

**LAPORAN PRAKTIKUM  
MODUL 7  
STACK**



**Disusun Oleh :  
Fauzan Rofif Ardiyanto/2211104036  
S1SE06 2**

**Asisten Praktikum :  
Aldi Putra  
Andini Nur Hidayah**

**Dosen Pengampu :  
Wahyu Andi Saputra**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK  
FAKUTLAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2024**

## A. TUJUAN PRAKTIKUM

- Mampu memahami konsep stack pada struktur data dan algoritma
- Mampu mengimplementasikan operasi-operasi pada stack
- Mampu memecahkan permasalahan dengan solusi stack

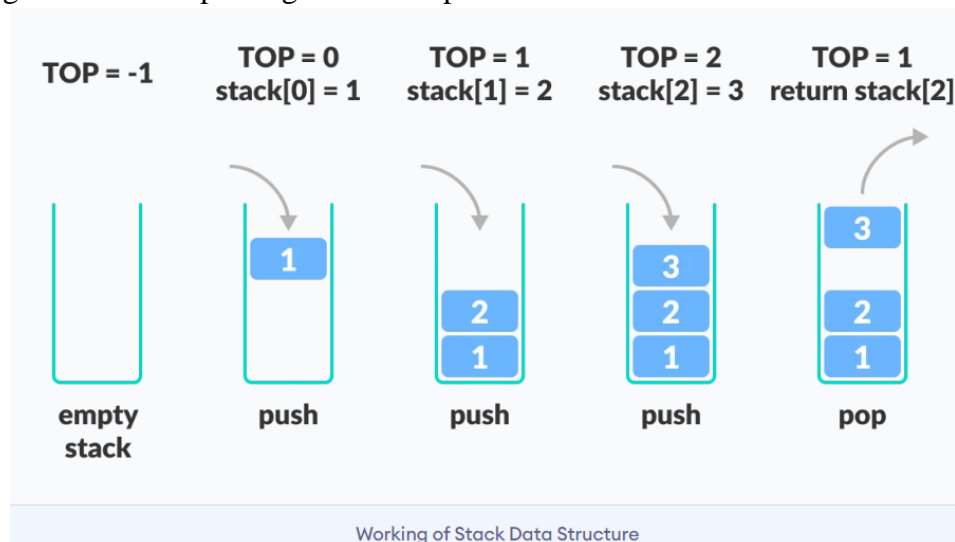
## B. DASAR TEORI

Sebuah stack atau tumpukan merupakan struktur data yang berfungsi untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out (LIFO). Analogi yang sering digunakan adalah tumpukan piring di kafetaria, di mana piring terakhir yang ditambahkan akan menjadi yang pertama diambil.

Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat dalam berbagai aplikasi, termasuk manajemen memori komputer, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Sebagai contoh, dalam manajemen memori, stack digunakan untuk menyimpan alamat-alamat memori yang dialokasikan untuk variabel dan fungsi.

Dengan prinsip LIFO ini, stack memungkinkan akses data dengan efisiensi, di mana elemen yang terakhir dimasukkan akan menjadi yang pertama diambil. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak dan pemrograman komputer..



Operasi pada stack melibatkan beberapa fungsi dasar yang dapat dilakukan pada struktur data ini. Berikut adalah beberapa operasi umum pada stack:

- Push (Masukkan):** Menambahkan elemen ke dalam tumpukan pada posisi paling atas atau ujung.

- b. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan.
- c. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya.
- d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.
- e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas).
- f. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan.
- g. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa menghapusnya.
- h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
- i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

### C. GUIDED 1

#### Sourcecode

```
#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;
}

```

### Output

```

PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output> & .\'pert7.exe'
30 20 10
Stack elements: 30 20 10
Top element: 30
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output>

