LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
NAMA : AGUNG RAMADHAN
NIM : 103112430060

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Stack merupakan salah satu bentuk struktur data linier yang prinsip kerjanya mengikuti konsep LIFO (Last In First Out), artinya elemen yang terakhir dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Konsep ini dapat dianalogikan seperti tumpukan piring; piring yang paling atas adalah yang pertama diambil. Struktur data stack memiliki dua operasi utama, yaitu push dan pop. Operasi *push* digunakan untuk menambahkan elemen ke dalam stack, sedangkan operasi *pop* digunakan untuk menghapus elemen teratas dari stack. Selain itu, terdapat juga operasi tambahan seperti *peek* atau *top* untuk melihat elemen paling atas tanpa menghapusnya, serta *isEmpty* untuk memeriksa apakah stack kosong. Stack dapat diimplementasikan menggunakan array (representasi statis) maupun linked list (representasi dinamis). Dalam implementasi berbasis array, posisi elemen dikontrol oleh variabel penunjuk *top* yang menunjukkan indeks elemen paling atas. Struktur data stack banyak digunakan dalam berbagai aplikasi komputer, seperti pengelolaan memori rekursif, pembalikan urutan data (seperti pembalikan string), serta dalam kompilasi dan evaluasi ekspresi aritmatika.

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

Guided 1

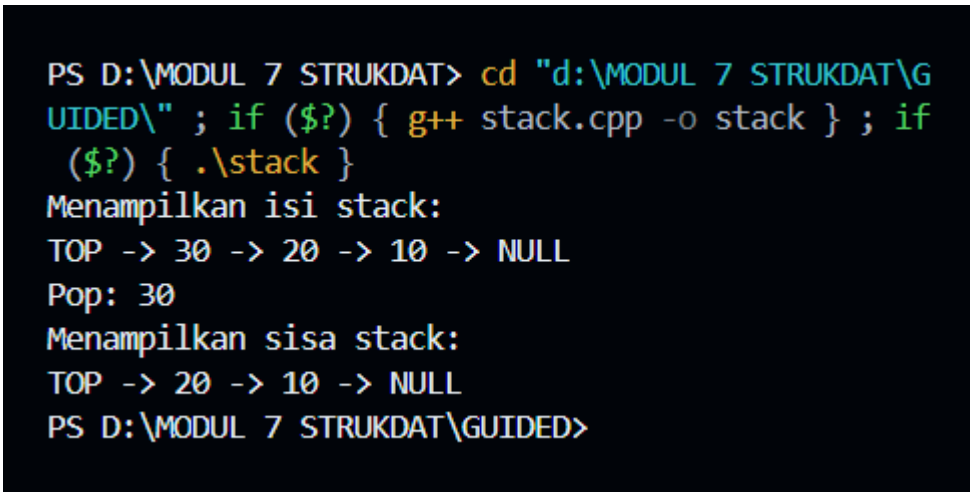
Kode stack.cpp

```cpp
// AGUNG RAMADHAN
// 103112430060

#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:

Stack merupakan salah satu struktur data dasar yang bekerja dengan prinsip LIFO (Last In, First Out), yaitu elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini dapat dianalogikan seperti tumpukan piring—piring yang terakhir ditaruh di atas akan diambil terlebih dahulu. Dalam implementasinya, stack memiliki dua operasi utama, yaitu push untuk menambahkan elemen ke dalam stack dan pop untuk menghapus elemen dari puncak stack. Selain itu, dapat pula ditambahkan operasi peek atau top untuk melihat elemen teratas tanpa menghapusnya. Stack dapat diimplementasikan menggunakan array maupun linked list, di mana linked list memberikan keunggulan berupa alokasi memori yang dinamis sehingga tidak memerlukan ukuran tetap sejak awal. Struktur data ini banyak digunakan dalam berbagai aplikasi komputer, seperti pengelolaan memori (fungsi rekursif), pembalikan urutan data, dan proses kompilasi program.

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

Unguided 1

Kode stack.h

```
// AGUNG RAMADHAN
// 103112430060

#ifndef STACK_H
#define STACK_H

const int MAX_SIZE = 20;

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

void initStack(Stack &S);
bool isFull(const Stack &S);
bool isEmpty(const Stack &S);
void push(Stack &S, int value);
int pop(Stack &S);
void showStack(Stack S);
void reverseStack(Stack &S);
void pushInOrder(Stack &S, int value);
void inputToStack(Stack &S);

#endif
```

Kode stack.cpp

```
// AGUNG RAMADHAN
// 103112430060

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

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

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

bool isEmpty(const Stack &S) {
```

```
    return S.top == -1;
}

void push(Stack &S, int value) {
    if (!isFull(S)) {
        S.data[++S.top] = value;
    }
}

int pop(Stack &S) {
    if (!isEmpty(S)) {
        return S.data[S.top--];
    }
    return -1;
}

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

void reverseStack(Stack &S) {
    Stack temp;
    initStack(temp);
    while (!isEmpty(S)) {
        push(temp, pop(S));
    }
    S = temp;
}

void pushInOrder(Stack &S, int value) {
    Stack temp;
    initStack(temp);

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

    push(S, value);

