.head;
    for (int n = 0; n < Q.count; n++) {
        cout << Q.info[i];
        if (n < Q.count - 1) cout << ", ";
        i = (i + 1) % MAX_QUEUE;
    }

    cout << "]" << endl;
}
```

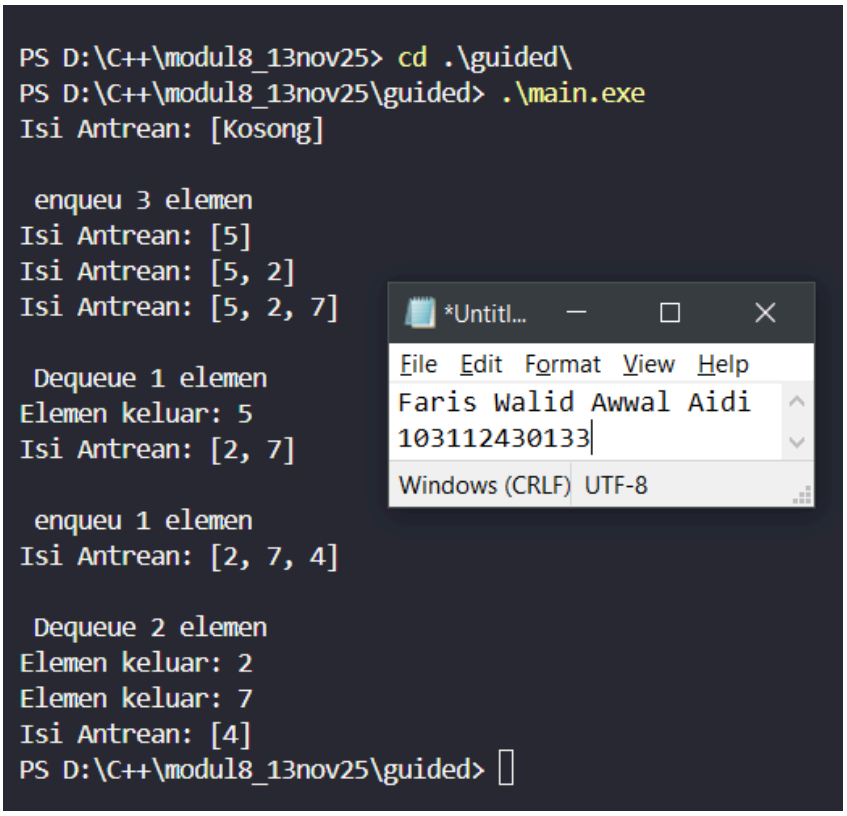
*Untitl...
File Edit Format View Help
Faris Walid Awwal Aidi
103112430133
Windows (CRLF) UTF-8

- main.cpp

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

using namespace std;

int main() {
    Queue Q;

    createQueue(Q);
    printInfo(Q);

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

    cout <<"\n Dequeue 1 elemen"<<endl;

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

    cout <<"\n enqueu 1 elemen"<<endl;
    enqueue(Q, 4);
    printInfo(Q);

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

    return 0;

}
```

*Untitl...
File Edit Format View Help
Faris Walid Awwal Aidi
103112430133
Windows (CRLF) UTF-8

Screenshots Output



Deskripsi:

Program ini merupakan implementasi dari struktur data Queue (Antrean) menggunakan Circular Array dengan kapasitas maksimum 5 elemen. Implementasi dibagi menjadi tiga file: queue.h mendefinisikan struktur Queue (yang berisi array info, serta variabel head, tail, dan count) dan mendeklarasikan semua fungsi primitif seperti createQueue, isEmpty, isFull, enqueue (memasukkan data), dequeue (mengambil data), dan printInfo. File queue.cpp berisi logika utama dari semua fungsi tersebut, khususnya menggunakan operasi modulo (% MAX_QUEUE) pada penunjuk head dan tail untuk menciptakan efek melingkar, yang memungkinkan ruang yang telah dikosongkan untuk digunakan kembali, sesuai dengan prinsip FIFO. Terakhir, file main.cpp berfungsi sebagai program uji coba yang mendemonstrasikan operasi enqueue dan dequeue secara berurutan, menunjukkan bagaimana data masuk dan keluar dari antrean, serta menampilkan status antrean saat kosong, terisi sebagian, dan setelah elemen dihapus.

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

Unguided

- queue.h

unguided > C queue.h > Queue

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

*Untitl...

