

LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN 7

STACK



Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

- Mampu memahami konsep stack pada struktur data dan algoritma
- Mampu mengimplementasikan operasi-operasi pada stack
- Mampu memecahkan permasalahan dengan solusi stack

II. TOOL

- Visual Studio Code
- GCC

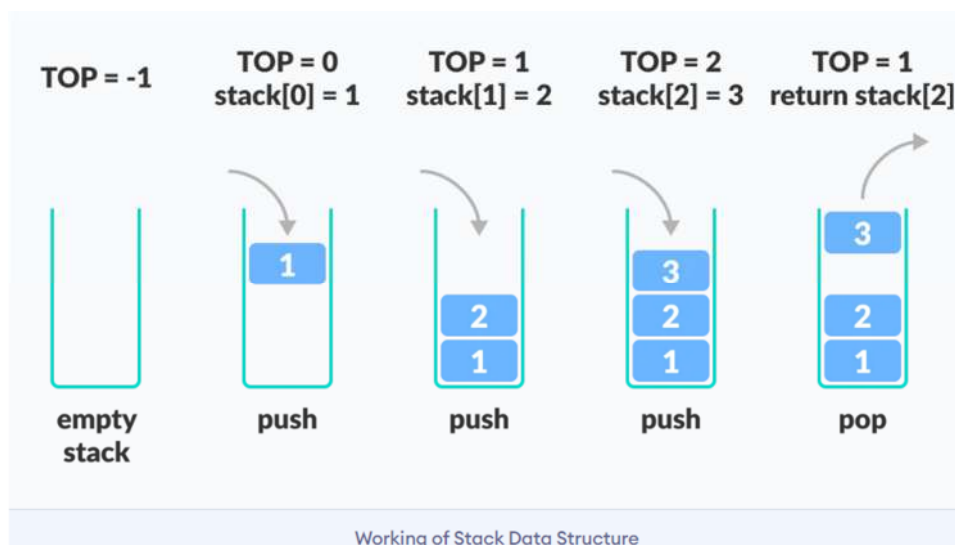
III. DASAR TEORI

Sebuah stack atau tumpukan merupakan struktur data yang berfungsi untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out (LIFO). Analogi yang sering digunakan adalah tumpukan piring di kafetaria, di mana piring terakhir yang ditambahkan akan menjadi yang pertama diambil.

Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat dalam berbagai aplikasi, termasuk manajemen memori komputer, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Sebagai contoh, dalam manajemen memori, stack digunakan untuk menyimpan alamat-alamat memori yang dialokasikan untuk variabel dan fungsi.

Dengan prinsip LIFO ini, stack memungkinkan akses data dengan efisiensi, di mana elemen yang terakhir dimasukkan akan menjadi yang pertama diambil. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak dan pemrograman komputer..



Operasi pada stack melibatkan beberapa fungsi dasar yang dapat dilakukan pada struktur data ini. Berikut adalah beberapa operasi umum pada stack:

- Push (Masukkan):** Menambahkan elemen ke dalam tumpukan pada posisi paling atas atau ujung.
- Pop (Keluarkan):** Menghapus elemen dari posisi paling atas atau ujung tumpukan.

- c. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya.
- d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.
- e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas).
- f. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan.
- g. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa menghapusnya.
- h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
- i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

IV. GUIDED

1. guided.cpp

```
#include <iostream>
#define MAX 100

using namespace std;

class Stack
{
    int top;
    int arr[MAX];

public:
    Stack() { top = -1; }

    bool isFull() { return top == MAX - 1; }
    bool isEmpty() { return top == -1; }

    void push(int x) {
        if (isFull())
        {
            cout << "Stack Overflow";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
        if (isEmpty())
        {
            cout << "Stack Underflow";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty())
        {
            return arr[top];
        }
    }
}
```

```

        cout << "Stack Underflow";
        return 0;
    }

    void display() {
        if (isEmpty())
        {
            cout << "Stack is Empty";
            return;
        }
        for (int i = top; i >= 0; i--){
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main()
{
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    cout << "Stack elements are: ";
    s.display();

    cout << "Top element is: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, the stack is ";
    s.display();

    return 0;
}

```

Output :

```

Stack elements are: 30 20 10
Top element is: 30
After popping, the stack is 10
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

2. guided2.cpp

```

#include <iostream>

using namespace std;

class Node
{
public:
    int data;
    Node *next;
    Node(int value)
    {

```

