LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL STACK

Disusun Oleh :

NAMA : Ayu Setyaning Tyas

NIM : 103112430119

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Stack adalah struktur data linear yang mengikuti urutan operasi tertentu. Urutannya bisa LIFO (Last In First Out) atau FILO (First In Last Out) . LIFO berarti elemen yang dimasukkan terakhir akan keluar pertama, dan FILO berarti elemen yang dimasukkan pertama akan keluar terakhir. Banyak tugas pemrograman melibatkan penggunaan stack – misalnya manajemen memori atau proses fungsi rekursif – sehingga memahami *apa itu stack* mempermudah pengorganisasian data dalam kode.

Banyak tugas pemrograman melibatkan struktur data stack. Contohnya, ketika seorang programmer menulis kode seperti di atas, ia sering menggunakan stack untuk mengelola data yang bersifat LIFO. Struktur data stack mempermudah pengorganisasian data dalam berbagai situasi.

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

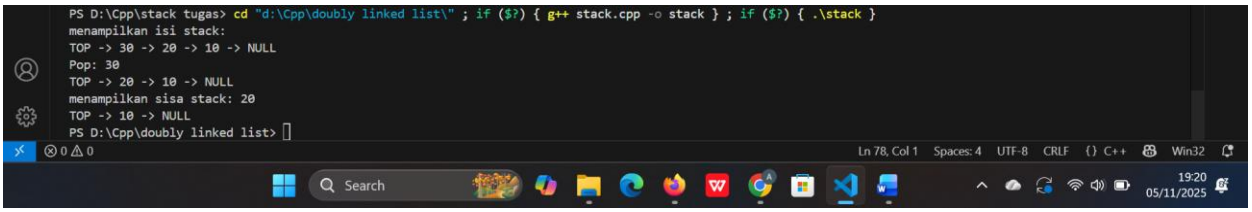
Guided 1

Code Program

```
#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;
    show(stack);

    cout << "menampilkan sisa stack: " << pop(stack) << endl;
    show(stack);

    return 0;
}
```

Screenshoot Output



Deskripsi:

Program ini mendemontrasikan operasi dasar stack yang terdiri dari push, pop, dan tampikan isi. Program ini akan menggunakan TOP sebagai penunjuk pointer yang di urutkan dari yang terbesar dan menggunakkan POP untuk menghapus pointer yang akan di hapus.

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

Unguided 1

Code *stack.h*

```
#ifndef STACK_H
#define STACK_H

#define MAX 100

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

void createStack(Stack &S);
bool isEmpty(Stack S);
bool isFull(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);

#endif
```

Code *Stack.cpp*

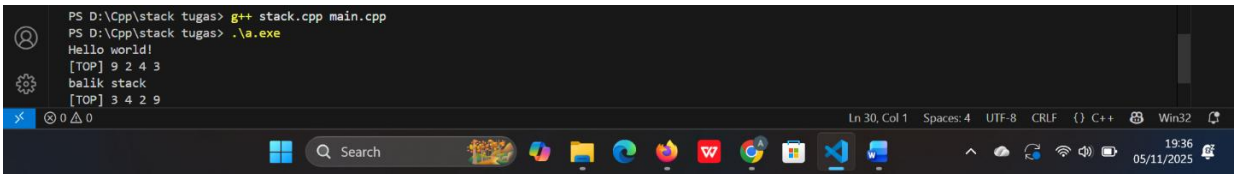
```
#include <iostream>
#include "stack.h"
using namespace std;

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

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

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

void push(Stack &S, int x) {
   if (!isFull(S)) {
      S.top++;
      S.data[S.top] = x;
   } else {
      cout << "Stack penuh!" << endl;
   }
}

int pop(Stack &S) {
   if (!isEmpty(S)) {
      int temp = S.data[S.top];
      S.top--;
      return temp;
   } else {
      cout << "Stack kosong!" << endl;
      return -1;
   }
}

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

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

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

Code *main.cpp*

```
#include <iostream>
#include "stack.h"
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);

    return 0;
}
```

Screenshots Output



Deskripsi:

Program ini menunjukkan bagaimana konsep ADT dan Stack diterapkan dalam C++ dengan pemisahan antara *interface* (deklarasi fungsi) dan *implementation* (definisi fungsi), sekaligus menggambarkan prinsip kerja stack menggunakan array secara terstruktur. Program ini akan

menggeluakan kalimat “hallo world!” disertai dengan TOP yang mengurutkan pointer dari nilai terbesar lalu di balik menjadi TOP yang mengurutkan pointer dari nilaii terkecil.

Unguided 2

Code *stack.h*

**Sama dengan code sebelumnya hanya saja ada sedikit penambahan*

