

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



**Disusun Oleh:**  
**Dewi Atika Muthi -2211104042**  
**SE-07-02**

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

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

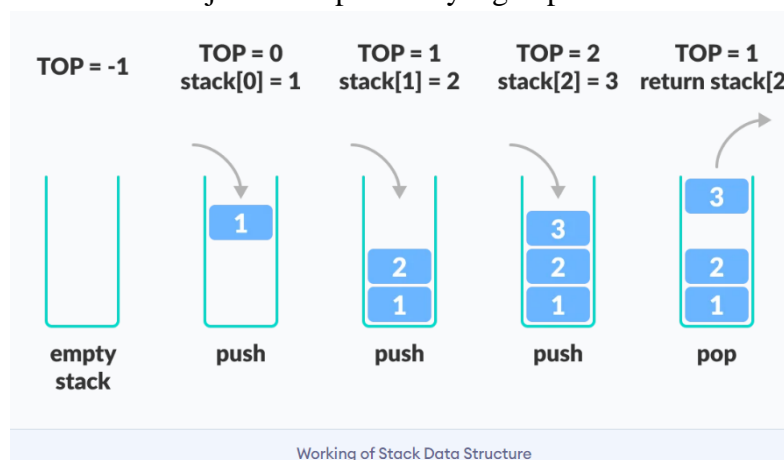
## 1. Tujuan

Berikut tujuan praktikum pada modul ini adalah:

- Mengimplementasikan queue dengan menggunakan pointer dan table
- Mampu memahami konsep stack pada struktur data dan algoritma
- Mampu mengimplementasikan operasi-operasi pada stack
- Mampu memecahkan permasalahan dengan solusi stack

## 2. Landasan Teori

Stack merupakan struktur data yang menggunakan prinsip LIFO (Last In First Out) dimana elemen yang terakhir dimasukkan adalah elemen yang pertama keluar. Stack dapat dianalogikan seperti tumpukan buku - buku yang terakhir ditumpuk akan berada di atas dan menjadi buku pertama yang dapat diambil.



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

### 2.1 Komponen Stack

- TOP** - Pointer yang menunjuk ke elemen paling atas stack

```
// Deklarasi TOP pada stack
struct Stack {
    int top; // untuk array
    Node* top; // untuk linked list
};
```

b. **Info/Data** - Tempat penyimpanan nilai pada stack

```
// Deklarasi data pada stack array
int data[MAX_SIZE];

// Deklarasi data pada stack linked list
struct Node {
    int data;
    Node* next;
};
```

## 2.2 Operasi Dasar Stack

a. **PUSH** - Menambah elemen baru ke puncak stack

```
void push(Stack &S, int x) {
    if(!isFull(S)) {
        S.top++;
        S.data[S.top] = x;
    }
}
```

b. **POP** - Mengambil elemen dari puncak stack

```
int pop(Stack &S) {
    int x = -1;
    if(!isEmpty(S)) {
        x = S.data[S.top];
        S.top--;
    }
    return x;
}
```

## 2.3 Primitif-Primitif Stack

a. **createStack()** – Membuat stack kosong

```
void createStack(Stack &S) {
    S.top = -1; // Inisialisasi stack kosong
}
```

b. **isEmpty()** – Mengecek apakah stack kosong

```
bool isEmpty(Stack S) {
    return (S.top == -1);
}
```

c. **isFull()** – Mengecek apakah stack penuh (untuk implementasi array)

```
bool isFull(Stack S) {
    return (S.top == MAX_SIZE-1);
}
```

d. **peek()** – Melihat elemen teratas tanpa menghapusnya

```
int peek(Stack S) {
    if(!isEmpty(S)) {
        return S.data[S.top];
    }
    return -1;
}
```

e. **printStack()** – Menampilkana isi stack

```
int peek(Stack S) {
```

```

        if(!isEmpty(S)) {
            return S.data[S.top];
        }
        return -1;
    }
}

