LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 7
STACK

Disusun Oleh :
NAMA : FANDIKA PRIMADANI
NIM : 103112400231

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Stack adalah struktur data linear yang menerapkan prinsip **LIFO (Last In, First Out)**, di mana elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Operasi dasarnya meliputi **push** (menambah data), **pop** (menghapus data teratas), dan **printInfo** (menampilkan isi stack).

B. SOAL 1

stack.h

```h
#ifndef STACK_H
#define STACK_H

const int 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
```

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 - 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) {
    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;
}
```

```
#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, 5);
    pop(S);
    pop(S);
    push(S, 2);
    push(S, 3);
    push(S, 9);

    printInfo(S);

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

    return 0;
}
```

SOAL 2

stack1.h

```
#ifndef STACK_H
#define STACK_H

const int 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);
void pushAscending(Stack &S, infotype x);

#endif
```

stack1.cpp

```
#include <iostream>
#include "stack1.h"
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) {
    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));
    }
}
```

main1.cpp

```
#include <iostream>
#include "stack1.h"
#include "stack1.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;
}
```

SOAL 3

stack2.h

```
#ifndef STACK_H
#define STACK_H

const int 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);
void pushAscending(Stack &S, infotype x);
void getInputStream(Stack &S);

#endif
```

stack2.cpp

```
#include <iostream>
#include "stack2.h"
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) {
    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) {
    cout << "Masukkan input (akhiri dengan ENTER): ";
    char c;

    while (true) {
        c = cin.get();
        if (c == '\n')
            break;

        int x = c - '0';
        if (x >= 0 && x <= 9) {
            push(S, x);
        }
    }
}
```

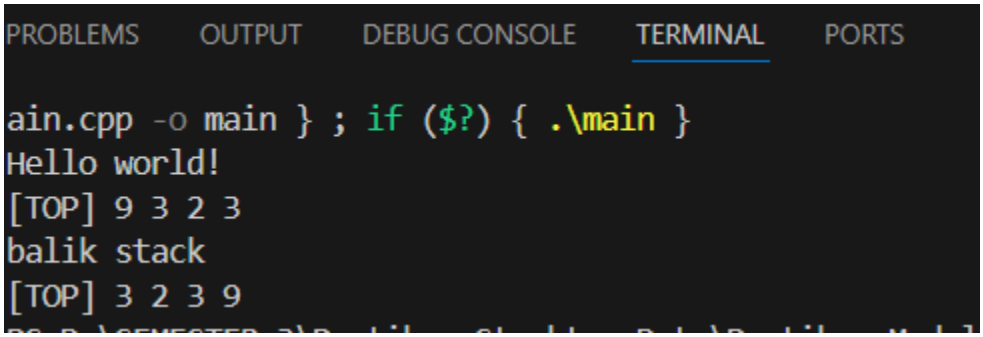
main2.cpp

```
#include <iostream>
#include "stack2.h"
#include "stack2.cpp"
using namespace std;

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

    Stack S;
    createStack(S);
```

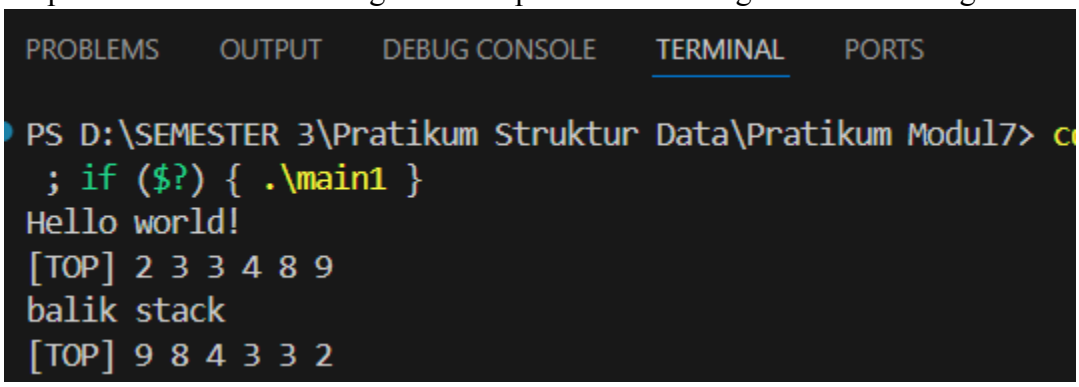
```
    getInputStream(S);
    printInfo(S);

