LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 7
STACK

Disusun Oleh :

NAMA : LOH SUWARGI NITIS HAMENGKU BINTANG

NIM : 1031124001116

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Stack merupakan struktur data linear yang beroperasi berdasarkan prinsip *Last In, First Out* (LIFO), di mana elemen yang terakhir dimasukkan akan menjadi elemen pertama yang keluar. Implementasi stack biasanya dilakukan dengan dua cara: menggunakan array dan menggunakan *linked list*. Pendekatan berbasis array menawarkan kesederhanaan serta efisiensi dalam akses data berindeks dengan alokasi memori tetap, sehingga ideal untuk aplikasi yang memiliki batas jumlah elemen yang sudah ditentukan. Sebaliknya, implementasi dengan *linked list* memberikan fleksibilitas karena kapasitasnya dapat menyesuaikan kebutuhan saat program berjalan, meskipun membutuhkan pengelolaan memori tambahan dan sedikit beban dari penggunaan pointer.

Array menawarkan efisiensi dalam kecepatan dan struktur memori yang ringkas, sementara *linked list* unggul dalam skalabilitas dan fleksibilitas. Berbagai artikel akademik juga menekankan bahwa stack memiliki peranan penting dalam banyak algoritme dan fitur aplikasi, seperti *undo/redo*, evaluasi ekspresi, serta teknik *backtracking*. Dengan demikian, pemahaman mendalam terhadap kedua pendekatan ini menjadi esensial dalam pembelajaran dan pengembangan sistem berbasis struktur data.

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

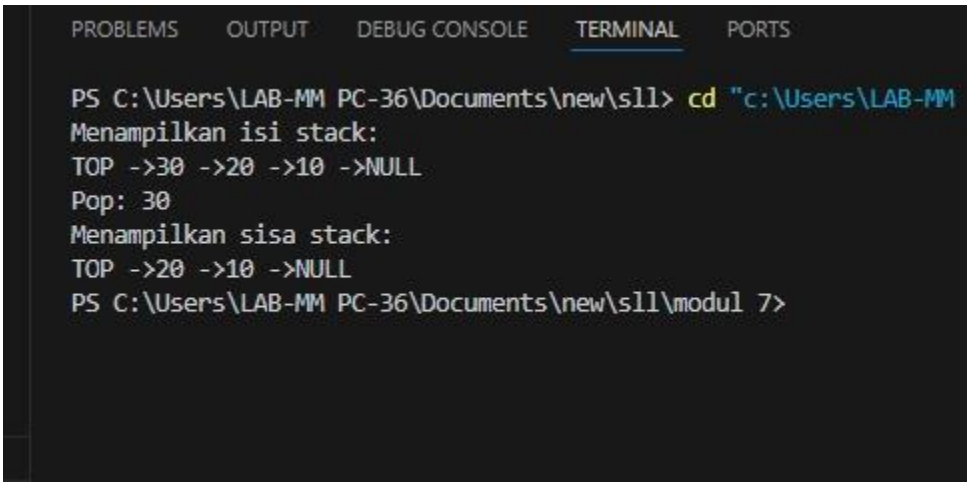
Guided 1

```
#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



Deskripsi: Program ini mengimplementasikan struktur data stack menggunakan pointer dengan pendekatan linked list. Program ini memiliki beberapa fungsi utama yaitu fungsi isEmpty() digunakan untuk memeriksa apakah stack kosong dengan memeriksa pointer top bernilai nullptr, yang kedua ada prosedur push() menambahkan elemen baru ke stack dengan cara membuat node baru kemudian menghubungkannya ke node sebelumnya melalui pointer next, lalu yang ketiga ada fungsi pop() berungsi untuk menghapus elemen paling atas stack dan mengembalikan nilainya. Prses ii dilakukan dengan memindahkan pointer tab ke node berikutnya dan menghapus node lama dari memori menggunakan delete.

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

Unguided 1

