LAPORAN PRAKTIKUM
PERTEMUAN 7
STACK

Nama :

Haza Zaidan Zidna Fann

(2311104056)

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Memahami konsep stack.

Mengimplementasikan operasi stack

Memecahkan masalah dengan stack.

II. LANDASAN TEORI

Stack

Stack adalah struktur data yang menerapkan prinsip Last-In, First-Out (LIFO), seperti tumpukan piring: elemen terakhir yang dimasukkan akan diambil pertama.

Operasi Dasar Stack pada Double Linked List

Push: Menambahkan elemen ke stack.

Pop: Menghapus elemen teratas dari stack.

Operasi Tambahan

Top: Melihat elemen teratas tanpa menghapusnya.

IsEmpty: Memeriksa apakah stack kosong.

IsFull: Memeriksa apakah stack penuh (tergantung implementasi).

Size: Menghitung jumlah elemen dalam stack.

Peek: Melihat elemen pada posisi tertentu tanpa menghapusnya.

Clear: Menghapus semua elemen dalam stack.

Search: Mencari elemen tertentu dalam stack.

Kegunaan Stack

Manajemen Memori: Menyimpan data sementara, seperti pemanggilan fungsi dan variabel lokal.

Evaluasi Ekspresi Aritmatika: Digunakan untuk parsing infix ke postfix/prefix dan evaluasi.

Manajemen Fungsi Rekursif: Stack menyimpan alamat pengembalian dan parameter fungsi.

Undo/Redo: Menyimpan tindakan terakhir pada aplikasi seperti editor teks.

Traversal Struktur Data: DFS pada graf atau pohon.

Implementasi Stack

Stack dapat diimplementasikan dengan:

Array: Lebih sederhana tetapi memiliki ukuran tetap.

Linked List: Dinamis, tidak ada batasan ukuran stack.

Double Linked List: Memungkinkan traversal ke elemen sebelumnya jika dibutuhkan.

III. GUIDED

```cpp
#include <iostream>

using namespace std;

// Mendefinisikan ukuran maksimum tumpukan
#define MAX 100

class Stack {
private:
    int arr[MAX];  // Array untuk menyimpan elemen tumpukan
    int top;       // Indeks elemen teratas

public:
    // Konstruktor: Inisialisasi indeks teratas ke -1 (tumpukan kosong)
    Stack() {
        top = -1;
    }

    // Mengecek apakah tumpukan penuh
    bool isFull() {
        return top == MAX - 1;
    }

    // Mengecek apakah tumpukan kosong
    bool isEmpty() {
        return top == -1;
    }

    // Menambahkan elemen ke tumpukan
    void push(int x) {
        if (isFull()) {
            cout << "Stack Overflow\n";
            return;
        }
        arr[++top] = x;
    }

    // Menghapus elemen teratas dari tumpukan
    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        top--;
    }

    // Mengembalikan elemen teratas tanpa menghapusnya
    int peek() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return -1; // Mengembalikan nilai sentinel jika tumpukan kosong
        }
        return arr[top];
    }

    // Menampilkan semua elemen tumpukan
    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

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

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

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

    s.pop();
    s.pop();

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

    return 0;
}
```

```
stack elements: 302010
top element: 30
after popping, stack elements: 10
```

Kelas Stack:
Atribut:
arr[MAX]: Array untuk menyimpan elemen stack, dengan ukuran maksimum MAX (100).
top: Indeks elemen teratas di stack, diinisialisasi dengan -1 (stack kosong).

Metode:
isFull(): Memeriksa apakah stack sudah penuh (jika top == MAX - 1).
isEmpty(): Memeriksa apakah stack kosong (jika top == -1).
push(int x): Menambahkan elemen ke stack jika tidak penuh. Jika penuh, mencetak pesan "Stack Overflow".
pop(): Menghapus elemen teratas jika stack tidak kosong. Jika kosong, mencetak pesan "Stack Underflow".
peek(): Mengembalikan elemen teratas tanpa menghapusnya. Jika kosong, mengembalikan -1 dan mencetak pesan "Stack is empty".
display(): Menampilkan elemen stack dari teratas ke dasar. Jika kosong, mencetak pesan "Stack is empty".

