LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :
NAMA : Afief Amar Purnomo
NIM : 103112430067

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Struktur data stack merupakan salah satu tipe struktur data linier yang beroperasi berdasarkan prinsip Last In First Out (LIFO), dimana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dapat diakses atau dihapus. Konsep stack dapat dianalogikan seperti tumpukan piring, dimana untuk mengambil piring paling atas, piring yang ada di bawah harus menunggu sampai yang di atas diambil terlebih dahulu. Stack memiliki operasi utama berupa push (menambah elemen ke atas tumpukan) dan pop (menghapus elemen dari atas tumpukan).

Struktur data ini banyak digunakan dalam berbagai aplikasi pemrograman seperti manajemen memori, evaluasi ekspresi matematika, algoritma rekursif, serta undo-redo pada aplikasi perangkat lunak. Implementasi stack dapat dilakukan menggunakan array atau linked list, dimana masing-masing memiliki kelebihan dalam hal efisiensi memori dan kecepatan akses. Kelebihan stack adalah kesederhanaannya dan kemampuan mengakses data secara cepat pada bagian atas tumpukan, namun memiliki keterbatasan dalam hal kapasitas dan tidak mendukung akses elemen secara acak.

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

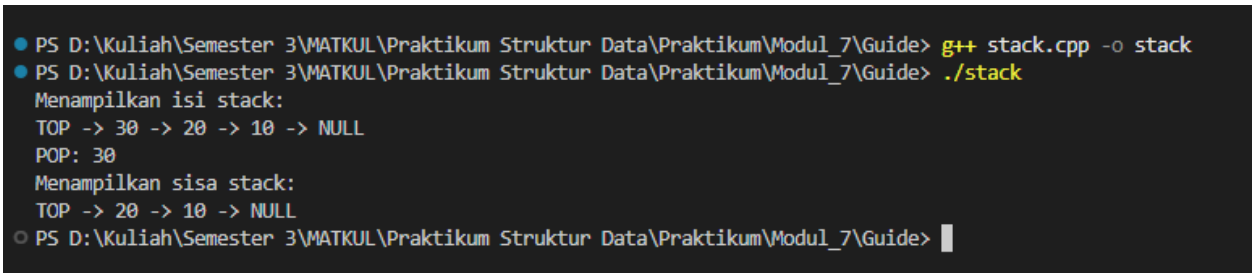
Guided 1

stack.cpp > main()

```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;
}
```

Screenshots Output



Deskripsi:

Program di atas merupakan implementasi struktur data stack menggunakan bahasa C++ dengan pendekatan linked list satu arah. Setiap elemen stack direpresentasikan oleh struct Node yang memiliki dua komponen, yaitu data untuk menyimpan nilai dan next sebagai pointer ke node berikutnya. Program ini menyediakan beberapa fungsi utama: push untuk menambahkan data ke bagian atas stack, pop untuk menghapus dan mengembalikan data teratas, isEmpty untuk memeriksa apakah stack kosong, serta show untuk menampilkan seluruh isi stack dari elemen teratas hingga terbawah. Dalam fungsi main, program membuat stack kosong, kemudian menambahkan tiga data yaitu 10, 20, dan 30 secara berurutan menggunakan push. Saat ditampilkan, hasilnya adalah “TOP -> 30 -> 20 -> 10 -> NULL” karena stack bersifat LIFO (Last In, First Out). Setelah dilakukan operasi pop, elemen teratas (30) dihapus, dan hasil tampilan stack menjadi “TOP -> 20 -> 10 -> NULL”, menunjukkan bahwa data terakhir yang dimasukkan akan menjadi data pertama yang keluar.

