LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
NAMA : RISKY CAHAYU
NIM : 103112430121

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Menurut Wikipedia (2024), stack atau tumpukan adalah salah satu bentuk abstract data type (ADT) yang digunakan untuk menyimpan kumpulan elemen dengan aturan akses khusus, yaitu LIFO (Last In, First Out) — elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Struktur ini memiliki dua operasi utama, yaitu push, untuk menambahkan elemen ke bagian atas tumpukan, dan pop, untuk menghapus elemen paling atas dari tumpukan. Selain itu, beberapa implementasi juga menyediakan operasi tambahan seperti peek atau top, yang digunakan untuk melihat elemen teratas tanpa menghapusnya. Stack banyak digunakan dalam berbagai proses komputasi, seperti penyimpanan data sementara, pengelolaan pemanggilan fungsi melalui call stack, pembalikan urutan data, serta penerapan algoritma rekursif. Dalam implementasinya, stack dapat dibuat menggunakan array maupun linked list, tergantung kebutuhan dan efisiensi memori yang diinginkan. Konsep stack ini menjadi dasar penting dalam ilmu komputer karena banyak algoritma dan struktur data lain yang bekerja dengan prinsip serupa.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

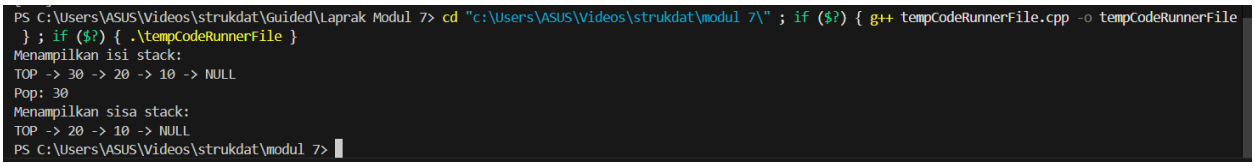
Kode stack.cpp

```cpp
// Risky Cahayu
// 103112430121

#include <iostream>
using namespace std;

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

bool isEmpty(Node *top)
{
    return top == nullptr;
}

void push(Node *&top, int data)
{
    Node *newNode = new Node();
    newNode-> data = data;
    newNode-> next = top;
    top = newNode;
}

int pop(Node *&top)
```

```
{
    if (isEmpty(top))
    {
        cout << "Stack kosong, tidak bisa pop!" << endl;
        return 0;
    }

    int poppedData = top->data;
    Node *temp = top;
    top = top->next;

    delete temp;
    return poppedData;
}

void show(Node *top)
{
    if (isEmpty(top))
    {
        cout << "Stack kosong." << endl;
        return;
    }

    cout << "TOP -> ";
    Node *temp = top;

    while (temp != nullptr)
    {
        cout << temp->data << " -> ";
        temp = temp->next;
    }

    cout << "NULL" << endl;
}

int main()
{
    Node *stack = nullptr;

    push(stack, 10);
    push(stack, 20);
    push(stack, 30);

    cout << "Menampilkan isi stack:" << endl;
    show(stack);

    cout << "Pop: " << pop(stack) << endl;
```

```
    cout << "Menampilkan sisa stack:" << endl;
    show(stack);

    return 0;
}
```

Screenshots Output



Deskripsi:

Kode program di atas merupakan implementasi struktur data stack (tumpukan) menggunakan linked list dinamis dalam bahasa C++. Stack adalah struktur data yang menerapkan prinsip LIFO (Last In, First Out), artinya elemen yang terakhir dimasukkan akan menjadi yang pertama dikeluarkan. Program ini mendefinisikan struktur Node yang memiliki dua komponen: data untuk menyimpan nilai dan next sebagai pointer yang menunjuk ke node berikutnya. Fungsi isEmpty() digunakan untuk memeriksa apakah stack kosong, sedangkan push() berfungsi menambahkan elemen baru di atas stack dengan membuat node baru yang menunjuk ke elemen sebelumnya. Fungsi pop() menghapus elemen teratas dari stack dan mengembalikan nilainya, sekaligus menghapus memori node tersebut untuk menghindari kebocoran memori. Fungsi show() menampilkan seluruh isi stack dari elemen teratas hingga terbawah dengan format yang jelas. Pada fungsi main(), program membuat sebuah stack kosong, kemudian menambahkan tiga elemen (10, 20, dan 30), menampilkan isi stack, melakukan operasi pop() satu kali untuk menghapus elemen teratas, lalu menampilkan kembali sisa isi stack, sehingga pengguna dapat memahami bagaimana elemen bertambah dan berkurang sesuai prinsip kerja stack.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

Kode stack.h

```
// Risky Cahayu
// 103112430121

#ifndef STACK_H
#define STACK_H

const int MAX = 20;

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

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

#endif
```

