LAPORAN PRAKTIKUM

STRUKTUR DATA

MODUL VII

STACK

Disusun Oleh :

Nama : Izzah Minkhotun Fannisa

NIM : 103112400198

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Struktur data merupakan cara untuk mengorganisasikan dan menyimpan data di dalam komputer agar dapat digunakan secara efisien. Salah satu struktur data dasar yang penting dalam pemrograman adalah stack. Stack merupakan struktur data linear yang menerapkan prinsip *Last In First Out (LIFO),* yaitu elemen yang terakhir dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan tumpukan benda, seperti tumpukan buku, di mana buku yang diletakkan terakhir akan diambil terlebih dahulu.

Stack memiliki dua operasi utama, yaitu push untuk menambahkan elemen ke bagian atas stack dan pop untuk menghapus elemen teratas dari stack. Selain itu, stack juga memiliki operasi pendukung seperti pengecekan kondisi kosong (isEmpty), pengecekan kondisi penuh (isFull), serta penampilan isi stack. Elemen teratas pada stack selalu ditunjuk oleh sebuah penunjuk yang disebut top, yang berfungsi untuk mengetahui posisi data terakhir yang masuk.

Implementasi stack dapat dilakukan dengan beberapa cara, di antaranya menggunakan *array* dan *linked list*. Stack berbasis array menggunakan larik dengan ukuran tertentu sebagai tempat penyimpanan data, sehingga kapasitas stack bersifat statis. Pendekatan ini lebih sederhana dan memiliki akses data yang cepat, namun memiliki keterbatasan pada jumlah elemen yang dapat disimpan. Sebaliknya, stack berbasis *linked list* bersifat dinamis karena setiap elemen disimpan dalam node yang saling terhubung menggunakan pointer, sehingga kapasitas stack dapat bertambah sesuai kebutuhan selama memori masih tersedia.

B. Guide

1. Guide 1

a. Source Code

stack.h

```
#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;
}
```

b. Screenshot Output

```
menampilkan isi stack:
TOP -> 30 -> 20 -> 10 -> NULL
Pop: 30
TOP -> 20 -> 10 -> NULL
menampilkan sisa stack: 20
TOP -> 10 -> NULL
```

c. Deskripsi Program

Program ini dibuat untuk mengimplementasikan struktur data *stack* menggunakan *linked list* dalam bahasa pemrograman C++. Program mendefinisikan sebuah struktur Node yang berisi data bertipe integer dan pointer ke node berikutnya, sehingga setiap elemen stack saling terhubung secara dinamis. Stack dikelola menggunakan sebuah pointer top yang selalu menunjuk ke elemen teratas.

Operasi utama yang disediakan meliputi pengecekan kondisi stack kosong melalui fungsi isEmpty, penambahan data ke stack menggunakan fungsi push, penghapusan data teratas dengan fungsi pop, serta penampilan seluruh isi stack melalui fungsi show. Pada fungsi push, data baru ditambahkan ke bagian atas stack dengan mengubah pointer top, sedangkan pada fungsi pop, data teratas diambil lalu node tersebut dihapus dari memori.

Program utama (main) digunakan untuk menguji fungsi-fungsi tersebut dengan menambahkan beberapa data ke *stack*, menampilkan isi *stack*, serta melakukan proses penghapusan data dan menampilkan sisa elemen yang masih tersimpan. Dengan demikian, program ini menunjukkan cara kerja stack yang menerapkan prinsip *Last In First Out (LIFO)* menggunakan struktur data linked list.

C. Unguide

1. Unguide 1

a. Source Code

stack.h

```
#ifndef STACK_H
```

```h
#define STACK_H

#define MAX_STACK 20
typedef int infotype;

struct Stack {
    infotype info[MAX_STACK];
    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

```cpp
#include <iostream>
#include "stack.h"

using namespace std;

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

void push(Stack &S, infotype x) {
    if (S.top < MAX_STACK - 1) {
        S.top++;
        S.info[S.top] = x;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    }
    return 0;
}

void printInfo(Stack S) {
    if (S.top == -1) {
        cout << "Stack kosong" << endl;
    } else {
```

```
        cout << "[TOP] ";
        for (int i = S.top; i >= 0; i--) {
            cout << S.info[i] << " ";
        }
        cout << endl;
    }
}

void balikStack(Stack &S) {
    int i = 0;
    int j = S.top;
    while (i < j) {
        infotype temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}
```

main.cpp

```
#include <iostream>
#include "stack.h"
#include "stack.cpp"

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;
}
```

b. Screenshot Output

