LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun oleh :

Junadil Muqorobin (103112400281)

Dosen

Fahrudin Mukti Wibowo S.Kom., M.Eng

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Stack merupakan salah satu struktur data linear yang menerapkan prinsip Last In First Out (LIFO), yaitu data yang terakhir masuk akan menjadi data pertama yang keluar. Seluruh proses manipulasi data hanya dilakukan pada satu ujung struktur yang disebut Top, sehingga stack bersifat terbatas dalam akses data. Dalam modul praktikum disebutkan bahwa operasi utama pada stack adalah push untuk menambah elemen dan pop untuk mengambil elemen dari bagian atas stack

Menurut penelitian oleh Siregar dan Manalu (2022), struktur data stack memiliki peran penting dalam proses komputasi seperti pengelolaan *function call*, algoritma rekursif, parsing ekspresi matematika, serta pengelolaan memori pada program yang kompleks. Implementasi stack umumnya dilakukan melalui dua pendekatan yaitu menggunakan array (statis) dan pointer/linked list (dinamis). Representasi array mudah diterapkan namun memiliki batas kapasitas tertentu, sedangkan representasi pointer lebih fleksibel karena dapat menyesuaikan jumlah data sesuai memori yang tersedia (Putri & Kurniawan, 2021).

Operasi dasar seperti push dan pop memiliki kompleksitas waktu O(1) karena hanya melibatkan perpindahan top tanpa melakukan traversal keseluruhan struktur, sehingga sangat efisien pada proses yang membutuhkan akses cepat (Ramdani & Nugroho, 2020). Sebagai tambahan, terdapat fungsi isEmpty() untuk mengecek kondisi stack kosong dan isFull() pada implementasi berbasis array untuk memastikan kapasitas tidak melebihi batas.

Stack juga memiliki sejumlah aplikasi penting dalam pengembangan perangkat lunak dan pemrosesan data. Dewi dan Pratama (2023) menjelaskan bahwa stack digunakan dalam proses *undo-redo* pada aplikasi editor, algoritma Depth First Search (DFS), penyimpanan status program dalam sistem operasi, serta evaluasi ekspresi aritmatika. Oleh karena sifat operasional yang efisien dan struktur yang sederhana, stack menjadi struktur data fundamental yang wajib dipahami dalam ilmu komputer, seperti yang ditegaskan pada modul praktikum ini

B. Guided

1. Implementasi Stack

```
#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 ini mengimplementasikan struktur data Stack menggunakan Linked List di bahasa C++. Stack bekerja dengan prinsip LIFO (Last In First Out), artinya data terakhir yang dimasukkan akan menjadi data pertama yang dikeluarkan. Program berjalan dengan langkah kerja sebagai berikut:

- Definisikan node

  Setiap elemen stack disebut Node, int data untuk menyimpan data dan pointer next untuk menunjuk ke Node berikutnya.

- Fungsi isEmpty()

  Fungsi isEmpty() berfungsi untuk mengecek apakah stack kosong. Jika top == nullptr , maka artinya tidak ada data di stack.

- Prosedur push()

  Langkah kerja pada prosedur push yaitu pertama-tama membuat node baru, kemudian isi node baru dengan data, next dari node baru menunjuk ke top sebelumnya, kemudian top digeser menjadi node baru (data bar berada paling atas).

- Fungsi pop()

  Langkah kerja pertama dengan mengecek apakah stack kosong dengan percabangan if,

kemudian ambil data dari top, kemudian pindahkan top ke node berikutnya, hapus node sebelumnya dari memori untuk menghindari memory leak, kemudian kembalikan nilai yang dipop dengan return poppedData.

- Prosedur show()

  Prosedur ini digunakan untuk menampilkan seluruh stack, mulai dari top hingga stack terakhir yang menunjuk ke NULL.

- Fungsi main()

  Pada fungsi main() pertama-tama membuat stack kosong, kemudian menambahkan tiga angka ke stack yaitu 10, 20, 30. Kemudian menampilkan semua isi stack dari yang paling atas. Setelah itu melakukan pop satu data paling atas dan menampilkan data yang terhapus. Kemudian terakhir menampilkan kembali isi stack setelah salah satu data di pop atau dihapus.

C. Unguided

1. Membuat ADT Stack Menggunakan Array

Source code stack.h

```
#ifndef STACK_H
#define STACK_H

#define MAX 20
typedef int infotype;

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

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

// TAMBAHAN SOAL 2
void pushAscending(Stack &S, infotype x);

// TAMBAHAN SOAL 3
void getInputStream(Stack &S);

#endif
```

Source code stack.cpp

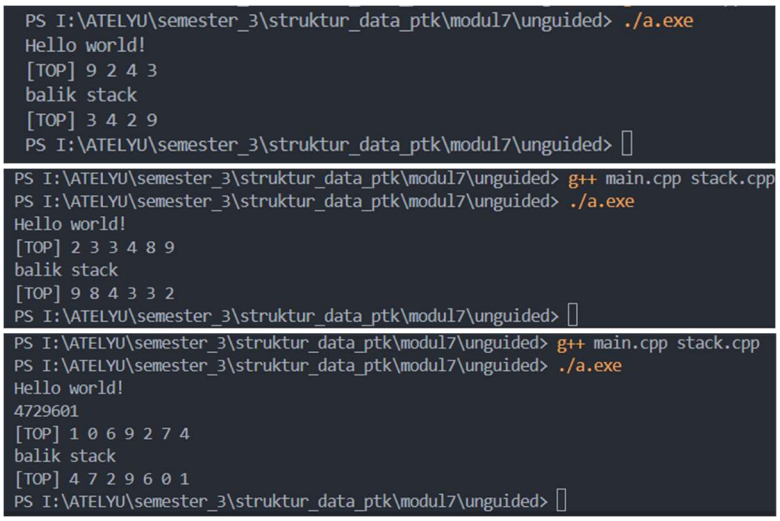
```
#include <iostream>
#include "stack.h"
using namespace std;

