

LAPORAN PRAKTIKUM

Modul 7

Stack



Disusun Oleh :

Arzario Irsyad Al Fatih/2211104032

SE 07 2

Asisten Praktikum :

Aldi Putra

Andini Nur Hidayah

Dosen Pengampu :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. Tujuan

- Memahami konsep stack oada struktur data dan algoritma.
- Mampu mengimplementasikan operasi-operasi pada stack.
- Mampu memecahkan permasalahan dengan solusi stack.

2. Landasan Teori

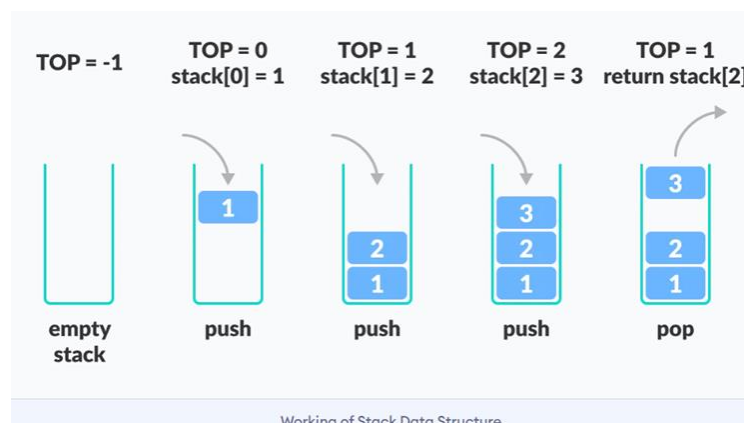
a. Stack

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

3. Guided

a. Guided 1

Source Code

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

using namespace std;

class Stack { // Corrected class name capitalization
private:
    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\n";
        return;
    }
    arr[++top] = x;
}

void pop() {
    if (isEmpty()) {
        cout << "Stack underflow\n";
        return;
    }
    top--; // Fixed missing semicolon
}

int peek() {
    if (!isEmpty()) {
        return arr[top];
    }
    cout << "Stack is empty\n";
    return -1;
}

void display() {
    if (isEmpty()) {
        cout << "Stack is empty\n";
        return;
    }
    for (int i = top; i >= 0; i--) { // Fixed for-loop syntax
        cout << arr[i] << " ";
    }
    cout << "\n"; // Fixed missing semicolon
}

};

int main() {
    Stack s;

```

```

s.push(10);
s.push(20);
s.push(30);

cout << "Stack elements: ";
s.display();

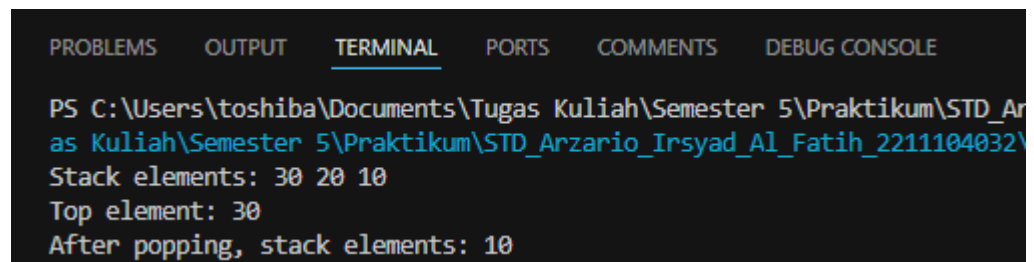
cout << "Top element: " << s.peek() << "\n";

s.pop();
s.pop();
cout << "After popping, stack elements: ";
s.display();

return 0;
}

```

Output



```

PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_Ar
as Kuliah\Semester 5\Praktikum\STD_Arzario_Irsyad_Al_Fatih_2211104032\
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10

```

Deskripsi

Program ini mendemonstrasikan implementasi struktur data stack menggunakan array dengan kapasitas maksimum 100 elemen, dilengkapi dengan metode untuk operasi dasar seperti push (menambahkan elemen ke tumpukan), pop (menghapus elemen dari puncak tumpukan), dan peek (melihat elemen di puncak tumpukan). Program juga menyediakan fungsi untuk memeriksa apakah stack kosong (isEmpty) atau penuh (isFull), serta menampilkan seluruh elemen dalam stack (display). Pada fungsi main, program menambahkan beberapa elemen ke stack, menampilkan isinya, melihat elemen puncak, dan menghapus elemen sambil menunjukkan perubahan isi stack. Program ini juga menangani kondisi overflow dan underflow dengan pesan kesalahan yang sesuai.

b. Guided 2

Source Code

```
#include <iostream>

using namespace std;

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

class Stack
{
private:
    Node *top;

public:
    Stack()
    {
        top = nullptr;
    }

    bool isEmpty()
    {
        return top == nullptr;
    }

    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())
        {
            cout << "Stack is empty\n";
            return -1;
        }
        return top->data;
    }