```

## 2.4 Representasi Stack

Stack dapat diimplementasikan dengan dua cara

### 1. Representasi Array

- Menggunakan array dengan ukuran tetap
- Akses elemen lebih cepat
- Keterbatasan ukuran maksimum

```

struct Stack {
    int data[MAX_SIZE];
    int top;
};

```

### 2. Representasi Linked List

- Menggunakan pointer/node
- Ukuran dinamis
- Penggunaan memori lebih efisien

```

struct Node {
    int data;
    Node* next;
};

struct Stack {
    Node* top;
};

```

## 3. 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\n";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
        if (isEmpty()) {

```

```

        cout << "Stack Underflow\n";
        return;
    }
    top--;
}

int peek() {
    if (!isEmpty()) {
        return arr[top];
    }
    cout << "Stack is empty\n";
    return -1; // Return a sentinel value
}

void display() {
    if (isEmpty()) {
        cout << "Stack is empty\n";
        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);

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

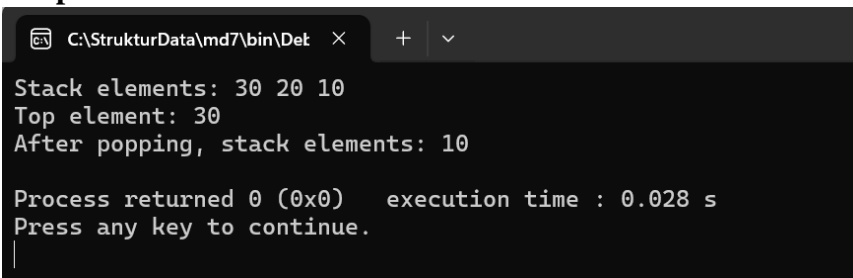
    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, stack elements: ";
    s.display();

    return 0;
}

```

### Output:



```

C:\StrukturData\md7\bin\Det  ×  +  ▾
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10

Process returned 0 (0x0)   execution time : 0.028 s
Press any key to continue.

```

### Penjelasan program:

Stack diimplementasikan menggunakan array dengan ukuran maksimal 100 (MAX)

Operasi-operasi dasar Stack: Push, pop, peek, isEmpty, isFull, display

Cara kerja program:

- Pertama push angka 10, 20, 30 ke dalam stack

- Menampilkan semua elemen (30 di atas karena LIFO - Last In First Out)
- Menampilkan elemen teratas (30)
- Melakukan pop 2 kali (menghapus 30 dan 20)
- Menampilkan sisa elemen (hanya tersisa 10)

## 2. Source code:

```
#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()) {
            return top->data;
        }
        cout << "Stack is empty\n";
        return -1; // Return a sentinel value
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        Node* current = top;
        while (current) {
            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 << "After popping, stack elements: ";
    s.display();

    return 0;
}
```

### Output:

```
C:\StrukturData\modul2_KAC/ x + v
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 20 10

Process returned 0 (0x0)   execution time : 0.137 s
Press any key to continue.
```

### Penjelasan program:

Program memiliki dua class utama:

- Node: Menyimpan data (integer) dan pointer ke node selanjutnya
- Stack: Mengatur operasi stack dengan pointer top

Operasi dasar Stack yang diimplementasikan: Push, pop, peek, isEmpty, display.

Alur kerja program:

- Push angka 10, 20, 30 secara berurutan
- Menampilkan semua elemen (30 di atas karena LIFO)
- Menampilkan elemen teratas (30)
- Pop satu elemen (menghapus 30)
- Menampilkan sisa elemen (20, 10)

## 4. 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.

### SourceCode

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

bool isPalindrom(string str) {
    stack<char> s;
```

```

string cleanStr = "";

// Membersihkan string dari spasi
for(char c : str) {
    if(c != ' ') cleanStr += tolower(c);
}

int length = cleanStr.length();
int mid = length / 2;

// Push setengah karakter pertama ke stack
for(int i = 0; i < mid; i++) {
    s.push(cleanStr[i]);
}

// Skip karakter tengah jika panjang string ganjil
if(length % 2 != 0) mid++;

// Bandingkan setengah karakter terakhir dengan stack
for(int i = mid; i < length; i++) {
    if(s.empty() || s.top() != cleanStr[i]) return false;
    s.pop();
}

return true;
}

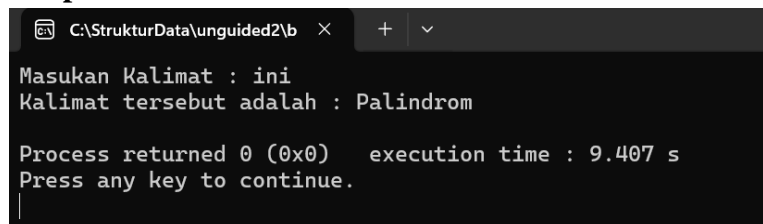
int main() {
    string kalimat;
    cout << "Masukan Kalimat : ";
    getline(cin, kalimat);

    cout << "Kalimat tersebut adalah : ";
    if(isPalindrom(kalimat))
        cout << "Palindrom" << endl;
    else
        cout << "Bukan Palindrom" << endl;

    return 0;
}

