LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
Nama: Dina Nadhyfa
NIM : 103112430052

Dosen
Fahrudin Mukti Wibowo

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Stack merupakan salah satu jenis struktur data linier yang mengikuti prinsip Last In First Out (LIFO), yaitu elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan tumpukan benda, di mana hanya elemen teratas yang dapat diakses, ditambahkan (push), atau dihapus (pop). Stack memiliki komponen utama seperti elemen data dan pointer/top yang menunjuk elemen paling atas pada stack. Operasi utama pada stack adalah push, yaitu menyisipkan data pada posisi paling atas, dan pop, mengambil dan menghapus elemen pada posisi teratas. Stack dapat diimplementasikan menggunakan representasi pointer (linked list) atau array (representasi tabel). Perbedaan utama terletak pada pengelolaan memori; representasi array biasanya memiliki kapasitas terbatas tetapi akses indeks yang cepat, sedangkan pointer fleksibel dengan kapasitas yang dinamis.

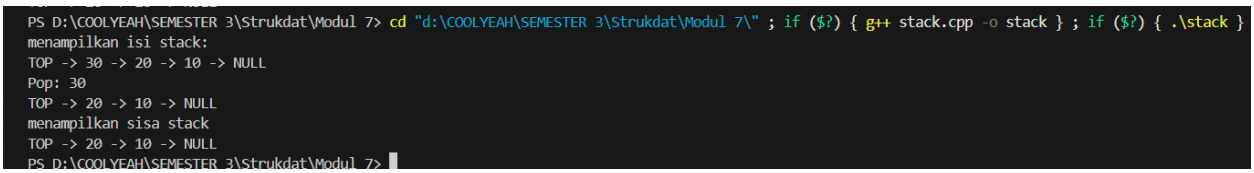
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

```cpp
// 103112430052_Dina Nadhyfa

#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" << endl;
    show(stack);

    return 0;
}
```

Screenshots Output



Deskripsi:

Program di atas mendefinisikan stack dengan node yang terdiri dari data integer dan pointer ke node berikutnya. Fungsi isEmpty digunakan untuk memeriksa apakah stack kosong, push untuk menambahkan elemen baru di top stack, pop untuk menghapus dan mengembalikan elemen dari top stack, serta show untuk menampilkan seluruh isi stack dari top ke bottom. Dalam fungsi main, program mendemonstrasikan operasi stack dengan menambahkan tiga elemen (10, 20, 30), menampilkan isi stack, melakukan pop satu elemen, dan menampilkan kembali sisa elemen dalam stack.

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

Unguided 1

stack.h

```h
// 103112430052_Dina Nadhyfa

#ifndef STACK_H
#define STACK_H

const int MAX_SIZE = 20;

typedef int infotype;

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

// Prototype fungsi
void createStack(Stack &S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

#endif
```

stack.cpp

```cpp
// 103112430052_Dina Nadhyfa

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

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

```cpp
void push(Stack &S, infotype x) {
    if (S.top < MAX_SIZE - 1) {
        S.top++;
        S.info[S.top] = x;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    } else {
        cout << "Stack kosong!" << endl;
        return -1; // Nilai default jika stack 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);

    // Pindahkan semua elemen dari S ke temp
    while (S.top >= 0) {
        push(temp, pop(S));
    }

    // Salin kembali dari temp ke S (sudah terbalik)
    S = temp;
}
```

main.cpp

```cpp
// 103112430052_Dina Nadhyfa

#include <iostream>
```

```cpp
#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

```
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> g++ main.cpp stack.cpp -o main
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> ./main
Hello world!
[TOP] 9 2 4 3
balik stack
[TOP] 3 4 2 9
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1>
```

Deskripsi:

Program di atas merupakan implementasi struktur data stack menggunakan array dengan kapasitas maksimum 20 elemen. Stack disimpan dalam struct yang memuat array `info` dan indeks `top` sebagai penanda elemen teratas. Program menyediakan operasi dasar seperti `createStack` untuk menginisialisasi stack, `push` untuk menambahkan elemen, `pop` untuk menghapus dan mengambil elemen teratas, serta `printInfo` untuk menampilkan isi stack dari elemen teratas ke terbawah. Selain itu, terdapat fungsi `balikStack` yang membalik urutan elemen dengan memindahkannya sementara ke stack lain. Pada fungsi `main`, beberapa data dimasukkan dan dihapus dari stack, ditampilkan, kemudian dibalik dan ditampilkan kembali.

