

**LAPORAN PRAKTIKUM**  
**Modul VII**  
**“STACK”**



**Disusun Oleh:**  
**Zivana Afra Yulianto -2211104039**  
**SE-07-02**

**Dosen:**  
**Wahyu Andi Saputra**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY**  
**PURWOKERTO**  
**2024**

## 1. Tujuan

Tujuan dari praktikum ini adalah untuk memahami dan mengimplementasikan struktur data stack menggunakan bahasa pemrograman C++. Praktikum ini bertujuan untuk:

1. Mempelajari konsep dasar dari struktur data stack (Last In First Out, LIFO).
2. Mengimplementasikan operasi dasar stack seperti push, pop, peek, dan display.
3. Memahami bagaimana stack dapat diimplementasikan menggunakan array dan linked list.
4. Mengaplikasikan pengetahuan struktur data dalam pemrograman menggunakan C++ untuk menyelesaikan masalah yang dapat dipecahkan dengan stack.

## 2. Landasan Teori

### 2.1 Pengertian Stack

Stack adalah sebuah struktur data yang mengikuti prinsip Last In First Out (LIFO), yang berarti elemen yang terakhir kali dimasukkan adalah elemen pertama yang akan dikeluarkan. Stack digunakan untuk menyimpan data sementara dan sering digunakan dalam proses seperti pemanggilan fungsi (function calls), pemeriksaan ekspresi matematika, dan algoritma pencarian.

### 2.2 Operasi Dasar Stack

Beberapa operasi dasar yang umum dilakukan pada stack antara lain:

- Push: Menambahkan elemen ke atas stack.
- Pop: Menghapus dan mengembalikan elemen teratas stack.
- Peek: Melihat elemen teratas stack tanpa menghapusnya.
- isEmpty: Mengecek apakah stack kosong.
- Display: Menampilkan seluruh elemen dalam stack.

### 2.3 Implementasi Stack

Stack dapat diimplementasikan menggunakan dua pendekatan utama:

- Array: Menggunakan array statis untuk menyimpan elemen stack. Kelemahannya adalah terbatasnya ukuran stack.
- Linked List: Menggunakan struktur data dinamis (linked list) untuk menyimpan elemen stack. Pendekatan ini lebih fleksibel karena ukuran stack dapat bertambah seiring waktu.

### 3. Guided

#### GUIDED 1 :

```
#include <iostream>
#define MAX 100
using namespace std;

class stack
{
private:
    int top;
    int arr[MAX];

public:
    stack()
    {
        top = -1;
    }

    bool isFull()
    {
        return top == MAX - 1;
    }

    bool isEmpty()
    {
        return top == -1;
    }

    void push(int x)
    {
        if (isFull())
        {
            cout << "Stack Overflow" << endl;
            return;
        }
        arr[++top] = x;
    }

    int pop()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return -1;
        }
        return arr[top--];
    }

    int peek()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return -1;
        }
        return arr[top];
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return;
        }
        for (int i = top; i >= 0; i--)
        {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main()
{
    stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.display();
    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";
    return 0;
}
```

Screenshoot output :

```
30 20 10
Stack elements: 30 20 10
Top element: 30
PS C:\STD_Zivana_Afra_Yulianto>
```

Deskripsi :

- Kelas stack memiliki array arr[MAX] untuk menyimpan elemen dan variabel top yang menunjukkan posisi elemen teratas.
- Fungsi isFull mengecek apakah stack penuh, sementara isEmpty mengecek apakah stack kosong.
- Fungsi push menambah elemen ke stack, sedangkan pop menghapus elemen teratas.
- Fungsi peek menampilkan elemen teratas tanpa menghapusnya, dan fungsi display menampilkan seluruh isi stack dari teratas ke terbawah.