```
#ifndef STACK_H
#define STACK_H


const int maxEl = 20;


typedef int infotype;


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

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

#endif
```

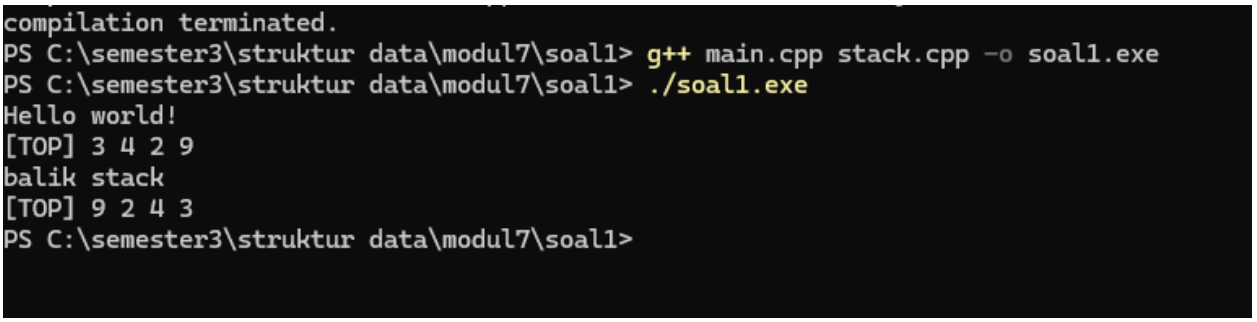
```
#include <iostream>
#include "stack.h"
using namespace std;

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

void push(Stack &S, infotype x) {
   if (S.top < maxEl - 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(const Stack &S) {
    cout << "[TOP] ";
    for (int i = 0; i <= S.top; i++) {
        cout << S.info[i] << " ";
    }
    cout << endl;
}

void balikStack(Stack &S) {
    Stack temp;
    createStack(temp);
    while (S.top >= 0) {
        push(temp, pop(S));
    }
    S = temp;
}
```

```
#include <iostream>
```

```
#include "stack.h"
using namespace std;

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

    Stack S;
    createStack(S);

    push(S, 9);
    push(S, 2);
    push(S, 4);
    push(S, 3);

    pop(S);
    push(S, 3);
    printInfo(S);

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

    return 0;
}
```

Screenshots Output



Deskripsi: Program ini dibuat untuk mengimplementasikan konsep Abstract Data Type (ADT) Stack menggunakan array sebagai representasi struktur datanya. Stack bekerja berdasarkan prinsip LIFO (Last In, First Out), artinya elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Struktur stack terdiri atas array info[20] dan variabel top yang menunjuk posisi elemen teratas. Program dibagi ke dalam tiga file utama, yaitu stack.h untuk deklarasi tipe data dan prosedur, stack.cpp untuk implementasi fungsi dasar seperti createStack(), push(), pop(), printInfo(), dan balikStack(), serta main.cpp untuk menjalankan pengujian program. Pada saat dijalankan, hasil yang ditampilkan adalah daftar elemen stack sebelum dan sesudah dibalik. Output yang benar menunjukkan urutan elemen awal [TOP] 9 2 4 3 kemudian setelah dibalik menjadi [TOP] 3 4 2 9.

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

Unguided 2

```
#ifndef STACK_H
#define STACK_H


const int maxEl = 20;
typedef int infotype;


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


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

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

#endif
```

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

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

void push(Stack &S, infotype x) {
    if (S.top < maxEl - 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 {
```