```
Hello world!
[TOP] 9 2 4 3
balik stack
[TOP] 3 4 2 9
```

c. Deskripsi Program

Program ini merupakan implementasi *Abstract Data Type (ADT) Stack* menggunakan array dalam bahasa pemrograman C++.

a) Stack menerapkan prinsip *Last In First Out (LIFO),* yaitu elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan.

b) Pada file stack.h, didefinisikan struktur Stack yang terdiri dari array info sebagai tempat penyimpanan data dan variabel top sebagai penunjuk elemen teratas pada stack.

c) File stack.cpp berisi implementasi fungsi-fungsi utama stack, yaitu:
- createStack untuk menginisialisasi stack agar berada dalam kondisi kosong.
- push untuk menambahkan elemen ke dalam stack selama kapasitas belum penuh.
- pop untuk menghapus dan mengembalikan elemen teratas dari stack.
- printInfo untuk menampilkan isi stack mulai dari elemen teratas hingga terbawah.
- balikStack untuk membalik urutan elemen di dalam stack.

d) File main.cpp digunakan untuk menguji fungsi-fungsi stack dengan melakukan serangkaian operasi penambahan dan penghapusan elemen.

e) Program juga menampilkan isi stack sebelum dan sesudah proses pembalikan untuk menunjukkan perubahan urutan data.

f) Secara keseluruhan, program ini menunjukkan cara kerja stack berbasis array secara terstruktur dan sesuai dengan konsep dasar ADT Stack.

2. Unguide 2

a. Source Code

stack.h

```
#ifndef STACK_H
#define STACK_H

#define MAX_STACK 20
typedef int infotype;
```

```
struct Stack {
    infotype info[MAX_STACK];
    int top;
};

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

#endif
```

stack.cpp

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

using namespace std;

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

void push(Stack &S, infotype x) {
    if (S.top < MAX_STACK - 1) {
        S.top++;
        S.info[S.top] = x;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    }
    return 0;
}

void printInfo(Stack S) {
    if (S.top == -1) {
        cout << "Stack kosong" << endl;
    } else {
        cout << "[TOP] ";
        for (int i = S.top; i >= 0; i--) {
            cout << S.info[i] << " ";
```

```
        }
        cout << endl;
    }
}

void balikStack(Stack &S) {
    int i = 0;
    int j = S.top;
    while (i < j) {
        infotype temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}

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

    while (S.top != -1 && S.info[S.top] > x) {
        push(temp, pop(S));
    }

    push(S, x);

    while (temp.top != -1) {
        push(S, pop(temp));
    }
}
```

main.cpp

```
#include <iostream>
#include "stack.h"
#include "stack.cpp"

using namespace std;

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

    Stack S;
    createStack(S);

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

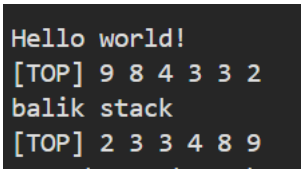
```
    pushAscending(S, 2);
    pushAscending(S, 3);
    pushAscending(S, 9);

    printInfo(S);

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

    return 0;
}
```

b. Screenshot Output



c. Deskripsi Program

Program ini merupakan implementasi Abstract Data Type (ADT) Stack berbasis array menggunakan bahasa pemrograman C++. Stack memiliki kapasitas maksimum sebanyak 20 elemen yang ditentukan oleh konstanta MAX_STACK. Struktur stack mengikuti prinsip *Last In First Out (LIFO),* di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Struktur data stack didefinisikan pada file stack.h yang terdiri dari array info sebagai tempat penyimpanan data dan variabel top sebagai penunjuk elemen teratas stack.

File stack.cpp berisi implementasi fungsi-fungsi utama stack, yaitu:

a) createStack untuk menginisialisasi stack dalam kondisi kosong.
b) push untuk menambahkan elemen ke dalam stack selama kapasitas belum penuh.
c) pop untuk menghapus dan mengembalikan elemen teratas dari stack.
d) printInfo untuk menampilkan isi stack dari elemen teratas hingga terbawah.
e) balikStack untuk membalik urutan elemen di dalam stack.
f) pushAscending untuk memasukkan elemen ke dalam stack dengan tetap menjaga urutan data secara menaik *(ascending)*.

Prosedur pushAscending bekerja dengan memanfaatkan sebuah stack sementara. Elemen-elemen pada stack utama yang memiliki nilai lebih besar dari nilai yang akan dimasukkan dipindahkan terlebih dahulu ke stack sementara. Setelah elemen baru dimasukkan ke stack utama, seluruh elemen pada stack sementara dikembalikan ke stack utama

sehingga urutan data tetap terjaga. Pendekatan ini memungkinkan stack tetap terurut tanpa melanggar prinsip dasar operasi stack.

Pada file main.cpp, program diuji dengan memasukkan beberapa data menggunakan prosedur pushAscending, kemudian menampilkan isi stack sebelum dan sesudah dilakukan pembalikan menggunakan prosedur balikStack. Hasil keluaran program menunjukkan bahwa data dapat dimasukkan secara terurut dan urutan stack dapat dibalik sesuai dengan prosedur yang dijalankan. Program ini menunjukkan bagaimana ADT Stack dapat dikembangkan tidak hanya sebagai struktur LIFO, tetapi juga untuk memenuhi kebutuhan pengolahan data yang lebih kompleks seperti pengurutan elemen.

3. Unguide 3

a. Source Code

stack.h