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

Unguided 1 (soal 1)

(file stack.h)

stack.h > ...

```
#ifndef STACK_H_INCLUDED
#define STACK_H_INCLUDED
#include <iostream>
using namespace std;

const int MAX = 20;
typedef int infotype;

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

Tabnine | Edit | Test | Explain | Document
void createStack(Stack &S);
Tabnine | Edit | Test | Explain | Document
bool isEmpty(Stack S);
Tabnine | Edit | Test | Explain | Document
bool isFull(Stack S);
Tabnine | Edit | Test | Explain | Document
void push(Stack &S, infotype x);
Tabnine | Edit | Test | Explain | Document
infotype pop(Stack &S);
Tabnine | Edit | Test | Explain | Document
void printInfo(Stack S);
Tabnine | Edit | Test | Explain | Document
void balikStack(Stack &S);
Tabnine | Edit | Test | Explain | Document
void pushAscending(Stack &S, infotype x);
Tabnine | Edit | Test | Explain | Document
void getInputStream(Stack &S);

#endif
```

(file doublylist.cpp)

stack.cpp > printInfo(Stack)

```cpp
#include "stack.h"

Tabnine | Edit | Test | Explain | Document
void createStack(Stack &S) {
    S.top = 0;
}

Tabnine | Edit | Test | Explain | Document
bool isEmpty(Stack S) {
    return S.top == 0;
}

Tabnine | Edit | Test | Explain | Document
bool isFull(Stack S) {
    return S.top == MAX;
}

Tabnine | Edit | Test | Explain | Document
void push(Stack &S, infotype x) {
    if (!isFull(S)) {
        S.info[S.top] = x;
        S.top++;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

Tabnine | Edit | Test | Explain | Document
infotype pop(Stack &S) {
    if (!isEmpty(S)) {
        S.top--;
        return S.info[S.top];
```

```
28      } else {
29          cout << "Stack kosong!" << endl;
30          return -1;
31      }
32  }
33
    Tabnine | Edit | Test | Explain | Document
34  void printInfo(Stack S) {
35      if (isEmpty(S)) {
36          cout << "[TOP] (kosong)" << endl;
37      } else {
38          cout << "[TOP] ";
39          for (int i = S.top - 1; i >= 0; i--) {
40              cout << S.info[i] << " ";
41          }
42          cout << endl;
43      }
44  }
45
    Tabnine | Edit | Test | Explain | Document
46  void balikStack(Stack &S) {
47      Stack temp;
48      createStack(temp);
49      while (!isEmpty(S)) {
50          push(temp, pop(S));
51      }
52      S = temp;
53  }
54
    Tabnine | Edit | Test | Explain | Document
55  void pushAscending(Stack &S, infotype x) {
56      Stack temp;
57      createStack(temp);
58      while (!isEmpty(S) && S.info[S.top - 1] > x) {
59          push(temp, pop(S));
60      }
61      push(S, x);
62      while (!isEmpty(temp)) {
63          push(S, pop(temp));
64      }
65  }
66
    Tabnine | Edit | Test | Explain | Document
67  void getInputStream(Stack &S) {
68      cout << "Masukkan angka (akhiri dengan Enter): " << endl;
69      char c;
70      while (true) {
71          c = cin.get();
72          if (c == '\n') break;
73          if (isdigit(c)) {
74              push(S, c - '0');
75          }
76      }
77  }
```

(file main.cpp)

main.cpp > main()

```cpp
#include "stack.h"

Tabnine | Edit | Test | Explain | Document
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);

    cout << "\nPush ascending:" << endl;
    cout << "Hello world!" << endl;
    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 << "\nInput stream:" << endl;
    createStack(S);
    getInputStream(S);
    printInfo(S);
    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

    return 0;
}
```

Screenshots Output

1) Output Stack

```
PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_7\Unguide> g++ main.cpp stack.cpp
PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_7\Unguide> ./a.exe
Hello world!
[TOP] 9 2 4 3
Balik stack:
[TOP] 3 4 2 9
```

2) Output stack push ascending

```
Push ascending:
Hello world!
[TOP] 9 8 4 3 3 2
Balik stack:
[TOP] 2 3 3 4 8 9
```

3) Output Stack

```
Input stream:
Masukkan angka (akhiri dengan Enter):
4729601
[TOP] 1 0 6 9 2 7 4
Balik stack:
[TOP] 4 7 2 9 6 0 1
```