File Edit Format View Help

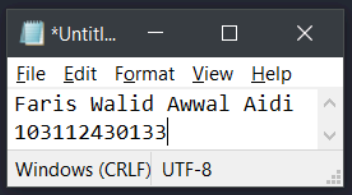
Faris Walid Awwal Aidi
103112430133

Windows (CRLF) UTF-8

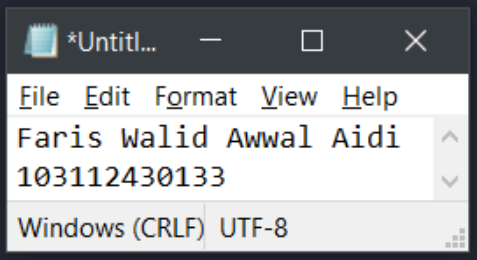
- queue.cpp (Head diam, Tail bergerak)

unguided > queue1.cpp > ...

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

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

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return Q.tail == MAX_QUEUE - 1;
}

void enqueue(Queue &Q, infotype x) {
    if (isFull(Q)) {
        cout << "Antrean Penuh! Tidak dapat menambahkan elemen " << x << endl;
        return;
    }

    if (isEmpty(Q)) {
        Q.head = 0;
    }

    Q.tail++;
    Q.info[Q.tail] = x;
    Q.count++;
}

infotype dequeue(Queue &Q) {
    infotype x = -1;

```



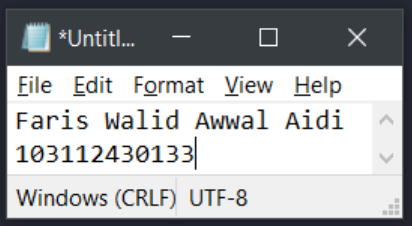
```
    if (isEmpty(Q)) {
        cout << "Antrean Kosong! " << endl;
        return x;
    }

    x = Q.info[Q.head];

    for (int i = Q.head + 1; i <= Q.tail; i++) {
        Q.info[i - 1] = Q.info[i];
    }

    Q.tail--;
    Q.count--;

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

    return x;
}

void printInfo(Queue Q) {
    if (isEmpty(Q)) {
        cout << "[empty queue]" << endl;
        return;
    }

    cout << "[";
    for (int i = Q.head; i <= Q.tail; i++) {
        cout << Q.info[i];
        if (i < Q.tail) {
            cout << ", ";
        }
    }
    cout << "]" << endl;
}
```



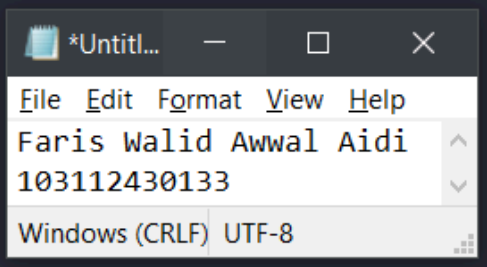
- queue.cpp (Head dan Tail bergerak)

unguided > queue2.cpp > printInfo(Queue)

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

using namespace std;

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

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return (Q.head == 0 && Q.tail == MAX_QUEUE - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFull(Q)) {
        cout << "Antrean Penuh Semu! Harus digeser dulu." << endl;
        return;
    }

    if (isEmpty(Q)) {
        Q.head = 0;
        Q.tail = 0;
    } else {
        Q.tail++;
    }

    Q.info[Q.tail] = x;
    Q.count++;
}

```



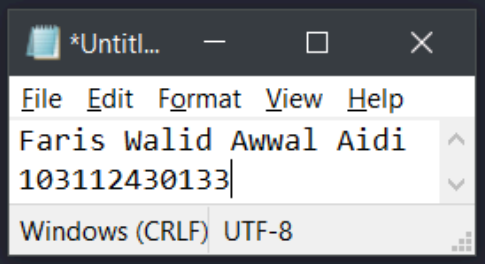
```cpp
infotype dequeue(Queue &Q) {
    infotype x = -1;

    if (isEmpty(Q)) {
        cout << "Antrean Kosong! " << endl;
        return x;
    }

    x = Q.info[Q.head];

    Q.head++;
    Q.count--;

    if (Q.head > Q.tail) {
        createQueue(Q);
    }

    return x;
}

void printInfo(Queue Q) {
    if (isEmpty(Q)) {
        cout << "[empty queue]" << endl;
        return;
    }

    cout << "[";
    for (int i = Q.head; i <= Q.tail; i++) {
        cout << Q.info[i];
        if (i < Q.tail) {
            cout << ", ";
        }
    }
    cout << "]" << endl;
}
```



- queue.cpp (Head dan Tail berputar)

unguided > queue3.cpp > printInfo(Queue)

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

using namespace std;