## GUIDED 2

```
#include <iostream>

using namespace std;

class Node
{
public:
    int data;
    Node *next;
    Node(int value)
    {
        data = value;
        next = nullptr;
    }
};

class Stack
{
private:
    Node *top;

public:
    Stack()
    {
        top = nullptr;
    }

    bool isEmpty()
    {
        return top == nullptr;
    }

    void push(int x)
    {
        Node *newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }

    void pop()
    {
        if (isEmpty())
        {
            cout << "Stack underflow\n";
            return;
        }
        Node *temp = top;
        top = top->next;
        delete temp;
    }

    int peek()
    {
        if (isEmpty())
        {
            cout << "Stack is empty\n";
            return -1;
        }
        return top->data;
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Stack is empty\n";
            return;
        }
        Node *current = top;
        while (current != nullptr)
        {
            cout << current->data << " ";
            current = current->next;
        }
        cout << "\n";
    }
};

int main()
{
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    cout << "Stack after pop: ";
    s.display();

    return 0;
}
```

Screenshoot :

```
Stack elements: 30 20 10
Top element: 30
Stack after pop: 20 10
PS C:\STD_Zivana_Afra_Yulianto>
```

Deskripsi :

- Fungsi `isEmpty` mengecek apakah stack kosong.
- Fungsi `push` menambahkan elemen ke stack dengan membuat node baru dan menghubungkannya ke node sebelumnya.
- Fungsi `pop` menghapus elemen teratas dari stack.
- Fungsi `peek` menampilkan elemen teratas tanpa menghapusnya.
- Fungsi `display` menampilkan semua elemen stack dari teratas hingga terbawah.

## 4. Unguided

### UNGUIDED 1

```
#include <iostream>
#include <stack>
#include <algorithm> // untuk transform
using namespace std;

// Fungsi untuk memeriksa apakah kalimat adalah palindrom
bool isPalindrome(string sentence)
{
    // Normalisasi kalimat: hapus spasi dan ubah ke huruf kecil
    string normalized_sentence = "";
    for (char c : sentence)
    {
        if (c != ' ')
        { // Abaikan spasi
            normalized_sentence += tolower(c);
        }
    }

    // Gunakan stack untuk membalikkan string
    stack<char> s;
    for (char c : normalized_sentence)
    {
        s.push(c);
    }

    // Bangun string terbalik dari stack
    string reversed_sentence = "";
    while (!s.empty())
    {
        reversed_sentence += s.top();
        s.pop();
    }

    // Periksa apakah string asli sama dengan versi terbalikanya
    return normalized_sentence == reversed_sentence;
}

int main()
{
    string sentence;

    // Input langsung menggunakan getline
    cout << "Masukkan kalimat: ";
    getline(cin, sentence); // Membaca seluruh kalimat dengan spasi
```

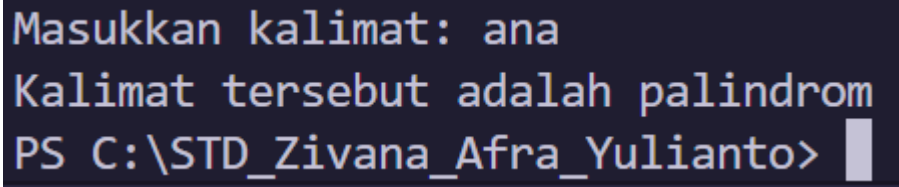
```

// Hasil
if (isPalindrome(sentence))
{
    cout << "Kalimat tersebut adalah palindrom" << endl;
}
else
{
    cout << "Kalimat tersebut bukan palindrom" << endl;
}

return 0;
}

```

Screenshoot output :



```

Masukkan kalimat: ana
Kalimat tersebut adalah palindrom
PS C:\STD_Zivana_Afra_Yulianto>

```

Deskripsi :

- Input Kalimat:
  - Pengguna memasukkan kalimat melalui input.
  - Input dibaca menggunakan `getline()` agar seluruh kalimat termasuk spasi diterima.
- Normalisasi Teks:
  - Spasi dihapus, dan huruf diubah menjadi huruf kecil menggunakan perulangan `for`.
- Pembalikan Teks Menggunakan Stack:
  - Setiap karakter dari kalimat yang telah dinormalisasi dimasukkan ke dalam stack.
  - Karena stack bekerja dengan prinsip Last In, First Out (LIFO), saat karakter diambil dari stack, teks akan terbalik.
- Perbandingan:
  - Teks asli yang dinormalisasi dibandingkan dengan teks yang sudah dibalik.
  - Jika kedua teks sama, maka kalimat tersebut adalah palindrom.
- Output Hasil:
  - Program mencetak pesan yang menyatakan apakah kalimat tersebut palindrom atau bukan.