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

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

infotype pop(Stack &S) {
    if (S.top == -1) {
        cout << "Stack kosong!" << endl;
        return -1;
    } else {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    }
}
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 (S.top != -1) {
        push(Temp, pop(S));
    }

    S = Temp;
}

// TAMBAHAN SOAL 2
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));
    }
}

// TAMBAHAN SOAL 3
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);
        }
    }
}
```

Source code main.cpp

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

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

    // SOAL 1
    push(S, 3);
    push(S, 4);
    push(S, 8);
    pop(S);
    push(S, 2);
    push(S, 3);
    pop(S);
    push(S, 9);

    // TAMBAHAN SOAL 2
    pushAscending(S,3);
    pushAscending(S,4);
    pushAscending(S,8);
    pushAscending(S,2);
    pushAscending(S,3);
    pushAscending(S,9);

    // TAMBAHAN SOAL 3
    getInputStream(S);

    printInfo(S);

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

    return 0;
}
```

Screenshot Output:



Deskripsi:

Program diatas adalah impelementasi ADT Stack dengan menggunakan array statis, stack bekerja denan prinsip LIFO yaitu Last In First Out dimana elemen yang terakhir masuk maka keluar pertama. Program berjalan dengan alur kerja pada file main.cpp sebagai berikut:

- Inisialisasi stack

  Program menampilkan teks Hello world!, kemudian variable stask S dibuat dan diinisialisasi menggunakan createStack(S) agar top = -1.

- Proses pada soal 1
  - Pertama-tama menambahkan tiga nilai ke dalam stack yaitu menggunakan push(S,3), push(S,4), push(S,8).
  - Kemdian menghapus satu elemen teratas yaitu sementara 8, dengan pop(S).
  - Selanjutnya menambahkan nilai 2 dan 3 dengan ke push ke stack

- Kemudian menjalankan pop(S) kembali untuk menghapus nilai 3 pada top.
- Terakhir push(S,9) untuk menambahkan nilai terbaru pada stack. Sehingga top yang terakhir yaitu jadi nilai 9 sebagai top.

- Proses pada soal 2
  - Proses awal masih serupa dengan soal nomor 1, namun sekarang stack diisi menggunakan fungsi khusus yaitu pushAscending().
  - Jika elemen baru lebih besar dari elemen teratas, maka elemen tersebut dipindah sementara ke stack lain.
  - Setelah posisi yang sesuai ditemukan, baru elemen baru tersebut dimasukkan.
  - Kemudian elemen sementara dikembalikan ke stack awal sehingga hasil akhirnya stack akan tersusun secara ascending yaitu dari bawah ke atas.
- Proses pada soal 3
  - Konspenya masih sama dengan soal sebelumnya, namun disini menggunakan getInputStream() yaitu fungsi untuk menerima input dari user. Ketika user menekan enter maka akan menhentikan input dan lanjut ke perhitungan. Setiap angka yang di inputkan oleh user akan diubah ke tipe integer dan akakn dimasukkan ke dalam stack.

- Fungsi printInfo(S)
  Fungsi ini digunakan untuk menampilkan seluruh isi stack dimulai dari elemen TOP sampai akhir stack.
- Fungsi balikStack(S)
  Fungsi ini digunakan untuk membalik susunan elemen stack

D. Kesimpulan

Pada praktikum modul ini, saya telah mempelajari dan menerapkan konsep dasar struktur data Stack dengan menggunakan array sebagai media penyimpanan. Melalui rangkaian percobaan dan unguided, dapat disimpulkan bahwa Stack memiliki karakteristik **Last In First Out (LIFO)**, di mana elemen terakhir yang masuk akan menjadi elemen pertama yang keluar.

Dari hasil implementasi fungsi-fungsi dasar seperti push, pop, printInfo, dan balikStack, dapat terlihat bahwa pengelolaan data dalam stack tetap teratur sesuai prinsip LIFO. Selain itu, pengembangan prosedur pushAscending() menunjukkan bahwa stack juga dapat digunakan dalam kondisi khusus, seperti memasukkan data yang harus tetap terurut meskipun tetap mengikuti mekanisme penyimpanan berbasis LIFO. Fungsi getInputStream() menambah wawasan dalam bagaimana stack dapat menangani input berbasis karakter secara dinamis dari pengguna. Secara umum, praktikum ini memberikan pengalaman langsung dalam mengelola memori dan data menggunakan array pada struktur stack. Saya juga belajar bagaimana algoritma pendukung disusun untuk memanipulasi data di dalam stack tanpa melanggar sifat utamanya.

E. Referensi

Dewi, R., & Pratama, F. (2023). Implementasi struktur data stack dalam evaluasi ekspresi aritmatika pada kompilator sederhana. *Jurnal Teknologi Informatika*, 11(2), 87–95.

Putri, A., & Kurniawan, R. (2021). Analisis penggunaan stack dengan array dan linked list pada aplikasi berbasis data dinamis. *Jurnal Sistem Informasi dan Sains Teknologi*, 5(3), 155–162.

Ramdani, A., & Nugroho, S. (2020). Studi komparatif efisiensi operasi push dan pop pada beberapa metode implementasi stack. *Jurnal Ilmiah Teknologi dan Rekayasa*, 9(1), 30–38.

Siregar, D., & Manalu, T. (2022). Perancangan struktur data stack untuk pengelolaan memori pada algoritma rekursif. *Jurnal Komputer dan Informatika*, 14(1), 41–50.