Deskripsi:

Program di atas merupakan implementasi struktur data stack (tumpukan) menggunakan array statis dalam bahasa C++. Stack ini didefinisikan dalam stack.h dengan kapasitas maksimum 20 elemen, dan dilengkapi berbagai fungsi operasi seperti push untuk menambahkan data ke puncak stack, pop untuk menghapus data teratas, isEmpty dan isFull untuk memeriksa kondisi stack, serta printInfo untuk menampilkan isi stack dari atas ke bawah. Fungsi balikStack digunakan untuk membalik urutan elemen di dalam stack, sedangkan pushAscending memungkinkan penyisipan elemen secara terurut naik dengan bantuan stack sementara. Selain itu, fungsi getInputStream membaca input berupa deretan angka dari pengguna, lalu setiap digit dimasukkan ke dalam stack secara berurutan. Pada fungsi main, program mendemonstrasikan berbagai operasi stack seperti penambahan, penghapusan, pembalikan, penyusunan ascending, serta pembacaan input pengguna. Misalnya, ketika pengguna mengetik “1069274”, maka hasil tampilan stack adalah [TOP] 4 7 2 9 6 0 1, dan setelah dilakukan pembalikan dengan balikStack, urutannya menjadi [TOP] 1 0 6 9 2 7 4, menunjukkan bahwa program ini berhasil memanipulasi data sesuai prinsip LIFO (Last In, First Out).

D. Kesimpulan

Berdasarkan hasil praktikum pada Modul VII tentang Struktur Data Stack, dapat disimpulkan bahwa stack merupakan struktur data linier yang bekerja berdasarkan prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang diambil. Melalui praktikum ini, mahasiswa memahami cara kerja operasi dasar pada stack seperti push, pop, isEmpty, isFull, serta printInfo untuk menampilkan isi stack. Implementasi stack dalam bahasa C++ dapat dilakukan menggunakan array statis maupun linked list, masing-masing memiliki kelebihan dan kekurangan. Array memberikan kemudahan dalam alokasi memori tetap dan kecepatan akses, sedangkan linked list memberikan fleksibilitas tanpa batasan kapasitas.

Dalam bagian guided, mahasiswa mempelajari penerapan stack berbasis linked list untuk menambah dan menghapus data secara dinamis. Sedangkan pada bagian unguided, konsep tersebut dikembangkan menjadi program stack menggunakan array yang dilengkapi fitur tambahan seperti balikStack untuk membalik isi stack dan pushAscending untuk menyisipkan elemen secara terurut naik.

Sebagai contoh, ketika pengguna memasukkan input angka 1069274, maka isi stack akan ditampilkan sebagai: [TOP] 4 7 2 9 6 0 1 Setelah dilakukan pembalikan menggunakan fungsi balikStack, urutannya berubah menjadi: [TOP] 1 0 6 9 2 7 4 Contoh ini membuktikan bahwa operasi pada stack mengikuti prinsip LIFO dan fungsi tambahan seperti pembalikan dapat diimplementasikan dengan mudah melalui manipulasi tumpukan data. Secara keseluruhan, praktikum ini memperkuat pemahaman tentang penggunaan pointer, manipulasi array, dan logika LIFO dalam pengelolaan data. Struktur data stack memiliki peranan penting dalam berbagai aplikasi seperti evaluasi ekspresi matematika, manajemen memori, algoritma rekursif, serta fitur undo-redo pada perangkat lunak.

E. Referensi

Prasetyo, E., & Santosa, A. B. (2024). Penggunaan Struktur Data Stack dalam Implementasi Algoritma. Jurnal Teknologi Informatika dan Komputer, 5(2), 485-494. Tersedia secara terbuka di https://jurnal.stkippersada.ac.id/jurnal/index.php/jutech/article/download/4263/pdf

Trivusi. (2022). Struktur Data Stack: Pengertian, Karakteristik, dan Contoh Implementasi. Diakses dari https://www.trivusi.web.id/2022/07/struktur-data-stack.html