## UNGUIDED 2

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

// Fungsi untuk membalikkan seluruh kalimat
string reverseSentence(string sentence)
{
    stack<char> s;
    string reversed_sentence = "";

    // Masukkan setiap karakter kalimat ke dalam stack
    for (char c : sentence)
    {
        s.push(c);
    }

    // Ambil karakter dari stack untuk membangun kalimat terbalik
    while (!s.empty())
    {
        reversed_sentence += s.top();
        s.pop();
    }

    return reversed_sentence;
}

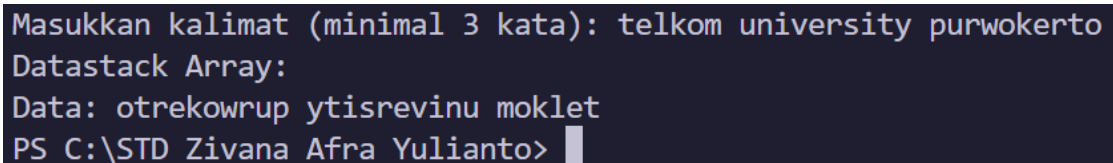
int main()
{
    string sentence;

    // Input kalimat dari pengguna
    cout << "Masukkan kalimat (minimal 3 kata): ";
    getline(cin, sentence);

    // Proses dan tampilkan hasil
    string reversed_sentence = reverseSentence(sentence);
    cout << "Datastack Array:" << endl;
    cout << "Data: " << reversed_sentence << endl;

    return 0;
}
```

Screenshoot output :

A screenshot of a terminal window showing the output of the C++ program. The text is as follows:  
Masukkan kalimat (minimal 3 kata): telkom university purwokerto  
Datastack Array:  
Data: otrekowrup ytisrevinu moklet  
PS C:\STD\_Zivana\_Afra\_Yulianto>

Deskripsi :

- Input Kalimat:
  - Pengguna diminta untuk memasukkan sebuah kalimat minimal 3 kata menggunakan `getline()` agar seluruh kalimat (termasuk spasi) dapat diterima.
- Proses Pembalikan dengan Stack:
  - Setiap karakter dalam kalimat (termasuk spasi) dimasukkan ke dalam stack menggunakan `push()`.
  - Karena stack menggunakan prinsip Last In, First Out (LIFO), karakter yang dimasukkan pertama akan keluar terakhir, sehingga menghasilkan urutan terbalik.



- Membangun Kalimat Terbalik:
  - Karakter diambil dari stack satu per satu menggunakan `s.top()` dan dihapus dari stack menggunakan `s.pop()`.
  - Setiap karakter yang diambil ditambahkan ke variabel `reversed_sentence` untuk membangun kalimat dalam urutan terbalik.
- Output Hasil:
  - Kalimat yang sudah terbalik ditampilkan kepada pengguna.

## 5. Kesimpulan

Berdasarkan praktikum ini, dapat disimpulkan bahwa:

1. Struktur data stack dapat diimplementasikan dengan dua cara, yaitu menggunakan array atau linked list. Keduanya memiliki kelebihan dan kekurangan masing-masing.
2. Penggunaan stack sangat berguna dalam berbagai aplikasi pemrograman, terutama yang membutuhkan pemrosesan data secara LIFO, seperti dalam implementasi fungsi rekursif dan pemeriksaan ekspresi.
3. Melalui praktikum ini, pemahaman tentang implementasi struktur data stack dalam C++ dapat diperoleh dengan baik, serta bagaimana mengimplementasikan operasi dasar stack menggunakan kedua metode (array dan linked list).
4. Praktikum ini memberikan pengalaman praktis dalam pemrograman C++ yang dapat diterapkan dalam berbagai proyek perangkat lunak.