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

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

Kode main.cpp

```
// AGUNG RAMADHAN
// 103112430060

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

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

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

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

void push(Stack &S, int value) {
    if (!isFull(S)) {
        S.data[++S.top] = value;
    }
}

int pop(Stack &S) {
    if (!isEmpty(S)) {
        return S.data[S.top--];
    }
    return -1;
}

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

```
    cout << endl;
}

void reverseStack(Stack &S) {
    Stack temp;
    initStack(temp);
    while (!isEmpty(S)) {
        push(temp, pop(S));
    }
    S = temp;
}

void pushInOrder(Stack &S, int value) {
    Stack temp;
    initStack(temp);

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

    push(S, value);

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

void inputToStack(Stack &S) {
    char ch;
    while (cin.get(ch)) {
        if (ch == '\n') break;
        if (isdigit(ch)) {
            push(S, ch - '0');
        }
    }
}// AGUNG RAMADHAN
// 103112430060

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

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

    Stack A;
    initStack(A);
```

```
    push(A, 3);
    push(A, 4);
    push(A, 8);
    pop(A);
    push(A, 2);
    push(A, 3);
    pop(A);
    push(A, 9);
    showStack(A);

    cout << "Setelah dibalik:\n";
    reverseStack(A);
    showStack(A);

    cout << "Hello world!\n";

    Stack B;
    initStack(B);

    pushInOrder(B, 3);
    pushInOrder(B, 4);
    pushInOrder(B, 8);
    pushInOrder(B, 2);
    pushInOrder(B, 3);
    pushInOrder(B, 9);
    showStack(B);

    cout << "Setelah dibalik:\n";
    reverseStack(B);
    showStack(B);

   cout << "Hello world!\n";
    Stack C;
    initStack(C);

    inputToStack(C);
    showStack(C);

    cout << "Setelah dibalik:\n";
    reverseStack(C);
    showStack(C);

    return 0;
}
```

Screenshots Output

```
PS D:\MODUL 7 STRUKDAT\UNGUIDED> g++ main.cpp stack.cpp -o main
>> ./main
Hello world!
[TOP] 9 2 4 3
Setelah dibalik:
[TOP] 3 4 2 9
Hello world!
[TOP] 9 8 4 3 3 2
Setelah dibalik:
[TOP] 2 3 3 4 8 9
Hello world!
4729601
[TOP] 1 0 6 9 2 7 4
Setelah dibalik:
[TOP] 4 7 2 9 6 0 1
PS D:\MODUL 7 STRUKDAT\UNGUIDED>
```

Deskripsi:

Program di atas merupakan implementasi sederhana dari konsep struktur data stack (tumpukan) menggunakan bahasa C++. Program ini dibagi menjadi tiga bagian utama yaitu file header (stack.h), file implementasi (stack.cpp), dan file utama (main.cpp). Pada bagian stack.h, didefinisikan struktur data Stack beserta fungsi-fungsi dasarnya seperti initStack, push, pop, showStack, reverseStack, pushInOrder, dan inputToStack. Bagian stack.cpp berisi implementasi dari fungsi-fungsi tersebut, yang berfungsi untuk menambahkan elemen ke dalam stack, menghapus elemen dari stack, menampilkan isi stack, membalik urutan elemen stack, serta memasukkan elemen secara berurutan. Sementara itu, pada file main.cpp, program utama dijalankan dengan membuat tiga objek stack (A, B, dan C) untuk menguji berbagai operasi stack. Stack A digunakan untuk operasi dasar push dan pop, Stack B untuk pengujian fungsi pushInOrder (menyimpan data secara ascending), dan Stack C untuk menerima input langsung dari pengguna. Setelah setiap operasi, isi stack ditampilkan sebelum dan sesudah dibalik menggunakan reverseStack. Selain itu, terdapat pesan “Hello world!” yang dicetak di beberapa bagian program sebagai penanda jalannya program. Secara keseluruhan, program ini menunjukkan cara kerja stack dengan prinsip LIFO (Last In, First Out) dan penerapannya dalam pengolahan data sederhana.

D. Kesimpulan

Kesimpulan dari program ini adalah bahwa struktur data stack dapat digunakan untuk menyimpan dan mengelola data dengan prinsip LIFO (Last In, First Out), yaitu data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan. Melalui percobaan yang dilakukan dalam tiga latihan, dapat dipahami cara kerja operasi dasar stack seperti push, pop, dan print, serta penerapan fungsi tambahan seperti reverseStack

untuk membalik urutan elemen dan pushInOrder untuk menyisipkan data secara terurut. Program ini juga membuktikan bahwa stack dapat diimplementasikan dengan mudah menggunakan array dan variabel indeks sebagai penanda posisi teratas (top). Dengan memahami program ini, mahasiswa dapat lebih mengerti konsep dasar stack dan bagaimana struktur data ini digunakan dalam berbagai kasus pemrograman, seperti pembalikan data, penyimpanan sementara, dan pemrosesan input berurutan.

E. Referensi

Wikipedia. (2024). *Stack (abstract data type)*. Diakses dari https://en.wikipedia.org/wiki/Stack_(abstract_data_type)

Sedgewick, R., & Wayne, K. (2011). *Algorithms (4th Edition)*. Addison-Wesley.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in C++* (2nd Edition). Wiley.

Wirth, N. (1976). *Algorithms + Data Structures = Programs*. Prentice-Hall.