Unguided 2

stack.h

```
// menambahkan prosedur baru, seperti di latihan soal nomor 2
void pushAscending(Stack &S, infotype x);
```

stack.cpp

```
// menambahkan prosedur pushAscending sesuai latihan soal nomor 2
void pushAscending(Stack &S, infotype x) {
    Stack temp;
    createStack(temp);

    // Pindahkan elemen dari S ke temp sampai menemukan posisi yang tepat
    while (S.top >= 0 && S.info[S.top] > x) {
        push(temp, pop(S));
    }

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

    // Kembalikan elemen dari temp ke S
    while (temp.top >= 0) {
        push(S, pop(temp));
    }
}
```

main.cpp

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

int main() {
    cout << "Hello world!" << endl;
    Stack S;
    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);
    return 0;
}
```

Screenshots Output

```
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> g++ main.cpp stack.cpp -o main
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> ./main
Hello world!
[TOP] 9 8 4 3 3 2
balik stack
[TOP] 2 3 3 4 8 9
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1>
```

Deskripsi:

Program di atas merupakan pengembangan dari program stack sebelumnya dengan penambahan prosedur pushAscending, yaitu prosedur untuk memasukkan data ke dalam stack secara berurutan naik (ascending). Saat memasukkan elemen baru, program memindahkan sementara elemen-elemen yang lebih besar ke stack lain, lalu memasukkan elemen baru pada posisi yang tepat, kemudian mengembalikan elemen-elemen tersebut kembali ke stack utama sehingga stack tetap terurut. Pada fungsi main, beberapa nilai dimasukkan menggunakan pushAscending, lalu seluruh isi stack ditampilkan. Setelah itu, stack dibalik menggunakan fungsi balikStack dan hasilnya ditampilkan kembali. Program ini menunjukkan cara menjaga urutan elemen dalam stack tanpa kehilangan sifat LIFO.

Unguided 3

stack.h

```h
// menambahkan prosedur baru, seperti di latihan soal nomor 3
void getInputStream(Stack &S);
```

stack.cpp

```cpp
// menambahkan prosedur getInputStream sesuai latihan soal nomor 3
void getInputStream(Stack &S) {
    char ch;

    // Baca karakter satu per satu hingga enter
    while (true) {
        ch = cin.get();

        // Jika tombol enter ditekan, keluar dari loop
        if (ch == '\n') {
            break;
        }

        // Jika karakter adalah digit, konversi ke integer dan push ke
stack
        if (isdigit(ch)) {
            int digit = ch - '0'; // Konversi char ke int
            push(S, digit);
        }
    }
}
```

main.cpp

```cpp
int main() {
    cout << "Hello world!" << endl;
    Stack S;
    createStack(S);
    getInputStream(S);
    printInfo(S);
    cout << "balik stack" << endl;
    balikStack(S);
    printInfo(S);
    return 0;
}
```

Screenshots Output

```
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> g++ main.cpp stack.cpp -o main
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1> ./main
Hello world!
4729601
[TOP] 1 0 6 9 2 7 4
balik stack
[TOP] 4 7 2 9 6 0 1
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 7\unguided\soal 1>
```

Deskripsi:

Program di atas menambahkan prosedur getInputStream yang berfungsi membaca input karakter dari pengguna secara langsung hingga tombol Enter ditekan. Setiap karakter yang terbaca akan dicek, dan jika merupakan digit angka (0–9), maka dikonversi menjadi bilangan integer dan dimasukkan ke dalam stack menggunakan fungsi push. Setelah input selesai, isi stack ditampilkan melalui printInfo, lalu stack dibalik menggunakan balikStack dan ditampilkan kembali. Program ini menunjukkan cara membaca input karakter per karakter dan menyimpannya sebagai angka di dalam stack.

E. Kesimpulan

Kesimpulannya, program-program di atas menunjukkan cara mengimplementasikan struktur data **stack** beserta berbagai operasi tambahan. Selain operasi dasar seperti push, pop, dan printInfo, program juga menambahkan fungsi untuk **membalik isi stack**, **memasukkan data secara terurut (ascending)**, dan **membaca input berupa karakter lalu menyimpannya ke stack**. Dengan demikian, program ini memperlihatkan bagaimana stack dapat dimodifikasi dan dimanfaatkan dalam berbagai kebutuhan pengolahan data.

F. Referensi

Modul-07-Struktur Data

https://rcf-indonesia.org/jurnal/index.php/jsit/article/view/591

https://en.wikipedia.org/wiki/Stack_(abstract_data_type)