LAPORAN PRAKTIKUM

Modul 07

“STACK”

Disusun Oleh:

Faishal Arif Setiawan 231104066

Kelas:

SE 07 02

Dosen:

WAHYU ANDI SAPUTRA S.PD,.M.ENG.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO 2024

A. TUJUAN PRAKTIKUM

a. Mampu memahami konsep stack pada struktur data dan algoritma

b. Mampu mengimplementasikan operasi-operasi pada stack

c. Mampu memecahkan permasalahan dengan solusi stack

B.DASAR TEORI

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..

C.GUIDED

```cpp
#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;
    }

    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:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS D:\struktur data pemograman\guided5\modulstack\output>
```

```cpp
#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;
    }

    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 are: ";
    s.display();

    cout << "Top element is: " << s.peek() << endl;

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

    return 0;

}
```

Output:

```
Stack elements are: 30 20 10
Top element is: 30
Stack Underflow
After popping top element, stack elements are: 30 20 10
PS D:\struktur data pemograman\guided5\modulstack\output>
```

D.UNGUIDED

```cpp
#include <iostream>
#include <stack>
#include <string>
#include <algorithm> // Untuk fungsi transform

using namespace std;

bool isPalindrome(const string& str) {
    stack<char> s;
    string cleanStr;


    for (char c : str) {
        if (isalnum(c)) {
            cleanStr += tolower(c);
        }
    }


    for (char c : cleanStr) {
        s.push(c);
    }


    for (char c : cleanStr) {
        if (c != s.top()) {
            return false;
        }
        s.pop();
    }

    return true;
}

int main() {
    string input;

    cout << "Masukkan kalimat: ";
    getline(cin, input);

    if (isPalindrome(input)) {
        cout << "Kalimat tersebut adalah Palindrom" << endl;
    } else {
        cout << "Kalimat tersebut adalah bukan Palindrom" << endl;
    }

    return 0;
}

```

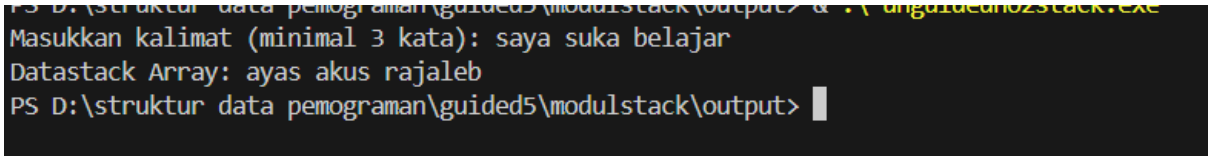
OUTPUT:

```
Masukkan kalimat: ini
Kalimat tersebut adalah Palindrom
```

```
Masukkan kalimat: telkom
Kalimat tersebut adalah bukan Palindrom
```

```cpp
#include <iostream>
#include <stack>
#include <sstream> // Untuk memisahkan kata-kata
#include <string>

using namespace std;

string reverseWord(const string& word) {
    stack<char> s;
    string reversedWord;

    // Memasukkan setiap karakter kata ke dalam stack
    for (char c : word) {
        s.push(c);
    }

    // Mengeluarkan karakter dari stack (membalik kata)
    while (!s.empty()) {
        reversedWord += s.top();
        s.pop();
    }

    return reversedWord;
}

string reverseSentence(const string& sentence) {
    stringstream ss(sentence); // Membantu memisahkan kata-kata
    string word;
    string result;

    // Membalik setiap kata dalam kalimat
    while (ss >> word) {
        result += reverseWord(word) + " "; // Membalikkan kata dan menambahkan spasi
    }

    return result;
}

int main() {
    string input;

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

    string reversedSentence = reverseSentence(input);

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

    return 0;
}

```

OUTPUT:



Penjelasan fungsi:

Fungsi reverseword yaitu membalikan suatu kata dengan memasukan setiap kata ke dalam stack

Fungsi reversesentence yaitu membaca setiap kata dalam kalimat menggunakan stringstream

Fungsi main yaitu mengambil input kalimat dari pengguna

Memanggil resersesentence untuk membalikan seluruh kata dalam kalimat dan menampilkan hasil

Penjelasan hasil:

Setiap kata dalam kalimat di balik menggunakan stack

Kata yang terbalik digabungkan kembali menjadi sebuah kalimat baru, dipisahkan dengan spasi.