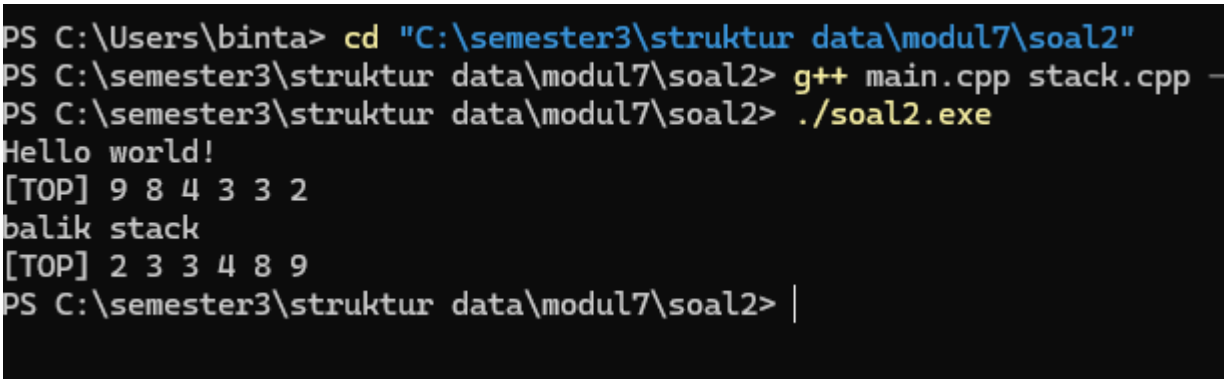
```
        cout << "Stack kosong!" << endl;
        return -1;
    }
}

void printInfo(const 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 >= 0) {
        push(temp, pop(S));
    }
    S = temp;
}

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

    while (S.top >= 0 && S.info[S.top] >= x){
        push(temp, pop(S));
    }
```

```
    push(S, x);

    while (temp.top >= 0) {
        push(S, pop(temp));
    }
}
```

```
#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;
}
```

Screenshots Output



Deskripsi: Mengembangkan program sebelumnya dengan menambahkan prosedur pushAscending() yang berfungsi untuk menyisipkan elemen baru ke dalam stack agar susunannya selalu terurut naik (ascending) dari bawah ke atas. Algoritma prosedur ini menggunakan satu stack tambahan sebagai penampung sementara. Saat elemen baru akan dimasukkan, program akan memindahkan semua elemen yang memiliki nilai lebih besar atau sama dengan nilai baru ke stack sementara. Setelah itu, elemen baru disisipkan ke posisi yang sesuai, kemudian seluruh elemen dari stack sementara dikembalikan ke stack utama. Cara ini memastikan bahwa data di dalam stack tetap terurut setiap kali terjadi penambahan elemen. Hasil akhir dari eksekusi program menampilkan urutan [TOP] 9 8 4 3 3 2 sebelum dibalik dan [TOP] 2 3 3 4 8 9 sesudah dibalik, yang menunjukkan bahwa data telah tersusun dengan benar secara ascending dan mendukung elemen bernilai sama.

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

Unguided 3

```
#ifndef STACK_H
#define STACK_H

const int maxEl = 20;

typedef int infotype;

struct Stack {
```

```
    infotype info[maxEl];
    int top;
};

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

#endif
```

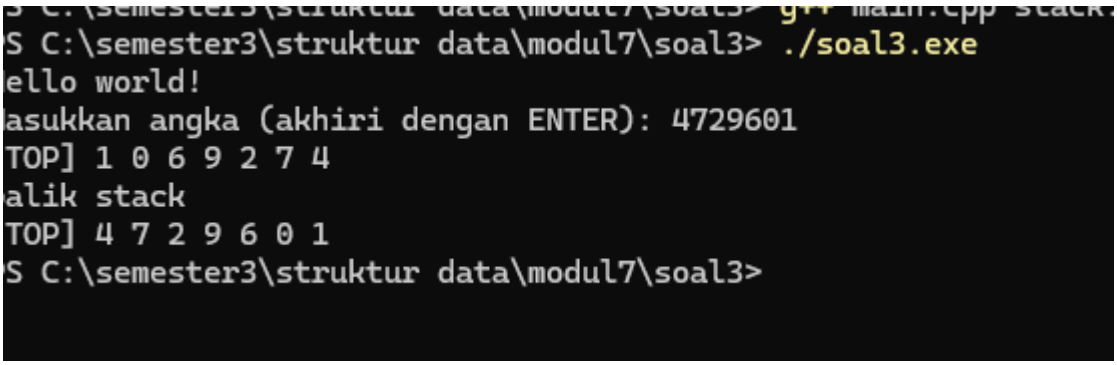
```
#include <iostream>
#include "stack.h"
using namespace std;

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

void push(Stack &S, infotype x) {
    if (S.top < maxEl - 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(const 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 >= 0) {
        push(temp, pop(S));
    }
    S = temp;
}
```