```

        data = value;
        next = NULL;
    }
};

class Stack {
private:
    Node *top;

public:
    Stack() { top = NULL; }
    bool isEmpty() { return top == NULL; }
    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }
    void pop() {
        if (isEmpty())
        {
            cout << "Stack Underflow\n";
            return;
        }
        Node* temp = top;
        top = top->next;
        delete temp;
    }
    int peek() {
        if (!isEmpty())
        {
            return top->data;
        }
        cout << "Stack Underflow\n";
        return -1;
    }

    void display() {
        if (isEmpty())
        {
            cout << "Stack is Empty\n";
            return;
        }
        Node* current = top;
        while (current){
            cout << current->data << " ";
            current = current->next;
        }

        cout << "\n";
    }
};

int main()
{
    Stack s;
    s.push(10);

```

```

    s.push(20);
    s.push(30);
    cout << "Stack elements are: ";
    s.display();

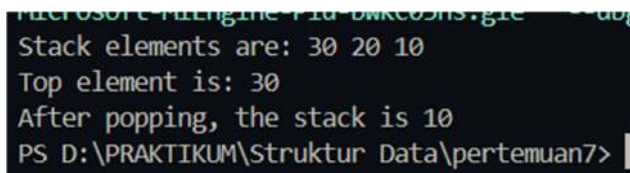
    cout << "Top element is: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, the stack is ";
    s.display();

    return 0;
}

```

Output :



```

Stack elements are: 30 20 10
Top element is: 30
After popping, the stack is 10
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

V. UNGUIDED

1. stack.h

```

#ifndef STACK_H
#define STACK_H

// Define the infotype as an integer
typedef int infotype;

// Define the Stack structure
typedef struct {
    infotype info[20]; // Array to store stack elements
    int top;           // Index of the top element
} Stack;

// Procedure prototypes
void createStack(Stack *S);
void push(Stack *S, infotype x);
void pushAscending(Stack *S, infotype x);
void getInputStream(Stack *S);
infotype pop(Stack *S);
void printInfo(const Stack *S);
void balikStack(Stack *S);

#endif // STACK_H

```

2. stack.cpp

```

#include "stack.h"
#include <iostream>
#include <sstream>

using namespace std;

```

```

/**
 * @brief Initializes the stack by setting the top index to -1.
 *
 * @param S Pointer to the stack to be initialized.
 */
void createStack(Stack *S){
    S->top = -1;
}

/**
 * @brief Pushes an element onto the stack.
 *
 * This function adds an element to the top of the stack. If the stack is full,
 * it prints an error message indicating a stack overflow.
 *
 * @param S Pointer to the stack.
 * @param x The element to be pushed onto the stack.
 */
void push(Stack *S, infotype x){
    if (S->top == 19){
        cerr << "Error : Stack Overflow\n";
        return;
    }
    S->top++;
    S->info[S->top] = x;
}

/**
 * @brief Removes and returns the top element from the stack.
 *
 * This function checks if the stack is empty. If it is, an error message is
 * printed to the standard error stream and the function returns -1.
 * Otherwise,
 * it retrieves the top element from the stack, decrements the top index, and
 * returns the retrieved element.
 *
 * @param S Pointer to the stack from which the top element is to be removed.
 * @return infotype The top element of the stack if the stack is not empty,
 * otherwise -1.
 */
infotype pop(Stack *S){
    if (S->top == -1){
        cerr << "Error : Stack Underflow\n";
        return -1;
    }
    infotype x = S->info[S->top];
    S->top--;
    return x;
}

/**
 * @brief Prints the elements of the stack from top to bottom.
 *
 * This function iterates through the stack from the top element to the bottom
 * and prints each element to the standard output.

```

```

*
* @param S Pointer to the stack to be printed.
*/
void printInfo(const Stack *S){
    cout << "[TOP] : ";
    if (S->top == -1){
        cout << "Stack is Empty\n";
        return;
    } else {
        for (int i = S->top; i >= 0; i--){
            cout << S->info[i] << " ";
        }
        cout << "\n";
    }
}

/**
* @brief Reverses the elements of the given stack.
*
* This function takes a pointer to a stack and reverses the order of its
elements.
* It uses a temporary stack to hold the elements in reverse order and then
assigns
* the reversed stack back to the original stack.
*
* @param S Pointer to the stack to be reversed.
*/
void balikStack(Stack *S){
    Stack temp; // Temporary stack to hold reversed elements
    createStack(&temp); // Initialize the temporary stack
    while (S->top != -1){
        push(&temp, pop(S)); // Pop from S and push to temp, so the elements
are reversed
    }
    *S = temp; // Assign the reversed stack back to S
}

/**
* @brief Pushes an element into the stack in ascending order.
*
* This function inserts an element into the stack while maintaining the
stack's
* elements in ascending order. If the stack is empty, the element is simply
pushed
* onto the stack. Otherwise, the function temporarily pops elements from the
stack
* until it finds the correct position for the new element, then pushes the
new element
* and restores the temporarily removed elements back onto the stack.
*
* @param S Pointer to the stack.
* @param x The element to be pushed into the stack.
*/
void pushAscending(Stack *S, infotype x){
    if (S->top == -1){ // If the stack is empty, simply push the element