Kode stack.cpp

```
// Risky Cahayu
// 103112430121

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

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

void push(Stack &S, int x) {
    if (S.top < MAX - 1) {
        S.top++;
        S.info[S.top] = x;
    }
}
```

```cpp
int pop(Stack &S) {
    if (S.top >= 0) {
        int temp = S.info[S.top];
        S.top--;
        return temp;
    }
    return -1;
}

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 != -1) {
        push(temp, pop(S));
    }
    S = temp;
}

void pushAscending(Stack &S, int x) {
    // Menyisipkan elemen secara urut naik
    Stack temp;
    createStack(temp);

    while (S.top != -1 && S.info[S.top] > x) {
        push(temp, pop(S));
    }
    push(S, x);
    while (temp.top != -1) {
        push(S, pop(temp));
    }
}

void getInputStream(Stack &S) {
    char ch;
    while (cin.get(ch)) {
        if (ch == '\n') break;
        push(S, ch - '0');
    }
}
```

Kode main.cpp

```cpp
// Risky Cahayu
// 103112430121

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

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

    Stack S;
    createStack(S);

    // ========== LATIHAN 1 ==========
    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);

    // ========== LATIHAN 2 ==========
    cout << "\nPush 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);

    // ========== LATIHAN 3 ==========
    cout << "\nInput stream:" << endl;
```

```
    cout << "Hello world!" << endl;

    createStack(S);
    getInputStream(S);
    printInfo(S);

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

    return 0;
}
```

Screenshots Output

```
PS C:\Users\ASUS\Videos\strukdat\Guided\Laprak Modul 7> g++ main.cpp stack.cpp -o main
>>
PS C:\Users\ASUS\Videos\strukdat\Guided\Laprak Modul 7> ./main
Hello world!
[TOP] 9 2 4 3
balik stack:
[TOP] 3 4 2 9

Push ascending:
Hello world!
[TOP] 9 8 4 3 3 2
balik stack:
[TOP] 2 3 3 4 8 9

Input stream:
Hello world!
4729601
[TOP] 1 0 6 9 2 7 4
balik stack:
[TOP] 4 7 2 9 6 0 1
```

Deskripsi:

Kode program di atas bertujuan untuk mengimplementasikan struktur data Stack (tumpukan) secara manual menggunakan bahasa C++ dengan konsep Abstract Data Type (ADT), yang dipisahkan dalam tiga bagian file: stack.h (deklarasi), stack.cpp (implementasi fungsi), dan main.cpp (program utama untuk pengujian). Stack sendiri menggunakan prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang keluar. Program ini menyediakan beberapa operasi dasar, yaitu createStack() untuk inisialisasi stack kosong, push() untuk menambahkan data ke dalam stack, pop() untuk menghapus data paling atas, printInfo() untuk menampilkan isi stack dari atas ke bawah, serta balikStack() untuk membalik urutan isi stack. Selain itu, terdapat dua fitur tambahan, yaitu pushAscending() yang berfungsi menyisipkan elemen baru secara otomatis dalam urutan menaik (ascending

order) menggunakan stack sementara, dan getInputStream() yang membaca input karakter angka dari pengguna dan menyimpannya ke dalam stack. Program utama (main()) kemudian melakukan tiga percobaan (latihan) untuk menguji semua fungsi tersebut: operasi dasar stack, penyisipan ascending, dan input data dari pengguna, sekaligus menampilkan hasil pembalikan stack dari setiap percobaan. Dengan demikian, program ini tidak hanya menunjukkan penggunaan stack dasar, tetapi juga penerapan manipulasi dan pengelolaan data secara efisien menggunakan prinsip LIFO.

D. Kesimpulan

Kesimpulannya, dari hasil program yang sudah dibuat, kita bisa memahami bagaimana cara kerja struktur data stack dengan lebih jelas. Program ini menunjukkan bahwa stack bekerja dengan prinsip LIFO (Last In First Out), yaitu data yang dimasukkan terakhir akan keluar pertama. Melalui beberapa latihan, kita bisa melihat berbagai operasi seperti menambah data (push), menghapus data (pop), menampilkan isi stack (printInfo), membalik urutan isi stack (balikStack), memasukkan data secara urut naik (pushAscending), dan juga membaca input langsung dari pengguna (getInputStream). Semua fungsi ini membantu kita memahami bagaimana stack dapat digunakan untuk mengelola data secara teratur dan efisien. Jadi, secara sederhana, program ini memperlihatkan penerapan konsep stack dalam pemrograman C++ dan bagaimana kita bisa memodifikasi fungsinya agar lebih fleksibel sesuai kebutuhan.

E. Referensi

Stack (abstract data type) — penjelasan mengenai stack sebagai ADT dengan operasi utama push dan pop, serta prinsip LIFO https://en.wikipedia.org/wiki/Stack_%28abstract_data_type%29

Abstract data type — pembahasan tentang tipe data abstrak, salah satunya stack, termasuk definisi operasi-dasarnya. *https://en.wikipedia.org/wiki/Abstract_data_type*

Call stack — struktur data stack khusus dalam pemrograman untuk menyimpan informasi subrutin/aktivasi fungsi. https://en.wikipedia.org/wiki/Call_stack