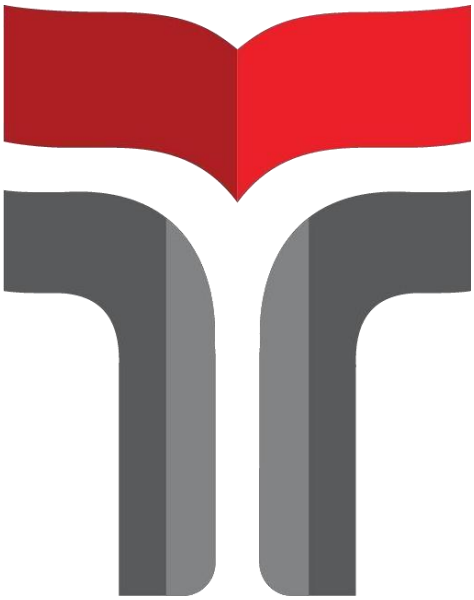
LAPORAN PRAKTIKUM
STRUKTUR DATA 7
"STACK"



Oleh:

NAMA: Ammar Dzaki Nandana

NIM: 2311104071

KELAS: SE 07 B

DOSEN: Wahyu Andi Saputra

PRODI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2023/2024

I. TUJUAN

Tujuan dari praktikum ini adalah untuk:

1. Memahami konsep dasar struktur data stack dan prinsip kerjanya yang mengikuti Last In First Out (LIFO).
2. Mempelajari cara mengimplementasikan struktur data stack menggunakan bahasa pemrograman C++.
3. Mengidentifikasi dan mempraktikkan operasi dasar pada stack, yaitu push, pop, peek, dan isEmpty.
4. Mengetahui cara menangani kondisi error seperti stack overflow dan stack underflow dalam implementasi stack.
5. Menggunakan struktur data stack untuk menyelesaikan masalah nyata, seperti manajemen data dengan prinsip LIFO.
6. Mempelajari alokasi dinamis untuk membuat stack dengan ukuran yang fleksibel.

II. DASAR TEORI

Pengertian Stack:

Stack merupakan salah satu bentuk struktur data dimana prinsip operasi yang digunakan seperti tumpukan. Seperti halnya tumpukan, elemen yang bisa diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang pertama kali masuk, prinsip ini biasa disebut LIFO (*Last In First Out*).

Operasi-Operasi pada Stack

1. Push

Push adalah operasi untuk menambahkan elemen ke dalam stack. Elemen baru akan dimasukkan di atas elemen yang sebelumnya berada di stack.

Contoh: Jika stack awalnya kosong, dan kita melakukan push(5), maka angka 5 akan menjadi elemen pertama dalam stack.

2. Pop

Pop adalah operasi untuk menghapus elemen teratas dari stack. Setelah elemen teratas dihapus, elemen berikutnya menjadi elemen teratas.

Contoh: Jika stack berisi angka 5 di atas dan angka 10 di bawahnya, dan kita melakukan pop(), maka angka 5 akan dihapus dan angka 10 menjadi elemen teratas yang baru.

3. Peek/Top

Peek atau Top adalah operasi untuk melihat elemen teratas dari stack tanpa menghapusnya. Fungsi ini memungkinkan kita untuk mengetahui nilai elemen teratas saat ini tanpa memodifikasi stack.

Contoh: Jika stack berisi angka 10 di atas dan 20 di bawahnya, melakukan peek() akan memberikan nilai 10 tanpa menghapusnya

4. isEmpty

isEmpty adalah operasi untuk memeriksa apakah stack kosong. Jika stack kosong, operasi lainnya seperti pop() atau peek() tidak dapat dilakukan karena tidak ada elemen yang dapat dihapus atau dilihat.

Contoh: Jika stack tidak memiliki elemen, maka isEmpty() akan mengembalikan nilai true.

Karakteristik Stack

1. LIFO (Last In, First Out): Hanya elemen yang paling baru ditambahkan yang dapat diakses terlebih dahulu.
2. Akses Terbatas: Hanya elemen teratas stack yang bisa diakses atau dimodifikasi pada satu waktu.
3. Dinamika Data: Stack bisa berubah ukuran selama program berjalan, tergantung pada elemen yang dimasukkan atau dikeluarkan.