```

### 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();

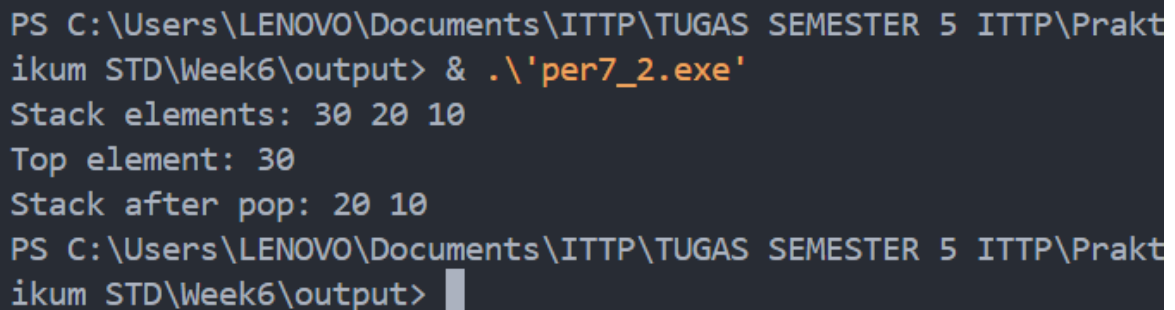
    cout << "Top element: " << s.peek() << "\n";

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

    return 0;
}

```

### Output



```

PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output> & .\per7_2.exe
Stack elements: 30 20 10
Top element: 30
Stack after pop: 20 10
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output>

```

### D. UNGUIDED

1. Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

```

2. #include <iostream>
3. #include <stack>
4. #include <algorithm> // untuk transform

```

```

5. using namespace std;
6.
7. // Fungsi untuk memeriksa apakah kalimat adalah palindrom
8. bool isPalindrome(string sentence)
9. {
10.     // Normalisasi kalimat: hapus spasi dan ubah ke huruf kecil
11.     string normalized_sentence = "";
12.     for (char c : sentence)
13.     {
14.         if (c != ' ')
15.         { // Abaikan spasi
16.             normalized_sentence += tolower(c);
17.         }
18.     }
19.
20.     // Gunakan stack untuk membalikkan string
21.     stack<char> s;
22.     for (char c : normalized_sentence)
23.     {
24.         s.push(c);
25.     }
26.
27.     // Bangun string terbalik dari stack
28.     string reversed_sentence = "";
29.     while (!s.empty())
30.     {
31.         reversed_sentence += s.top();
32.         s.pop();
33.     }
34.
35.     // Periksa apakah string asli sama dengan versi terbalikny
36.     return normalized_sentence == reversed_sentence;
37. }
38.
39. int main()
40. {
41.     string sentence;
42.
43.     // Input langsung menggunakan getline
44.     cout << "Masukkan kalimat: ";
45.     getline(cin, sentence); // Membaca seluruh kalimat dengan
    spasi
46.
47.     // Hasil
48.     if (isPalindrome(sentence))
49.     {
50.         cout << "Kalimat tersebut adalah palindrom" << endl;
51.     }

```



```

52.     else
53.     {
54.         cout << "Kalimat tersebut bukan palindrom" << endl;
55.     }
56.
57.     return 0;
58.}

```

Output

```

PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output> & .\'un1.exe'
Masukkan kalimat: gue
Kalimat tersebut bukan palindrom
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\Week6\output>

```

59. Buatlah program untuk melakukan pembalikan terhadap kalimat menggunakan stack dengan minimal 3 kata. Jelaskan output program dan source codenya beserta operasi/fungsi yang dibuat?

```

60.#include <iostream>
61.#include <stack>
62.using namespace std;
63.
64.// Fungsi untuk membalikkan seluruh kalimat
65.string reverseSentence(string sentence)
66.{
67.    stack<char> s;
68.    string reversed_sentence = "";
69.
70.    // Masukkan setiap karakter kalimat ke dalam stack
71.    for (char c : sentence)
72.    {
73.        s.push(c);
74.    }
75.
76.    // Ambil karakter dari stack untuk membangun kalimat terbalik
77.    while (!s.empty())
78.    {
79.        reversed_sentence += s.top();
80.        s.pop();
81.    }
82.
83.    return reversed_sentence;
84.}
85.
86.int main()
87.{

```

```

88.     string sentence;
89.
90.     // Input kalimat dari pengguna
91.     cout << "Masukkan kalimat (minimal 3 kata): ";
92.     getline(cin, sentence);
93.
94.     // Proses dan tampilkan hasil
95.     string reversed_sentence = reverseSentence(sentence);
96.     cout << "Datastack Array:" << endl;
97.     cout << "Data: " << reversed_sentence << endl;
98.
99.     return 0;
100.    }

```

Output

```

SEMESTER 5 ITTP\Praktikum STD\Week6\output'
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Prakt
ikum STD\Week6\output> & .\'un2.exe'
Masukkan kalimat (minimal 3 kata): aku sayang kamu
Datastack Array:
Data: umak gnayas uka
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Prakt
ikum STD\Week6\output>

```