Fungsi main():
Membuat objek stack (Stack s).
Menambahkan tiga elemen ke stack (push(10), push(20), push(30)).
Menampilkan elemen stack dengan display().
Menampilkan elemen teratas dengan peek().
Menghapus dua elemen teratas menggunakan pop().
Menampilkan elemen stack setelah penghapusan.

Alur Program:
Push menambahkan elemen ke stack.
Display mencetak elemen stack: 30 20 10.
Peek menampilkan elemen teratas: 30.
Pop menghapus elemen teratas, memperbarui stack.
Display mencetak elemen sisa setelah dua kali pop: 10.

```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";
    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();
   s.pop();

   cout << "after popping, stack elements: ";
   s.display();
   return 0;
}
```

```
stack elements: 302010
top element: 30
after popping, stack elements: 10
```

Kelas Node:
Node adalah elemen dalam stack, dengan dua atribut:
data: Menyimpan nilai elemen.
next: Pointer ke node berikutnya.
Konstruktor Node(int value) menginisialisasi data dengan nilai dan next dengan nullptr.

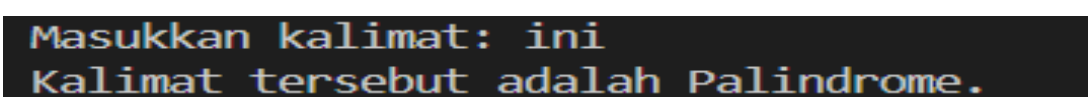
Kelas Stack:
Atribut:
top: Pointer ke elemen teratas stack.
Metode:
isEmpty(): Memeriksa apakah stack kosong.
push(int x): Menambahkan elemen baru ke stack dengan menghubungkannya ke top.
pop(): Menghapus elemen teratas dan memperbarui top.
peek(): Mengembalikan nilai elemen teratas tanpa menghapusnya.
display(): Menampilkan semua elemen stack dari atas ke bawah.

Fungsi main():
Membuat objek stack (Stack s).
Menambahkan tiga elemen ke stack (push(10), push(20), push(30)).
Menampilkan elemen stack dengan display().
Menampilkan elemen teratas dengan peek().
Menghapus dua elemen teratas menggunakan pop().
Menampilkan elemen stack setelah penghapusan.

Alur Program:
Push menambahkan elemen ke stack.
Display mencetak elemen stack: 30 20 10.
Peek menampilkan elemen teratas: 30.
Pop menghapus elemen teratas dan memperbarui stack.
Display mencetak elemen stack setelah dua kali pop: 10.

IV. UNGUIDED

```cpp
#include <iostream>
#include <stack>
#include <string>
#include <cctype>

bool isPalindrome(const std::string& str) {
    std::stack<char> stack;
    std::string cleanedStr;

    for (char ch : str) {
        if (std::isalnum(ch)) {
            cleanedStr += tolower(ch);
            stack.push(tolower(ch));
        }
    }

    for (char ch : cleanedStr) {
        if (ch != stack.top()) {
            return false;
        }
        stack.pop();
    }

    return true;
}

int main() {
    std::string input;
    std::cout << "Masukkan kalimat: ";
    std::getline(std::cin, input);

    if (isPalindrome(input)) {
        std::cout << "Kalimat tersebut adalah Palindrome.\n";
    } else {
        std::cout << "Kalimat tersebut bukan Palindrome.\n";
    }

    return 0;
}
```



Fungsi isPalindrome:

Menghapus karakter non-alfanumerik dan mengubah karakter menjadi huruf kecil.

Menggunakan stack untuk menyimpan karakter-karakter yang valid (huruf dan angka) dari kalimat tersebut.

Setelah itu, memeriksa apakah karakter-karakter dalam string yang sudah dibersihkan (cleanedStr) sama dengan urutan karakter yang ada di stack (dari belakang ke depan).