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

    return 0;
}
```

Screenshots Output

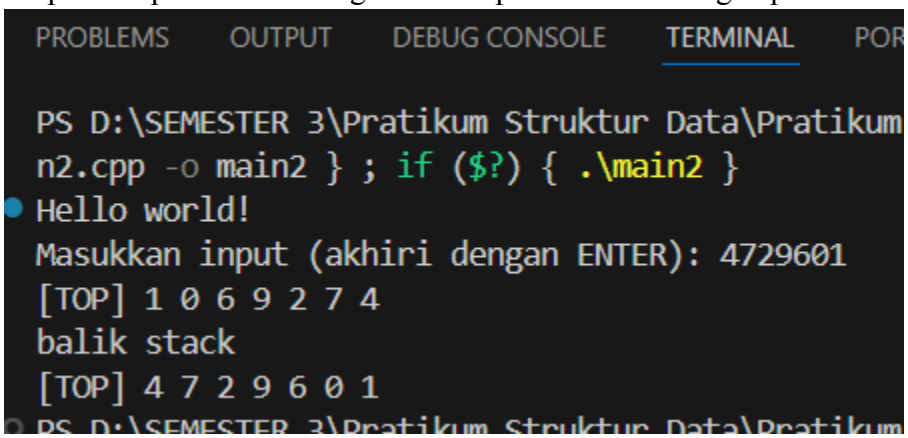
1. Output 1



2. Output 2 Carilah elemen dengan nomor polisi B001P dengan membuat fungsi baru.



3. Output 3 hapus elemen dengan nomor polisi B002P dengan procedure delete.



Deskripsi:

Soal 1 : Pada bagian pertama, program dibuat untuk mengimplementasikan **Abstract Data Type (ADT) Stack** menggunakan **array statis** sebagai media penyimpanan data.

Stack bekerja dengan prinsip **LIFO (Last In, First Out)**, yaitu data yang terakhir dimasukkan akan menjadi data yang pertama dikeluarkan.

Soal 2 : Pada bagian kedua, dikembangkan prosedur baru bernama **pushAscending()** yang berfungsi untuk menambahkan data ke dalam stack dengan **urutan menaik (ascending)**.

Artinya, setiap kali elemen baru dimasukkan, posisinya akan disesuaikan agar elemen pada bagian bawah memiliki nilai lebih kecil dibandingkan elemen di atasnya.

Proses ini dilakukan dengan menggunakan stack sementara untuk memindahkan elemen-elemen yang lebih kecil sebelum data baru dimasukkan, kemudian elemen-elemen tersebut dikembalikan ke stack utama. Hasilnya, isi stack akan selalu terurut naik dari bawah ke atas.

Soal 3 : Pada bagian ketiga, ditambahkan prosedur **getInputStream()** yang memungkinkan pengguna memasukkan data secara interaktif melalui **input keyboard**.
Program akan membaca setiap karakter yang diketik menggunakan fungsi cin.get() dan memasukkannya ke dalam stack hingga pengguna menekan tombol **Enter**.
Setiap karakter yang dimasukkan akan dikonversi menjadi bilangan integer sebelum disimpan.

C. Kesimpulan

Kesimpulannya, implementasi **ADT Stack menggunakan array** beserta penambahan prosedur **pushAscending()** dan **getInputStream()** menunjukkan bahwa struktur data stack mampu bekerja secara **efisien, terstruktur, dan fleksibel** dalam pengelolaan data. Stack tidak hanya berfungsi untuk menyimpan dan mengambil data berdasarkan prinsip **LIFO (Last In, First Out)**, tetapi juga dapat dikembangkan menjadi lebih **dinamis dan adaptif**, seperti mengatur elemen secara **menaik (ascending)** menggunakan stack tambahan serta menerima **input langsung dari pengguna** melalui keyboard. Dengan demikian, implementasi ini membuktikan bahwa stack merupakan salah satu struktur data penting yang mudah dimodifikasi sesuai kebutuhan pemrosesan dan pengolahan data dalam berbagai aplikasi.

D. Referensi

Drozdek, Adam. *Data Structures and Algorithms in C++*. Cengage Learning, 2012

Malik, D.S. *Data Structures Using C++*. Course Technology, 2010.