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

```

```

};

int main()
{
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

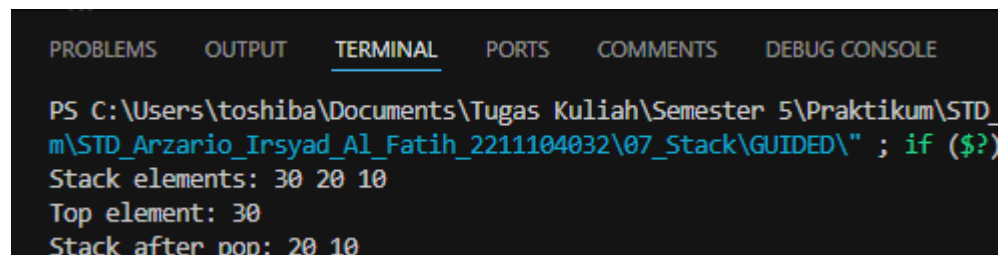
    cout << "Top element: " << s.peak() << "\n";

    s.pop();
    cout << "Stack after pop: ";
    s.display();

    return 0;
}

```

Output



```

PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_
m\STD_Arzario_Irsyad_Al_Fatih_2211104032\07_Stack\GUIDED\> if ($?)
Stack elements: 30 20 10
Top element: 30
Stack after pop: 20 10

```

Deskripsi

Program ini mengimplementasikan struktur data stack menggunakan pendekatan berbasis linked lis. Program mendefinisikan kelas `Node` untuk merepresentasikan setiap elemen dalam stack, dengan atribut `data` untuk menyimpan nilai dan `next` sebagai penunjuk ke elemen berikutnya. Kelas `Stack` menyediakan metode operasi dasar seperti push (menambahkan elemen ke puncak stack), pop (menghapus elemen dari puncak stack), peek (mengakses elemen teratas), dan display (menampilkan seluruh elemen stack). Metode isEmpty digunakan untuk memeriksa apakah stack kosong. Pada

fungsi `main`, beberapa elemen ditambahkan ke stack, ditampilkan, elemen puncak diakses, kemudian satu elemen dihapus sambil menunjukkan perubahan isi stack. Implementasi ini menangani situasi stack kosong dengan memberikan pesan kesalahan yang sesuai.

4. Unguided

a. Soal 1

Source Code

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

bool isPalindrome(string str) {
    stack<char> s;
    string filteredStr = "";

    for (char c : str) {
        if (isalnum(c)) {
            filteredStr += tolower(c);
            s.push(tolower(c));
        }
    }

    for (char c : filteredStr) {
        if (s.top() != c) {
            return false;
        }
        s.pop();
    }

    return true;
}

int main() {
    string input;
    cout << "Masukkan kalimat: ";
    getline(cin, input);
```

```

    if (isPalindrome(input)) {
        cout << "Kalimat tersebut adalah palindrom.\n";
    } else {
        cout << "Kalimat tersebut bukan palindrom.\n";
    }

    return 0;
}

```

Output

```

PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_
Arzario_Irsyad_Al_Fatih_2211104032\07_Stack\UNGUIDED\" ; if ($?) { g-
Masukkan kalimat: Ini
Kalimat tersebut adalah palindrom.

PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_
kum\STD_Arzario_Irsyad_Al_Fatih_2211104032\07_Stack\UNGUIDED\" ; if
Masukkan kalimat: Telkom
Kalimat tersebut bukan palindrom.

```

Deskripsi

Program ini memeriksa apakah sebuah kalimat merupakan palindrom dengan memanfaatkan stack. Kalimat difilter untuk hanya menyertakan huruf dan angka, lalu dikonversi menjadi huruf kecil agar tidak *case-sensitive*. Setiap karakter yang valid dimasukkan ke dalam stack, kemudian dibandingkan dengan urutan asli kalimat. Proses pengecekan dilakukan dengan mengeluarkan karakter satu per satu dari stack untuk memastikan urutannya sama dari depan dan belakang. Jika ada ketidakcocokan, kalimat tersebut bukan palindrom.

b. Soal 2

Source Code

```

#include <iostream>
#include <stack>
#include <string>
using namespace std;

void reverseCharacters(string sentence) {

```

```

stack<char> charStack;

for (char c : sentence) {
    charStack.push(c);
}

cout << "Kalimat setelah dibalik: ";
while (!charStack.empty()) {
    cout << charStack.top();
    charStack.pop();
}
cout << endl;
}

int main() {
    string input;
    cout << "Masukkan kalimat: ";
    getline(cin, input);

    reverseCharacters(input);

    return 0;
}

```

Output

```

PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_A
rzario_Irsyad_Al_Fatih_2211104032\07_Stack\UNGUIDED\" ; if ($?) { g+
Masukkan kalimat: Telkom Purwokerto
Kalimat setelah dibalik: otrekowruP mokleT

```

Deskripsi

Program ini membalikkan huruf dalam sebuah kalimat dengan memanfaatkan stack. Setiap karakter dalam kalimat, termasuk spasi dan tanda baca, dimasukkan ke dalam stack. Setelah itu, karakter dikeluarkan dari stack satu per satu sehingga menghasilkan urutan huruf yang terbalik dari kalimat asli. Proses ini menggunakan prinsip Last-In, First-Out (LIFO) dari stack untuk membalikkan urutan karakter secara efisien.