LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
NAMA : KHOIRUL ADDIFA
NIM : 103112400172

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Stack adalah salah satu struktur data yang cara kerjanya menyerupai tumpukan benda, di mana elemen yang berada di posisi paling atas adalah elemen yang dapat diakses atau diambil terlebih dahulu. Struktur data ini menggunakan prinsip LIFO (Last In First Out), yaitu elemen yang terakhir dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Dengan mekanisme tersebut, stack sering digunakan dalam berbagai proses komputasi seperti pengelolaan memori, rekursi, dan pengolahan data yang membutuhkan urutan pemrosesan terbalik.

Guided 1

stack.h

```h
#ifndef STACK_H
#define STACK_H

const int MAX = 20;

typedef int infotype;

struct Stack {
    infotype info[MAX];
    int top;
};

void createStack(Stack &S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);
void pushAscending(Stack &S, infotype x);
void getInputStream(Stack &S);

#endif
```

stack.cpp

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

void createStack(Stack &S) {
    S.top = -1;
}

void push(Stack &S, infotype x) {
    if (S.top < MAX - 1) {
        S.top++;
```

```
        S.info[S.top] = x;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        int temp = S.info[S.top];
        S.top--;
        return temp;
    }
    return -1; // jika kosong
}

void printInfo(Stack S) {
    cout << "[TOP] ";
    for (int i = S.top; i >= 0; i--) {
        cout << S.info[i] << " ";
    }
    cout << endl;
}

void balikStack(Stack &S) {
    Stack temp;
    createStack(temp);

    while (S.top >= 0) {
        push(temp, pop(S));
    }

    S = temp;
}

void pushAscending(Stack &S, infotype x) {
    // Gunakan stack bantu agar urutan jadi ascending dari TOP
    Stack temp;
    createStack(temp);

    // pindahkan elemen yang lebih kecil
    while (S.top >= 0 && S.info[S.top] > x) {
        push(temp, pop(S));
    }

    // push elemen baru
    push(S, x);

    // kembalikan elemen lain
    while (temp.top >= 0) {
        push(S, pop(temp));
```

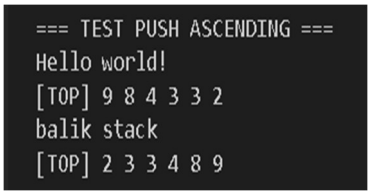
```
    }
}

void getInputStream(Stack &S) {
    char c = cin.get();
    while (c != '\n') {
        if (c >= '0' && c <= '9') {
            push(S, c - '0'); // ubah char ke integer
        }
        c = cin.get();
    }
}
```

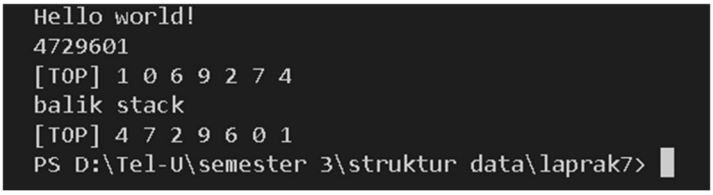
main.cpp

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

int main() {
    cout << "Hello world!" << endl;

    // ================================
    // TEST 1 : push biasa
    // ================================
    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);

    // ================================
    // TEST 2 : push ascending
    // ================================
    cout << "\n=== TEST PUSH ASCENDING ===" << endl;
    cout << "Hello world!" << endl;
    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);

    // ================================
    // TEST 3 : input stream
    // ================================
    cout << "\n";
    cout << "Hello world!" << endl;
    createStack(S);
    getInputStream(S);

    printInfo(S);
    cout << "balik stack" << endl;
    balikStack(S);
    printInfo(S);

    return 0;
}
```

Screenshots Output





Deskripsi:

Program ini bertujuan mengimplementasikan struktur data Stack menggunakan array untuk menyimpan data berupa bilangan integer. Operasi yang disediakan meliputi push, pop, menampilkan isi stack, membalik isi stack, memasukkan elemen secara ascending, serta membaca input user secara langsung dengan input stream. Seluruh operasi mengikuti prinsip LIFO (Last In First Out), di mana elemen terakhir yang masuk akan menjadi elemen pertama yang keluar. Program ini melatih mahasiswa dalam memahami konsep dasar struktur data stack, pengelolaan array, serta pembuatan prosedur dan fungsi untuk memanipulasi data secara terstruktur.

Kesimpulan

Dalam praktikum Modul 4 ini, isimpulkan bahwa konsep Stack berhasil diterapkan menggunakan struktur data array sebagai media penyimpanan. Mahasiswa mempelajari cara membuat stack, menambah elemen (push), menghapus elemen (pop), menampilkan isi stack, membalik urutan data, serta melakukan penyisipan data secara ascending dan membaca input data melalui input stream. Berbeda dengan struktur data lain yang lebih fleksibel, stack memiliki aturan khusus yaitu LIFO (Last In First Out) sehingga elemen yang terakhir masuk akan menjadi elemen pertama yang keluar. Secara keseluruhan, praktikum ini melatih pemahaman mahasiswa tentang implementasi Abstract Data Type (ADT) Stack, pengelolaan data terstruktur menggunakan array, serta penerapan prosedur dan fungsi untuk membangun program yang efisien dan terorganisir.

B. Referensi

- Drozdek, A. (2013). Data Structures and Algorithms in C++ (4th ed.). Cengage Learning.
- Lafore, R. (2017). Data Structures & Algorithms in C++ (2nd ed.). Sams Publishing.
- Weiss, M. A. (2014). Data Structures and Algorithm Analysis in C++ (4th ed.). Pearson.