LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
NAMA : ZAKI MAULA DHIA
NIM : 103112400127

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Stack atau tumpukan merupakan struktur data linier yang mengikuti prinsip LIFO (Last In First Out), yaitu elemen yang terakhir dimasukkan ke dalam tumpukan merupakan elemen yang pertama kali akan dibuang dari tumpukan. Stack merupakan struktur data linier dimana penambahan elemen baru ke dalam tumpukan dan pengambilan elemen dari tumpukan yang ada hanya dapat dilakukan pada tempat yang sama, yaitu pada bagian pertama.

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

Guided 1

```
#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

```
…/strukturdata/modul 7test ❯ ./stack
Menampilkan isi stack:
TOP -> 30 -> 20 -> 10 -> NULL
Pop: 30
Menampilkan sisa stack:
TOP -> 20 -> 10 -> NULL
```

Deskripsi:

Program ini menunjukan cara menggunakan struktur data stack, dengan menggunakan operasi dasar seperti push, pop, dan show yang menggunakan pointer untuk melihat setiap isi dari stack. Program akan memasukan angka yang sudah ditentukan kedalam stack dan kemudian program akan menghapus elemen pertama dari isi stack.

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

Unguided 1

```
// 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, 9);
    pop(S);
    push(S, 9);
    printInfo(S);

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

```
    printInfo(S);

    return 0;
}
```

```
// stack.h
#ifndef STACK_H
#define STACK_H

#include <iostream>
using namespace std;

const int maxElement = 20;
typedef int infotype;

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

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
#include "stack.h"
void createStack(Stack &S) {
    S.top = -1;
}

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

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

Screenshots Output

```
…/modul 7test/unguided1 ❯ ./stack
Hello world!
[TOP] 9 2 4 3
balik stack
[TOP] 3 4 2 9
```

Deskripsi:

Program memiliki fungsi untuk mengimplementasikan struktur data stack menggunakan array, data akan diakses dengan prinsip LIFO, element pertama yang dimasukan akan akan menjadi angka terakhir, dan kemudian dengan menggunakan function pop() data yang terakhir dimasukan atau data pertama ada di stack akan dihapus. Karena program ini menggunakan prinsip LIFO, maka data harus di balik sehingga kita tahu data mana yang pertama kali dimasukan.

Unguided 2

```cpp
// 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;
```

```
}
```

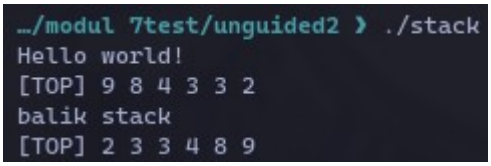
```
// stack.h
#ifndef STACK_H
#define STACK_H

#include <iostream>
using namespace std;

const int maxElement = 20;
typedef int infotype;

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

void createStack(Stack &S);
void push(Stack &S, infotype x);
void pushAscending(Stack &S, int x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

#endif
```

```
// stack.cpp
#include "stack.h"
void createStack(Stack &S) {
    S.top = -1;
}

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

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

void push(Stack &S, infotype x) {
```

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

void pushAscending(Stack &S, int 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));
    }
}

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;
}
```

Screenshots Output



Deskripsi:

Program unguided 1 yang ditambah dengan function pushAscending(), yang berfungsi untuk menambahkan element ke dalam array secara urut dari yang terkecil ke yang terbesar.

Unguided 3

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

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

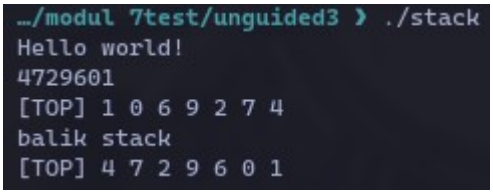
```
// stack.h
#ifndef STACK_H
#define STACK_H

#include <iostream>
using namespace std;

const int maxElement = 20;
typedef int infotype;

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

void createStack(Stack &S);
void push(Stack &S, infotype x);
void pushAscending(Stack &S, int x);
void getInputStream(Stack &S);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

#endif
```

```
// stack.cpp
#include "stack.h"
void createStack(Stack &S) {
    S.top = -1;
}

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

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

void push(Stack &S, infotype x) {
    if (!isFull(S)) {
        S.top++;
        S.info[S.top] = x;
```

```
    } else {
        cout << "Stack penuh!" << endl;
    }
}

void pushAscending(Stack &S, int 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));
    }
}

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 getInputStream(Stack &S) {
    char ch;

    while (true) {
        ch = cin.get();
        if (ch == '\n') break;
        if (ch >= '0' && ch <= '9') {
            int x = ch - '0';
            push(S, x);
        }
    }
}
```

Screenshots Output



Deskripsi:

Program yang sama seperti di nomor 1 dan 2 tetapi di soal 3 ditambah function untuk menambahkan inputan user yang berupa type data string, sehingga harus di ubah terlebih dahulu sebelum dimasukan kedalam array.

D. Kesimpulan

kesimpulan dari materi doubly linked list yang telah dipelajari pada minggu 7 bahwa program stack merupakan salah satu bentuk struktur data yang menggunakan prinsip LIFO(Last In First Out), yaitu element yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Stack dapat diimplementasikan menggunakan pointer atau array. Dengan memahami konsep dan operasi pada stack, user dapat mengaplikasikan struktur data ini untuk menyelesaikan berbagai permasalahan yang memerlukan pengelolaan data secara terstruktur dan efisien.

E. Referensi

- Ginting, S. H. N., Effendi, H., Kumar, S., Marsisno, W., Sitanggang, Y. R. U., Anwar,

K., ... & Smrti, N. N. E. (2024). Pengantar struktur data. *Penerbit Mifandi Mandiri Digital*, *1*(01).