```

### Output:



```

C:\StrukturData\unguided2\b x + v
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom

Process returned 0 (0x0)   execution time : 9.407 s
Press any key to continue.

```

### Penjelasan Program:

- Program menggunakan stack untuk memeriksa palindrom
- Karakter pertama hingga tengah di-push ke stack
- Karakter setelah tengah dibandingkan dengan top stack
- Jika semua karakter cocok, maka palindrom



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

SourceCode:

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

class Stack {
private:
    char data[MAX];
    int top;

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

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

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

    void push(char x) {
        if (!isFull()) {
            data[++top] = x;
        }
    }

    char pop() {
        if (!isEmpty()) {
            return data[top--];
        }
        return '\0';
    }
};

void reverseString(string str) {
    Stack s;

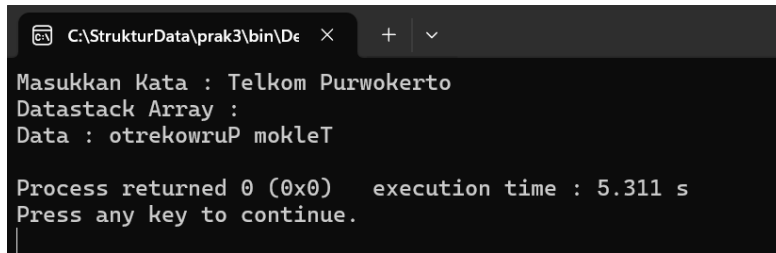
    // Push semua karakter ke stack
    for(int i = 0; i < str.length(); i++) {
        s.push(str[i]);
    }

    cout << "Datastack Array : " << endl;
    cout << "Data : ";

    // Pop dan cetak karakter dari stack
    while(!s.isEmpty()) {
        cout << s.pop();
    }
    cout << endl;
}
```

```
int main() {  
    string kalimat;  
    cout << "Masukkan Kata : ";  
    getline(cin, kalimat);  
  
    reverseString(kalimat);  
  
    return 0;  
}
```

### Output:

A screenshot of a terminal window with a dark background. The window title bar shows the file path 'C:\StrukturData\prak3\bin\De' and standard window controls. The terminal output shows the program's execution: it prompts 'Masukkan Kata :', receives the input 'Telkom Purwokerto', and then outputs 'Datastack Array : Data : otrekowruP mokleT'. At the bottom, it shows 'Process returned 0 (0x0) execution time : 5.311 s' and 'Press any key to continue.'

```
C:\StrukturData\prak3\bin\De x + v  
Masukkan Kata : Telkom Purwokerto  
Datastack Array :  
Data : otrekowruP mokleT  
  
Process returned 0 (0x0) execution time : 5.311 s  
Press any key to continue.
```

### Deskripsi program:

Fungsi reverseString:

- Menerima input string
- Push setiap karakter ke stack
- Pop semua karakter untuk mendapatkan string terbalik

## 5. Kesimpulan

Pemahaman yang baik tentang stack sangat penting dalam pengembangan software karena merupakan struktur data fundamental yang sering digunakan dalam berbagai algoritma dan aplikasi.