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

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE;
}

void enqueue(Queue &Q, infotype x) {
    if (isFull(Q)) {
        cout << "Antrean Penuh! " << endl;
        return;
    }

    if (isEmpty(Q)) {
        Q.head = 0;
    }

    Q.tail = (Q.tail + 1) % MAX_QUEUE;
    Q.info[Q.tail] = x;
    Q.count++;
}

infotype dequeue(Queue &Q) {
    infotype x = -1;

```



```
    if (isEmpty(Q)) {
        cout << "Antrean Kosong! " << endl;
        return x;
    }

    x = Q.info[Q.head];

    Q.head = (Q.head + 1) % MAX_QUEUE;
    Q.count--;

    if (Q.count == 0) {
        createQueue(Q);
    }

    return x;
}

void printInfo(Queue Q) {
    if (isEmpty(Q)) {
        cout << "[empty queue]" << endl;
        return;
    }

    cout << "[";
    int i = Q.head;

    for (int n = 0; n < Q.count; n++) {
        cout << Q.info[i];
        if (n < Q.count - 1) {
            cout << ", ";
        }
        i = (i + 1) % MAX_QUEUE;
    }
    cout << "]" << endl;
}
```

*Untitl...

File Edit Format View Help

Faris Walid Awwal Aidi
103112430133

Windows (CRLF) UTF-8

- main.cpp

unguided > main.cpp > main()

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

using namespace std;

int main() {
    Queue Q;

    cout << "Hello World!" << endl;

    createQueue(Q);
    cout << "H | T | Count | Queue Info" << endl;
    cout << "-----------------------------------" << endl;

    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "-----------------------------------" << endl;

    cout << "enqueue (Q,5)" << endl;
    enqueue(Q, 5);
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "enqueue (Q,2)" << endl;
    enqueue(Q, 2);
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "enqueue (Q,7)" << endl;
    enqueue(Q, 7);
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "dequeue (Q) -> " << dequeue(Q) << endl;
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "enqueue (Q,4)" << endl;
    enqueue(Q, 4);
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "dequeue (Q) -> " << dequeue(Q) << endl;
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    cout << "dequeue (Q) -> " << dequeue(Q) << endl;
    cout << Q.head << " | " << Q.tail << " | " << Q.count << " | ";
    printInfo(Q);

    return 0;
}
```

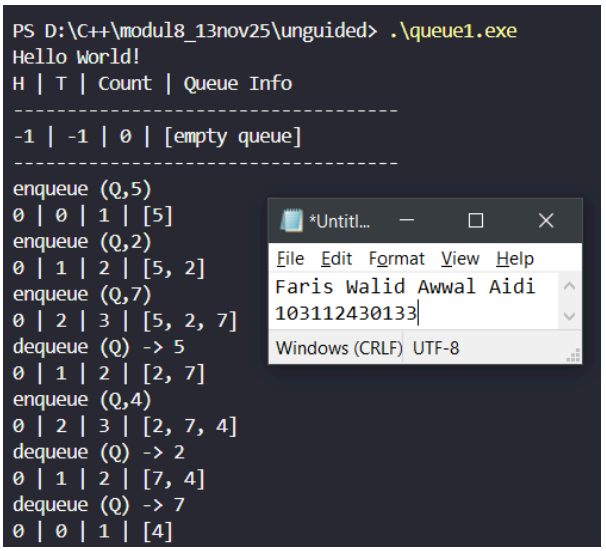
*Untitl...