```



```

        push(S, x);
        return;
    }
    Stack temp; // Temporary stack to hold elements temporarily removed from S
    createStack(&temp); // Initialize the temporary stack
    while (S->top != -1 && S->info[S->top] < x){ // Find the correct position
for x
        push(&temp, pop(S)); // Pop elements from S and push to temp until the
correct position is found
    }
    push(S, x); // Push x to S
    while (temp.top != -1){
        push(S, pop(&temp)); // Restore the temporarily removed elements back
to S
    }
}

/**
 * @brief Reads a stream of characters from the standard input and pushes
digits onto the stack.
 *
 * This function continuously reads characters from the standard input until
the Enter key is pressed.
 * If a digit character is encountered, it is converted to an integer and
pushed onto the provided stack.
 *
 * @param S A pointer to the Stack where the digits will be pushed.
 */
void getInputStream(Stack *S) {
    char input;
    cout << "Masukan angka (press Enter to stop):\n";
    while (true) {
        input = cin.get();
        if (input == '\n') {
            break;
        }
        if (isdigit(input)) {
            push(S, input - '0'); // Convert char to int and push onto the
stack
        }
    }
}

```

3. main1.cpp

```

#include "stack.cpp"
#include <iostream>

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;
}

```

Output :

```

Hello world!
[TOP] : 9 2 4 3
balik stack
[TOP] : 3 4 2 9
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

4. main2.cpp

```

#include "stack.cpp"
#include <iostream>

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;
}

```

Output :

```

Microsoft Windows [Version 10.0.17134.0] (c) 2018 Microsoft Corporation. All rights reserved.
Hello world!
[TOP] : 2 3 3 4 8 9
balik stack
[TOP] : 9 8 4 3 3 2
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

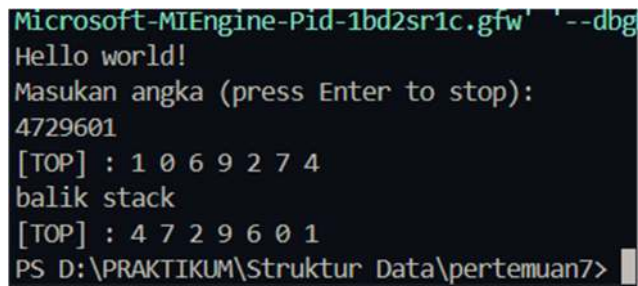
5. main3.cpp

```
#include <iostream>
#include "stack.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;
}
```

Output :



```
Microsoft-MIEngine-Pid-1bd2sr1c.gfw' '--dbg
Hello world!
Masukan angka (press Enter to stop):
4729601
[TOP] : 1 0 6 9 2 7 4
balik stack
[TOP] : 4 7 2 9 6 0 1
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>
```

VI. KESIMPULAN

Praktikum ini membahas materi mengenai struktur data Stack, meliputi penjelasan konsep dasar, metode, dan prosedur yang terkait seperti push, pop, dan peek, serta penerapannya dalam pemrograman C++. Stack, sebagai struktur data LIFO (Last In, First Out), digunakan untuk menyimpan elemen di mana elemen terakhir yang dimasukkan adalah elemen pertama yang dikeluarkan. Implementasi dalam C++ mencakup pembuatan program dengan fungsi-fungsi untuk menambah elemen (push), menghapus elemen (pop), dan melihat elemen puncak (peek). Praktikum ini menunjukkan bagaimana struktur data Stack dapat digunakan dalam berbagai aplikasi, seperti pemrosesan undo/redo, evaluasi ekspresi matematika, dan simulasi panggilan fungsi. Hasil dari praktikum ini membantu memahami penerapan Stack secara efektif dalam penyelesaian masalah komputasi.