Implementasi Stack

Stack dapat diimplementasikan dengan berbagai cara, antara lain:

1. Menggunakan Array: Stack diimplementasikan dengan menggunakan array statis. Kelemahannya adalah kapasitas yang terbatas.
2. Menggunakan Linked List: Stack dapat diimplementasikan dengan menggunakan struktur data linked list, yang memungkinkan alokasi dinamis dan tidak terbatas oleh kapasitas array.

Aplikasi Stack

Beberapa aplikasi penting dari stack meliputi:

1. Pemanggilan Fungsi (Call Stack): Saat fungsi dipanggil dalam program, informasi seperti alamat kembali dan variabel lokal disimpan dalam stack. Stack ini digunakan untuk menyimpan jejak pemanggilan fungsi.
2. Undo/Redo dalam Aplikasi: Aplikasi seperti pengolah kata menggunakan stack untuk menyimpan perubahan, yang memungkinkan pengguna untuk membatalkan (undo) atau mengulangi (redo) aksi terakhir.
3. Evaluasi Ekspresi Matematika: Stack digunakan dalam parsing ekspresi matematika untuk menghitung nilai ekspresi dalam bentuk infiks, prefiks, atau postfiks.
4. Navigasi Web (Back/Forward): Stack digunakan dalam penyimpanan halaman web yang dikunjungi, memungkinkan pengguna untuk kembali ke halaman sebelumnya (back) atau maju ke halaman berikutnya (forward).

III. 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 << "Stacl 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;
}

```

```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)
        {
            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;
}
```

IV. UNGUIDED

C: > C++ PEMROGRAMAN > MODUL 7 > unguided > Stack h.cpp > S

```cpp
Stack S;
createStack(S);
push(S, 3);
push(S, 4);
push(S, 8);
pop(S);
push(S, 2);
push(S, 3);
pop(S);
push(S, 9);
printInfo(S);
cout << "Balik stack:" << endl;
balikStack(S);
printInfo(S);

```

```cpp
Stack S;
createStack(S);
pushAscending(S, 3);
pushAscending(S, 4);
pushAscending(S, 8);
pushAscending(S, 2);
pushAscending(S, 3);
pushAscending(S, 9);
printInfo(S);
cout << "Balik stack:" << endl;
balikStack(S);
printInfo(S);

```

```cpp
#include <iostream>
#include "stack.h"
using namespace std;

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

// Menambahkan elemen ke stack
void push(Stack &S, int x) {
    if (S.top < MAX - 1) {
        S.info[++S.top] = x;
    } else {
        cout << "Stack overflow\n";
    }
}

// Menghapus elemen dari stack
int pop(Stack &S) {
    if (S.top >= 0) {
        return S.info[S.top--];
    }
    cout << "Stack underflow\n";
    return -1;
}

// Mencetak elemen stack
void printInfo(const Stack &S) {
    for (int i = 0; i <= S.top; i++) {
        cout << S.info[i] << " ";
    }
    cout << endl;
}

// Membalikkan elemen stack
void balikStack(Stack &S) {
    for (int i = 0, j = S.top; i < j; i++, j--) {
        swap(S.info[i], S.info[j]);
    }
}

// Menambahkan elemen secara ascending
void pushAscending(Stack &S, int x) {
    if (S.top < MAX - 1) {
        int i = S.top;
        while (i >= 0 && S.info[i] > x) {
            S.info[i + 1] = S.info[i];
            i--;
        }
        S.info[i + 1] = x;
        S.top++;
    } else {
        cout << "Stack overflow\n";
    }
}

// Membaca input dari user
void getInputStream(Stack &S) {
    char c;
    cout << "Masukkan elemen (akhiri dengan enter): ";
    while (cin.get(c) && c != '\n') {
        if (S.top < MAX - 1) {
            push(S, c - '0'); // Konversi char ke int
        } else {
            cout << "Stack overflow\n";
            break;
        }
    }
}

```

V. KESIMPULAN

Stack adalah struktur data yang sangat penting dalam pengelolaan data dalam urutan LIFO. Dengan operasi dasar seperti **push**, **pop**, **peek**, dan **isEmpty**, stack menyediakan cara yang efisien untuk mengelola elemen-elemen dalam urutan tertentu. Penggunaan stack sangat luas dalam berbagai aplikasi komputasi, seperti manajemen memori, evaluasi ekspresi, dan sistem undo/redo.