```h
#ifndef STACK_H
#define STACK_H

#define MAX_STACK 20
typedef int infotype;

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

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

#endif
```

stack.cpp

```cpp
#include <iostream>
#include "stack.h"

using namespace std;

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

```
void push(Stack &S, infotype x) {
    if (S.top < MAX_STACK - 1) {
        S.top++;
        S.info[S.top] = x;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    }
    return 0;
}

void printInfo(Stack S) {
    if (S.top == -1) {
        cout << "Stack kosong" << endl;
    } else {
        cout << "[TOP] ";
        for (int i = S.top; i >= 0; i--) {
            cout << S.info[i] << " ";
        }
        cout << endl;
    }
}

void balikStack(Stack &S) {
    int i = 0;
    int j = S.top;
    while (i < j) {
        infotype temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}

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

    while (S.top != -1 && S.info[S.top] > x) {
        push(temp, pop(S));
    }
```

```
    push(S, x);

    while (temp.top != -1) {
        push(S, pop(temp));
    }
}

void getInputStream(Stack &S) {
    cout << "Masukkan input (ENTER untuk selesai): ";
    char c;
    while ((c = cin.get()) != '\n') {
        push(S, c - '0');
    }
}
```

main.cpp

```
#include <iostream>
#include "stack.h"
#include "stack.cpp"

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;
}
```

b. Screenshot Output

```
Hello world!
Masukkan input (ENTER untuk selesai): 4729601
[TOP] 1 0 6 9 2 7 4
balik stack
[TOP] 4 7 2 9 6 0 1
```

c. Deskripsi Program

Program ini mengimplementasikan struktur data stack berbasis array menggunakan bahasa C++ dengan kapasitas maksimum 20 elemen. Stack bekerja dengan prinsip *Last In First Out (LIFO)* dan direpresentasikan menggunakan array sebagai penyimpan data serta variabel top sebagai penunjuk elemen teratas. Program menyediakan operasi dasar stack, yaitu inisialisasi, penambahan data *(push)*, penghapusan data *(pop),* dan penampilan isi stack (printInfo).

Selain operasi dasar, program dilengkapi dengan fungsi tambahan seperti balikStack untuk membalik urutan elemen di dalam stack dan pushAscending untuk memasukkan data dengan urutan menaik menggunakan stack sementara. Fungsi getInputStream memungkinkan pengguna memasukkan data angka secara langsung melalui input keyboard hingga menekan ENTER, kemudian data tersebut disimpan ke dalam stack.

Pada bagian utama program, stack dibuat dan diisi menggunakan input pengguna, lalu isi stack ditampilkan. Setelah itu, stack dibalik dan ditampilkan kembali untuk menunjukkan perubahan urutan data. Program ini menunjukkan penerapan stack dalam pengelolaan data, manipulasi urutan, serta pemrosesan input secara dinamis.

D. Kesimpulan

Berdasarkan praktikum Modul VII tentang struktur data stack, dapat disimpulkan bahwa stack merupakan struktur data yang bekerja dengan *prinsip Last In First Out (LIFO),* di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Melalui praktikum ini, telah dipelajari dan diimplementasikan stack menggunakan dua pendekatan, yaitu berbasis *linked list* dan berbasis array, yang masing-masing memiliki karakteristik dan cara pengelolaan memori yang berbeda.

Implementasi stack berbasis *linked list* menunjukkan fleksibilitas dalam penggunaan memori karena bersifat dinamis, sedangkan stack berbasis array memiliki keterbatasan kapasitas namun lebih sederhana dalam implementasinya. Selain operasi dasar seperti push, pop, dan pengecekan kondisi stack, praktikum ini juga mengembangkan operasi lanjutan seperti pembalikan urutan stack (balikStack), penyimpanan data secara terurut (pushAscending), serta pengambilan input secara langsung dari pengguna (getInputStream).

Secara keseluruhan, praktikum ini memberikan pemahaman yang lebih mendalam mengenai konsep ADT Stack serta penerapannya dalam pemrograman C++. Mahasiswa diharapkan mampu memahami cara kerja stack, mengimplementasikan berbagai operasi stack, serta mengembangkan logika program untuk menyelesaikan permasalahan yang lebih kompleks dengan memanfaatkan struktur data stack.

E. Referensi

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms (3rd ed.).* MIT Press.

GeeksforGeeks. (n.d.). *Stack data structure*. https://www.geeksforgeeks.org/stack-data-structure/

Malik, D. S. (2015). C++ *programming: From problem analysis to program design (7th ed.).* Cengage Learning.

Weiss, M. A. (2014). *Data structures and algorithm analysis in C++ (4th ed.).* Pearson Education.