```
void pushAscending(Stack &S, int x);
```

Code *stack.cpp*

**Sama dengan code sebelumnya hanya saja ada sedikit penambahan*

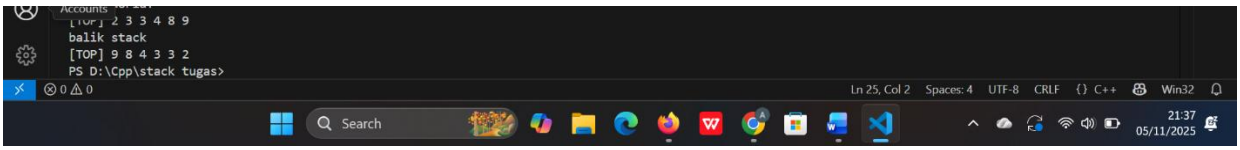
```
void pushAscending(Stack &S, int x) {
   Stack temp;
   createStack(temp);

   while (!isEmpty(S) && S.data[S.top] < x) {
      push(temp, pop(S));
   }

   push(S, x);

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

Code *main.cpp*

**Sama dengan code sebelumnya hanya saja ada sedikit perubahan pada penulisan code push*

```
pushAscending(S, 3);
pushAscending(S, 4);
pushAscending(S, 8);
pushAscending(S, 2);
pushAscending(S, 3);
pushAscending(S, 9);
```

Screenshoot Output



Deskripsi

Program kurang lebih sama dengan sebelumnya hanya saja ada bebrapa tambahan, tambahan terletak pada penambahan Prosedur pushAscending yang merupakan prosedur yang digunakan untuk mengurutkan nilai pointer dari yang terkecil keterbesar.

Unguided 3

Code *stack.h*

**Sama dengan code sebelumnya hanya saja ada sedikit penambahan*

```
void getInputStream(Stack &S);
```

Code *stack.cpp*

**Sama dengan code sebelumnya hanya saja ada sedikit penambahan*

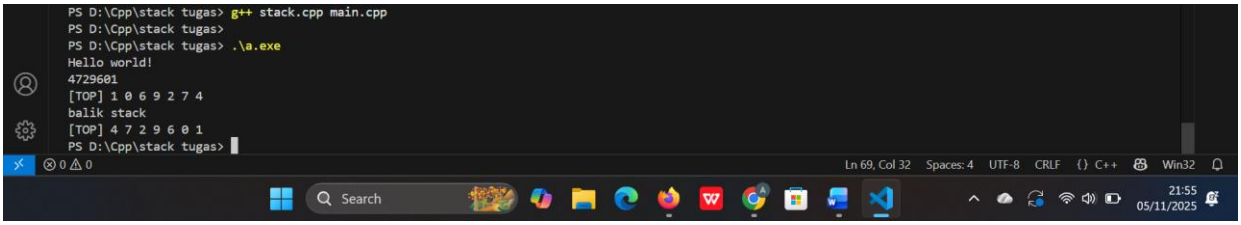
```
void getInputStream(Stack &S) {
    char c;

    c = cin.get();

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

Code *main.cpp*

**Sama dengan code sebelumnya hanya saja ada sedikit perubahan pada penulisan code push*

```
getInputStream(S);
    printInfo(S);
```

Screenshoot Output



Deskripsi

Program kurang lebih sama dengan sebelumnya hanya saja ada bebrapa tambahan, tambahan terletak pada penambahan Prosedur getInputStream yang merupakan prosedur untuk mengurutkan angka yang di input ke dalam pointer lalu di urutkan dengan prosedur pushAscending lalu di balikkan lagi sesuai dengan urutan yang di input tapi dalam bentuk pointer.

D. Kesimpulan

Program ADT Stack ini menggunakan array untuk menerapkan struktur data tumpukan yang mengikuti prinsip LIFO, yaitu yang terakhir masuk akan menjadi yang pertama keluar. Program ini diatur menjadi tiga file utama, yaitu stack. h yang berisi deklarasi tipe data dan fungsi, stack. cpp untuk menjalankan fungsi-fungsi tersebut, dan main. cpp yang berfungsi sebagai program utama untuk pengujian. Fungsi-fungsi dasar yang ada antara lain createStack, push, pop, printInfo, dan balikStack. Ada juga prosedur baru bernama pushAscending yang membuat elemen ditambahkan ke dalam stack secara otomatis dalam urutan naik tanpa perlu menata ulang, sehingga data tetap rapi. Hasil keluaran bisa berbeda tergantung pada cara membandingkan yang digunakan dalam fungsi tersebut, apakah menggunakan "<" atau ">". Jika prosedur getInputStream ditambahkan, program bisa menerima input langsung dari pengguna sehingga bisa lebih interaktif dan mudah dalam mengisi data. Secara keseluruhan, program ini menunjukkan bagaimana menerapkan konsep abstraksi data dan membuat fungsi stack menjadi lebih efektif dan mudah digunakan.

E. Referensi

https://www.geeksforgeeks.org/dsa/stack-data-structure/
https://www.codepolitan.com/blog/apa-itu-stack-dalam-ilmu-pemrograman/