Jika ada ketidaksamaan, maka kalimat bukan palindrome. Jika semua karakter cocok, kalimat adalah palindrome.

Fungsi main:

Menerima input kalimat dari pengguna.

Memanggil fungsi isPalindrome untuk memeriksa apakah kalimat tersebut palindrome.

Menampilkan hasil apakah kalimat tersebut palindrome atau tidak.

Alur Program:

Input: Pengguna memasukkan kalimat.

Pembersihan: Hanya huruf dan angka yang diproses, dan semuanya diubah menjadi huruf kecil.

Pengecekan Palindrome: String dibaca dari depan dan belakang (menggunakan stack) untuk membandingkan karakter satu per satu.

Output: Menampilkan apakah kalimat tersebut palindrome atau tidak.

```cpp
#include <iostream>
#include <stack>
#include <sstream>
#include <string>

using namespace std;

// Fungsi untuk membalikkan sebuah string
string reverseString(const string& str) {
    string reversedStr;
    for (int i = str.length() - 1; i >= 0; i--) {
        reversedStr += str[i];
    }
    return reversedStr;
}

// Fungsi untuk membalikkan huruf dalam setiap kata dalam sebuah kalimat
void reverseWordsInSentence(const string& sentence) {
    stack<string> wordsStack;
    stringstream ss(sentence);
    string word;

    // Memisahkan kata-kata dan memasukkannya ke dalam stack
    while (ss >> word) {
        wordsStack.push(word);
    }

    // Mengeluarkan kata-kata dari stack dan membalikkan hurufnya
    while (!wordsStack.empty()) {
        string currentWord = wordsStack.top();
        wordsStack.pop();
        cout << reverseString(currentWord) << " ";
    }
    cout << endl;
}

int main() {
    string sentence;
    cout << "Masukkan kalimat: ";
    getline(cin, sentence);

    cout << "Datastack Array: " << endl;
    cout << "Data : ";
    reverseWordsInSentence(sentence);

    return 0;
}
```

```
Masukkan kalimat: Telkom Purwokerto
Datastack Array:
Data : otrekowruP mokleT
```

Fungsi reverseString:

Menerima sebuah string dan membalikkan urutan karakter-karakternya.

Fungsi ini dipakai untuk membalikkan setiap kata dalam kalimat.

Fungsi reverseWordsInSentence:

Membaca kalimat dan memisahkan kata-kata menggunakan stringstream.

Setiap kata dimasukkan ke dalam stack.

Setelah itu, kata-kata diambil dari stack satu per satu dan dibalikkan menggunakan reverseString, kemudian ditampilkan.

Fungsi main:

Mengambil input kalimat dari pengguna.

Memanggil reverseWordsInSentence untuk membalikkan huruf dalam setiap kata pada kalimat.

Menampilkan hasil output.

Alur Program:

Input: Pengguna memasukkan kalimat.

Pemrosesan: Kalimat dipisah menjadi kata-kata dan setiap kata dibalikkan menggunakan stack.

Output: Menampilkan kalimat dengan huruf dalam setiap kata yang sudah dibalik.

V. KESIMPULAN

Stack adalah struktur data dengan prinsip **Last-In, First-Out (LIFO)**, di mana elemen terakhir yang dimasukkan akan diambil pertama. Operasi dasar termasuk **push** (menambah elemen) dan **pop** (menghapus elemen), serta operasi tambahan seperti **top**, **isEmpty**, **size**, dan **peek**,Stack berguna dalam **manajemen memori**, **evaluasi ekspresi aritmatika**, **fungsi rekursif**, **undo/redo**, dan **traversal struktur data** (misalnya, DFS). Stack dapat diimplementasikan menggunakan **array**, **linked list**, atau **double linked list**, tergantung pada kebutuhan ukuran dan operasi,Secara keseluruhan, stack adalah struktur data yang penting untuk pengelolaan elemen secara terstruktur dalam berbagai aplikasi.