LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
DOUBLY LINKED LIST

Disusun Oleh :
NAMA : Ridha Akifah
NIM : 103112400132

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Stack (Tumpukan) adalah *Abstract Data Type* (ADT) linear yang beroperasi dengan prinsip LIFO (Last-In, First-Out), di mana elemen yang terakhir dimasukkan (*paling atas*) akan menjadi yang pertama kali dikeluarkan. Implementasi Stack menggunakan Representasi Tabel (Array) memiliki prinsip dasar yang sama dengan *Linked List* tetapi menggunakan array berindeks dengan jumlah tumpukan yang terbatas. Seluruh operasi Stack hanya dapat dilakukan melalui satu titik akses, yaitu TOP. Operasi utama terdiri dari PUSH, yaitu penyisipan data ke dalam *array* yang dilakukan dengan menggeser indeks TOP ke indeks berikutnya (Top = Top + 1) , dan POP, yaitu pengambilan data di posisi TOP diikuti dengan pergeseran mundur indeks TOP (TOP = TOP - 1) , sementara Stack kosong biasanya ditandai dengan inisialisasi Top.

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

Guided 1

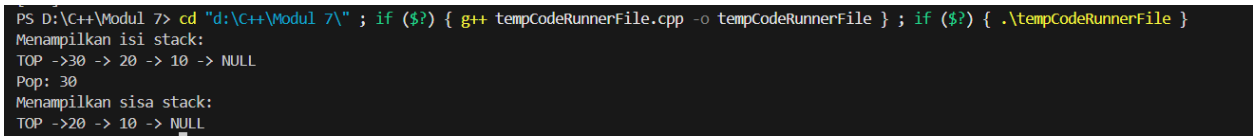
```cpp
#include <iostream>
using namespace std;

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 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;
}
```

Screenshoot Output



Deskripsi:

Program C++ ini mengimplementasikan Stack (Tumpukan) menggunakan Linked List dinamis, di mana operasi push (memasukkan) dan pop (mengeluarkan) selalu dilakukan pada elemen teratas (LIFO), dengan *pointer* top sebagai satu-satunya titik akses.

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

Unguided

```
#include <iostream>
#include <cctype>
using namespace std;

const int MAX = 20;
typedef int infotype;

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

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

infotype pop(Stack &S) {
    if (!isEmpty(S)) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    } else {
        cout << "Stack kosong!" << endl;
        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 (!isEmpty(S)) {
        push(temp, pop(S));
    }
    S = 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) {
```

```
    cout << "Masukkan angka (akhiri dengan Enter): ";
    char ch;
    while (true) {
        ch = cin.get();
        if (ch == '\n') break;
        if (isdigit(ch)) {
            int val = ch - '0';
            push(S, val);
        }
    }
}

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);


    cout << "\nSekarang coba getInputStream:\n";
    createStack(S);
    getInputStream(S);
    printInfo(S);
    cout << "balik stack" << endl;
    balikStack(S);
    printInfo(S);


    return 0;
}
```

Screenshoot Output

```
PS D:\C++\Modul 7> cd "d:\C++\Modul 7\" ; if ($?) { g++ unguided.cpp -o unguided } ; if ($?) { .\unguided }
Hello world!
[TOP] 2 3 3 4 8 9
balik stack
[TOP] 9 8 4 3 3 2

Sekarang coba getInputStream:
Masukkan angka (akhiri dengan Enter): 5 6 8 3
[TOP] 3 8 6 5
balik stack
[TOP] 5 6 8 3
```

Deskripsi:

Program C++ ini mengimplementasikan Tumpukan Data (Stack) menggunakan Array dengan ukuran tetap (20), yang berarti data dimasukkan dan dikeluarkan hanya melalui posisi teratas sesuai prinsip LIFO (Last-In, First-Out). Selain fungsi dasar (push dan pop), program ini dilengkapi dengan fitur khusus: pushAscending untuk memasukkan angka sambil menjaga tumpukan selalu terurut menaik, serta getInputStream untuk membaca setiap digit angka dari *stream* input pengguna dan memasukkannya ke dalam tumpukan.

D. Kesimpulan

Stack (Tumpukan) adalah struktur data yang wajib mengikuti prinsip LIFO (Last-In,

First-Out), artinya elemen yang terakhir dimasukkan akan menjadi yang pertama kali dikeluarkan. Implementasi Stack berhasil dilakukan menggunakan Representasi Tabel (Array), di mana variabel TOP berfungsi sebagai satu-satunya titik akses untuk operasi penyisipan (PUSH) dan pengambilan (POP). Prinsip PUSH dilakukan dengan menaikkan indeks TOP diikuti dengan penyisipan data, sementara POP dilakukan dengan mengambil data di TOP lalu menurunkan indeks TOP. Selain operasi dasar, modul ini menegaskan bahwa fungsionalitas Stack dapat diperluas, seperti pushAscending untuk menjaga urutan data dan balikStack untuk membalik urutan, yang semuanya memperkuat pemahaman mendalam tentang manipulasi data berdasarkan disiplin LIFO.

E. Referensi

https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B

https://learn.microsoft.com/id-id/cpp/cpp/references-cpp?view=msvc-170

https://medium.com/@mohamedeissabay/understanding-c-a-deep-dive-into-core-concepts-48a560679cdd