File Edit Format View Help

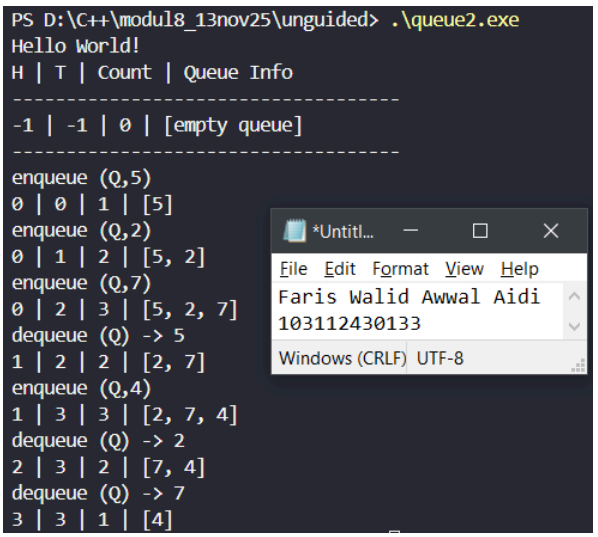
Faris Walid Awwal Aidi
103112430133

Windows (CRLF) UTF-8

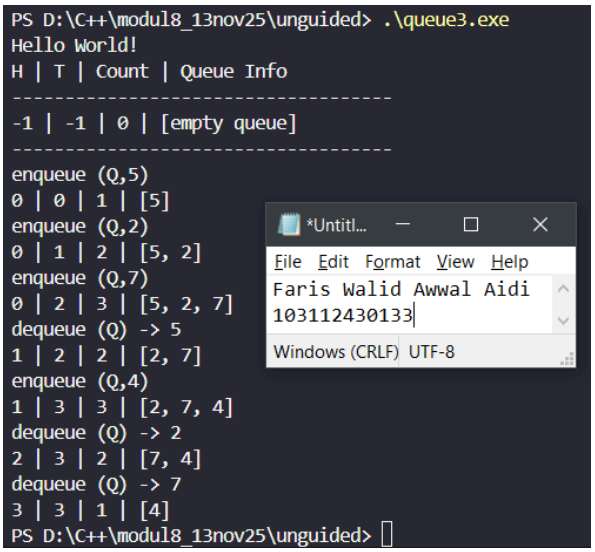
Screenshots Output (Head diam, Tail bergerak)



Screenshots Output (Head dan Tail bergerak)



Screenshots Output (Head dan Tail berputar)



Deskripsi:

Program ini merupakan implementasi dari Abstract Data Type (ADT) Queue atau Antrean menggunakan struktur data Array statis dengan kapasitas maksimum 5 elemen. Program ini terbagi menjadi tiga file, di mana queue.h berfungsi sebagai definisi struktur utama Queue (termasuk head, tail, dan count) dan deklarasi fungsi-fungsi primitifnya (enqueue, dequeue, dll.). Inti dari praktikum ini terletak pada tiga versi queue.cpp yang menyajikan mekanisme operasional Queue yang berbeda-beda: Alternatif 1 yang menggeser semua elemen ke depan saat dequeue (membuat head selalu di nol), Alternatif 2 yang membiarkan head dan tail bergerak linear tetapi berpotensi menciptakan kondisi "penuh semu," serta Alternatif 3 yang menggunakan operator modulo (%) untuk menciptakan Circular Array yang efisien dalam manajemen ruang memori. File main.cpp kemudian berfungsi sebagai test case utama yang menjalankan urutan operasi enqueue dan dequeue yang sama untuk mendemonstrasikan bagaimana setiap mekanisme mengelola antrean sesuai dengan prinsip FIFO (First In, First Out).

D. Kesimpulan

Kesimpulan Praktikum implementasi struktur data Queue (Antrean) menggunakan array statis telah berhasil dilakukan dengan menguji tiga mekanisme pergerakan Head dan Tail berdasarkan prinsip FIFO (First In, First Out). Alternatif 1 yang menetapkan Head tetap di indeks 0 dan melakukan penggeseran elemen saat Dequeue terbukti sebagai metode yang paling tidak efisien karena membutuhkan kompleksitas waktu O(N) untuk setiap operasi penghapusan. Alternatif 2 meningkatkan efisiensi dengan membiarkan Head bergerak maju, namun pendekatan ini menimbulkan kondisi "penuh semu," di mana antrean dianggap penuh meskipun masih ada ruang kosong di awal array yang tidak bisa diakses. Implementasi paling unggul adalah Alternatif 3 (Circular Array), yang memanfaatkan operasi modulo (%) pada penunjuk Head dan Tail, memungkinkan array berputar dan menggunakan kembali ruang yang dikosongkan secara efisien, sehingga mencapai kompleksitas waktu O(1) untuk Enqueue dan Dequeue dan secara optimal memanfaatkan kapasitas array.

Berdasarkan hasil pengujian, disarankan agar dalam pengembangan aplikasi nyata, Alternatif 3 (Circular Array) menjadi pilihan utama dalam mengimplementasikan Queue berbasis array karena menawarkan efisiensi waktu dan optimalisasi penggunaan memori. Meskipun Alternatif 2 juga memiliki kompleksitas O(1), kondisi "penuh semu" mengharuskan adanya proses penggeseran elemen secara periodik (re-indexing) untuk mengklaim kembali memori yang terbuang. Untuk mengatasi keterbatasan kapasitas array statis, langkah selanjutnya dapat mencakup implementasi Queue menggunakan Linked List, yang secara inheren menyediakan kapasitas tak terbatas dan manajemen memori yang lebih dinamis.

Referensi

Raharjo, Budi. 2025. *Buku Pemrograman C++ Mudah dan Cepat Menjadi Master C.*

Wikipedia contributors. (2024, 8 Mei). C++. Wikipedia, Ensiklopedia Bebas. Diakses pada 2 Oktober 2025, dari https://id.wikipedia.org/wiki/C%2B%2B