LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII
STACK

Disusun Oleh :

NAMA : HILMI HAKIM RAMADANI

NIM : 103112430016

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Stack adalah struktur data yang mengikuti prinsip LIFO (Last In, First Out), di mana elemen terakhir yang ditambahkan ke dalam stack akan menjadi yang pertama untuk dihapus atau diakses.

Dalam konteks pemrograman, operasi utama yang dilakukan pada stack adalah "push" (yang menambahkan elemen ke atas stack) dan "pop" (yang menghapus elemen teratas dari stack).

Implementasi stack biasanya menggunakan array atau struktur data linked list. Penggunaan array menyederhanakan penyimpanan karena memungkinkan indeksasi langsung. Sementara linked list memberikan fleksibilitas dalam penggunaan memori karena elemen dapat ditambahkan atau dihapus dengan mengubah pointer, tanpa perlu menggeser elemen lain dalam struktur data.

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

Guided

#Stack.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

```
menampilkan isi stack:
Top -> 30 -> 20 -> 10 -> NULL
pop: 30
menampilkan sisa stack:
Top -> 20 -> 10 -> NULL
PS D:\KULIAH\SEMESTER 3\Struktur Data>
```

Deskripsi:

Program ini merupakan implementasi struktur data Stack menggunakan Single Linked List dalam bahasa C++. Setiap elemen stack direpresentasikan sebagai node yang berisi data dan pointer ke node berikutnya, dengan pointer top sebagai penanda elemen paling atas. Program menyediakan operasi utama stack yaitu push untuk menambahkan data ke bagian atas stack, pop untuk menghapus dan mengambil data teratas, serta show untuk menampilkan isi stack dari atas ke bawah. Prinsip kerja stack yang diterapkan adalah LIFO (Last In First Out), yang ditunjukkan pada fungsi pop dimana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan.

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

Soal1

#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;
}
```

#stack.cpp

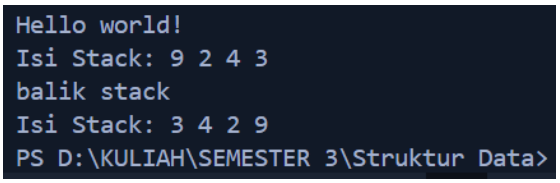
```
#include "stack.h"
#include <iostream>
using namespace std;

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

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

infotype Pop(Stack &S)
{
    if (S.top >= 0)
    {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    }
    else
    {
        cout << "Stack kosong!" << endl;
        return -1;
    }
}

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

```
    }
    else
    {
        cout << "Isi Stack: ";
        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)
    {
        int temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}
```

#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);

#endif
```

Screenshots Output



Soal 2

- Menambah code di stack.h

```
void pushAscending(Stack &S, infotype x);
```

- Menambah code di stack.cpp

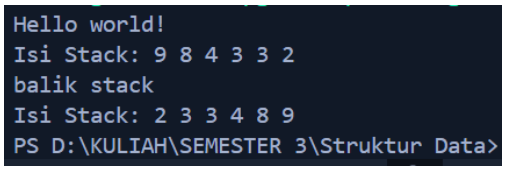
```
void pushAscending(Stack &S, infotype x)
{
   if (S.top == MAX - 1)
   {
      cout << "Stack penuh!" << endl;
      return;
   }

   int i = S.top;

   while (i >= 0 && S.info[i] > x)
   {
      S.info[i + 1] = S.info[i];
      i--;
   }

   S.info[i + 1] = x;
   S.top++;
}
```

- Menambah code di 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;
}
```

- Output



Soal 3

- Tambahkan code di stack.h

```
void getInputStream(Stack &S);
```

- Tambahkan code di stack.cpp

```
    void getInputStream(Stack &S)
    {
        char ch;
        cout << "Masukkan data: ";
        while ((ch = cin.get()) != '\n')
        {
            if (ch >= '0' && ch <= '9')
            {
                Push(S, ch - '0');
            }
        }
}
```

- Tambahkan code di 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;
}
```

- Output

```
Hello world!
Masukkan data: 4729601
[TOP] 1 0 6 9 2 7 4
balik stack
[TOP] 4 7 2 9 6 0 1
PS D:\KULIAH\SEMESTER 3\Struktur Data>
```

Deskripsi:

Program ini mengimplementasikan ADT Stack berbasis array dengan kapasitas maksimal 20 elemen. Stack menggunakan variabel top untuk menandai posisi elemen teratas. Operasi yang disediakan meliputi pembuatan stack, penambahan data (push), pengambilan data (pop), menampilkan isi stack, dan membalik urutan stack (balikStack). Prinsip kerja stack yang diterapkan adalah LIFO (Last In First Out). Prosedur getInputStream menggunakan cin.get() untuk membaca input karakter satu per satu hingga pengguna menekan Enter (\n), lalu setiap karakter dimasukkan ke stack. Dengan konsep LIFO, karakter terakhir yang dimasukkan akan berada di posisi paling atas stack.

D. Kesimpulan

Pada praktikum Modul Stack ini, telah berhasil diimplementasikan struktur data Stack menggunakan dua pendekatan, yaitu berbasis linked list dan berbasis array. Implementasi stack berbasis array menggunakan konsep ADT dengan komponen utama berupa array dan variabel top sebagai penanda elemen teratas, sehingga operasi push dan pop dapat

dilakukan secara efisien sesuai prinsip LIFO (Last In First Out). Selain operasi dasar, praktikum ini juga mengembangkan fitur tambahan seperti balikStack untuk membalik urutan elemen, pushAscending untuk memasukkan data secara terurut, serta getInputStream untuk membaca input karakter dari user hingga menekan tombol Enter. Melalui praktikum ini, dapat disimpulkan bahwa pemahaman terhadap konsep stack dan implementasinya sangat penting dalam pengelolaan data berurutan, serta membantu melatih ketelitian dalam pengaturan indeks, kapasitas array, dan alur logika program.

E. Referensi

RevoUpedia. 2025, 20 Desember. Stack. Diakses pada 20 Desember. Dari https://www.revou.co/kosakata/stack