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

   while (S.top >= 0 && S.info[S.top] > x) {
      push(temp, pop(S));
   }
   push(S, x);
   while (temp.top >= 0) {
      push(S, pop(temp));
   }
}

void getInputStream(Stack &S) {
   char ch;
   cout << "Masukkan angka (akhiri dengan ENTER): ";
   while (true) {
      ch = cin.get();
      if (ch == '\n') break;
      if (ch >= '0' && ch <= '9') {
         int num = ch - '0';
         push(S, num);
      }
   }
}
```

```
#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;
}
```

Screenshots Output



Deskripsi: Program diupgrade lagi dengan menambahkan prosedur getInputStream() yang memungkinkan pengguna untuk memasukkan data langsung melalui keyboard tanpa menentukan jumlah elemen terlebih dahulu. Prosedur ini bekerja dengan membaca input karakter satu per satu

menggunakan fungsi cin.get() hingga pengguna menekan tombol Enter. Setiap karakter yang terbaca akan diubah menjadi angka dan langsung dimasukkan ke dalam stack menggunakan prosedur push(). Dengan cara ini, data yang diketik pengguna akan disimpan dalam urutan terbalik secara otomatis karena mengikuti prinsip LIFO. Setelah semua data masuk, program akan menampilkan isi stack sebelum dan sesudah dibalik. Contoh hasil eksekusi menunjukkan bahwa jika pengguna mengetik 4729601, maka output awalnya adalah [TOP] 1 0 6 9 2 7 4 dan setelah dibalik menjadi [TOP] 4 7 2 9 6 0 1. Hasil ini membuktikan bahwa prosedur getInputStream() telah berfungsi dengan benar untuk membaca input dan memanfaatkan konsep dasar stack.

G. Kesimpulan

Kesimpulannya Program ini menunjukkan bahwa dengan menggunakan pointer, operasi pada stack seperti push(), pop(), dan show() dapat dijalankan dengan fleksibilitas tinggi. Setiap kali data baru dimasukkan, node baru dibuat dan ditautkan ke node sebelumnya melalui pointer next, sedangkan operasi pop menghapus node teratas dengan memanfaatkan penghapusan dinamis. Hasil uji coba memperlihatkan bahwa elemen terakhir yang dimasukkan (30) menjadi elemen pertama yang keluar, membuktikan penerapan prinsip LIFO secara tepat pada struktur stack dinamis ini.

H. Referensi

Setiyawan, R. D., Hermawan, D., Abdillah, A. F., Mujayanah, A., & Vindua, R. (2024). PENGGUNAAN STRUKTUR DATA STACK DALAM PEMROGRAMAN C++ DENGAN PENDEKATAN ARRAY DAN LINKED LIST. ***JUTECH: Journal Education and Technology*****, *5*(2), 484-498.**

Amaylia, S., Setiabud, V. A., Alvianino, R., Saputra, R. N., Wardhani, H. K., & Suroni, A. (2025). Application of Stack Data Structure in Application Development. ***Journal of Advanced Systems Intelligence and Cybersecurity*****, *1*(01).**

Wijoyo, A., Azzahra, A., Nabila, D., Silviana, F., & Lusiyanti, L. (2024, May 30). ***Perbandingan struktur linked list dan array dalam manajemen memori*****. JRIIN : Jurnal Riset Informatika dan Inovasi, 1(12), 1290-1293.**