LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
PENGENALAN CODE BLOCKS

Disusun Oleh :

NAMA : Rikza Nur Zaki

NIM : 103112430030

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori Stack

Stack (tumpukan) adalah salah satu bentuk struktur data linear. Prinsip operasinya menyerupai tumpukan fisik, di mana elemen yang dapat diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang terakhir kali masuk. Komponen utama dalam stack yang berfungsi untuk mengakses data hanyalah elemen paling awal saja, yang disebut TOP. Akses pada stack hanya bisa dilakukan pada awal stack saja. Dalam stack, terdapat dua operasi utama yang berfokus pada elemen TOP : Push (penyisipan) dan Pop (pengambilan).

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

Guided 1

```cpp
#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 di pop woi!!!!" << 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, tidak bisa di tampilkan woi!!!!" << 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

```
Menampilkan isi stack:
TOP -> 30 -> 20 -> 10 -> NULL
Pop: 30
Menampilkan sisa stack:
TOP -> 20 -> 10 -> NULL
```

Deskripsi:

Codingan diatas merupakan implementasi dari struktur data Stack menggunakan Linked List yang didefinisikan oleh struct Node dengan data dan pointer next, di mana fungsi push selalu menyisipkan elemen baru ke bagian top (insert first), fungsi pop menghapus elemen teratas dan melakukan dealokasi memori (delete temp) setelah mengecek kondisi kosong dengan isEmpty, dan fungsi show menampilkan urutan Stack dari TOP hingga NULL.

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

Unguided 1

Stack.h

```h
#ifndef STACK_H
#define STACK_H
#define MAX_SIZE 20

typedef int infotype;

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

void createStack(Stack &S);
bool isEmpty(Stack S);
bool isFull(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 "stack.h"
#include <iostream>

using namespace std;

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

bool isEmpty(Stack S) {
    return S.top == -1;
}

bool isFull(Stack S) {
    return S.top == MAX_SIZE - 1;
}

void push(Stack &S, infotype X) {
    if (isFull(S)) {
        cout << "Stack penuh." << endl;
    } else {
        S.top++;
        S.info[S.top] = X;
    }
}

infotype pop(Stack &S) {
    if (isEmpty(S)) {
        cout << "Stack kosong." << endl;

        return 0;
    } else {
        infotype popped_element = S.info[S.top];
        S.top--;
        return popped_element;
    }
}
```

```cpp
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 (!isEmpty(S)) {
        push(temp, pop(S));
    }

    while (!isEmpty(temp)) {
        push(S, pop(temp));
    }
}

void pushAscending(Stack &S, infotype X) {
    Stack temp;
    createStack(temp);

    while (!isEmpty(S) && S.info[S.top] > X) {
        push(temp, pop(S));
    }

    push(S, X);

    while (!isEmpty(temp)) {
        push(S, pop(temp));
    }
}

void getInputStream(Stack &S) {
    infotype inputChar;
    cout << "Masukkan input: ";

    char c;
    while (cin.get(c) && c != '\n') {
        if (isdigit(c)) {
            inputChar = c - '0';
            push(S, inputChar);
        }
    }
}
```

Main.cpp

```cpp
#include "stack.h"
#include "stack.cpp"
#include <iostream>

using namespace std;

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

    cout << "Hello world!" << endl;
    pushAscending(S, 3);
    pushAscending(S, 9);
    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);
    createStack(S);

    cout << "Hello world!" << endl;
    getInputStream(S);
    printInfo(S);

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

    return 0;
}
```

Screenshots Output

```
Hello world!
[TOP] 9 2 4 3
balik stack
[TOP] 9 2 4 3
```

```
Hello world!
[TOP] 9 9 8 4 3 3 2
balik stack
[TOP] 9 9 8 4 3 3 2
```

```
Hello world!
Masukkan input: 87291982
[TOP] 2 8 9 1 9 2 7 8
balik stack
[TOP] 2 8 9 1 9 2 7 8
```

Deskripsi:

Codingan diatas merupakan implementasi Abstract Data Type Stack menggunakan Array, yang terbagi menjadi header (prototipe fungsi seperti push, pop, balikStack, pushAscending, getInputStream), implementasi, dan main yang mendemonstrasikan tiga skenario berbeda: operasi dasar, pushAscending untuk menjaga urutan, dan getInputStream untuk memasukkan input dari user.

F. Kesimpulan

Operasi inti push dan pop selalu bekerja pada indeks top. Operasi push meningkatkan top sebelum penyisipan, dan pop menurunkan top setelah pengambilan elemen, memastikan prinsip LIFO(Last In, First Out). Selain primitif dasar seperti createStack, isEmpty, isFull, fungsi lanjutan seperti balikStack (membalik urutan Stack menggunakan Stack bantuan) , pushAscending (memasukkan elemen sambil menjaga urutan Stack tetap naik) , dan getInputStream (mengambil digit input karakter dari user dan memasukkannya ke Stack) berhasil diimplementasikan.

G. Referensi

Data Structures and Algorithms in C++ oleh Adam Drozdek (Thomson, 2012) — Bab 3 (Stack dan Queue).

https://www.geeksforgeeks.org/cpp/stack-in-cpp-stl/