LAPORAN PRAKTIKUM
MODUL 7
STACK

Disusun Oleh:

Satria Putra Dharma Prayudha - 21104036

SE07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. Tujuan

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

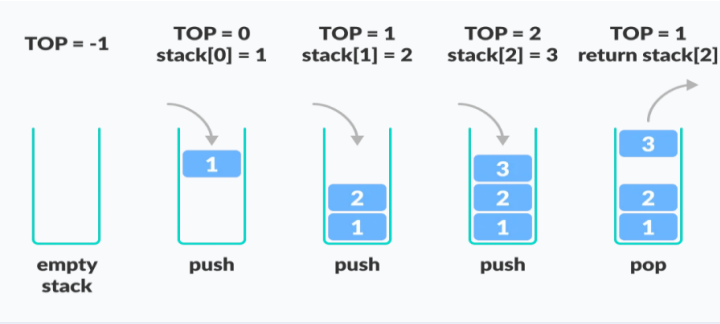
B. Landasan Teori

Landasan teori ini berdasarkan pada modul pembelajaran praktikum struktur data kali ini

2.1 Stack

Stack (tumpukan) adalah salah satu struktur data yang menggunakan prinsip *Last-In, First-Out* (LIFO). Ini berarti elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Analogi yang sering digunakan untuk menggambarkan stack adalah tumpukan piring: piring terakhir yang ditumpuk di atas akan menjadi yang pertama diambil.

Stack memiliki peran penting dalam berbagai aplikasi komputasi, seperti manajemen memori, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Misalnya, dalam manajemen memori, stack digunakan untuk menyimpan alamat memori yang dialokasikan untuk variabel dan fungsi. Dengan prinsip LIFO, stack memungkinkan akses data secara efisien, di mana elemen terakhir yang dimasukkan menjadi yang pertama diakses. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak.



Working of Stack Data Structure

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

- **Push**: Menambahkan elemen baru ke bagian paling atas stack.
- **Pop**: Menghapus elemen teratas dari stack.
- **Top/Peek**: Melihat elemen teratas tanpa menghapusnya.
- **IsEmpty**: Mengecek apakah stack kosong.
- **IsFull**: Mengecek apakah stack sudah penuh (terutama pada implementasi berbasis array yang memiliki batas ukuran).
- **Size**: Mengembalikan jumlah elemen dalam stack.
- **Clear**: Mengosongkan semua elemen dalam stack.
- **Search**: Mencari elemen tertentu dalam stack.

Dalam implementasinya, stack dapat dibangun dengan berbagai metode. Yang paling umum adalah menggunakan **array** atau **linked list**. Pada implementasi berbasis array, ukuran stack biasanya dibatasi oleh kapasitas array, sehingga perlu memperhatikan pengecekan kondisi penuh. Sedangkan pada stack berbasis linked list, tidak ada batasan ukuran kecuali keterbatasan memori, sehingga lebih fleksibel.

C. Guided

a. Stack Dengan Array

Code :

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

using namespace std;

class Stack {
    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--;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[top];
        }
        cout << "Stack is Empty\n";
        return -1; // Return a sentinel value
    }

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

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

    cout << "Stack elements: ";
    s.display();
    cout << "Top element is: " << s.peek() << endl;

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

    return 0;
}
```

Output :

```
Stack elements: 30 20 10
Top element is: 30
After popping two elements: 10
```

Penjelasan :

Stack diimplementasikan menggunakan array dengan ukuran maksimal 100 elemen. Fungsi utama yang disediakan adalah push() untuk menambah elemen, pop() untuk menghapus elemen teratas, peek() untuk melihat elemen teratas, dan display() untuk menampilkan seluruh elemen stack. Ketika program dijalankan, tiga elemen (10, 20, dan 30) dimasukkan, dan outputnya adalah "30 20 10" mengikuti prinsip LIFO. Setelah dua elemen dihapus, yang tersisa adalah elemen 10, yang ditampilkan oleh fungsi display(). Stack ini memiliki batasan jumlah elemen karena menggunakan array.

b. Stack Dengan Linked Link List

Code :

```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 is Empty\n";
        return -1; // Return a sentinel value
    }

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

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

    cout << "Stack elements: ";
    s.display();
    cout << "Top element is: " << s.peek() << endl;

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

    return 0;
    }
```

Output :

```
Stack elements: 30 20 10
Top element is: 30
After popping two elements: 20 10
```

Penjelasan :

Stack diimplementasikan menggunakan **linked list**, dengan setiap node menyimpan data dan menunjuk ke node berikutnya. Fungsi push() menambahkan node baru ke puncak stack, pop() menghapus node teratas, peek() melihat nilai teratas, dan display() menampilkan elemen-elemen stack. Outputnya serupa dengan Code 1, dengan elemen yang dimasukkan adalah 10, 20, dan 30, yang ditampilkan dalam urutan "30 20 10". Setelah satu elemen dihapus, sisa elemen yang ditampilkan adalah "20 10". Stack ini lebih fleksibel karena menggunakan alokasi memori dinamis, berbeda dengan array pada kode yang menggunakan array.

D. Unguided

a. Palindrom

Code:

```cpp
#include <iostream>
#include <stack>
#include <cctype>
using namespace std;

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

    // Memfilter karakter, menghapus spasi dan menjadikan huruf kecil
    for (char c : str) {
        if (isalpha(c)) {
            filteredStr += tolower(c);
        }
    }

    // Menambahkan karakter ke stack
    for (char c : filteredStr) {
        s.push(c);
    }

    // Memeriksa apakah string sama saat dibaca dari depan dan belakang
    for (char c : filteredStr) {
        if (c != s.top()) {
            return false;
        }
        s.pop();
    }

    return true;
}

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

    if (isPalindrome(str)) {
        cout << "Kalimat tersebut adalah palindrom." << endl;
    } else {
        cout << "Kalimat tersebut bukan palindrom." << endl;
    }

    return 0;
}
```

Output:

```
Masukkan kalimat: Sisat
Kalimat tersebut bukan palindrom.
```

```
Masukkan kalimat: Kasur ini rusak
Kalimat tersebut adalah palindrom.
```

Penjelasan:

Program ini digunakan untuk menentukan apakah sebuah kalimat adalah palindrom, yaitu kalimat yang terbaca sama dari depan maupun dari belakang. Pada prosesnya, program menggunakan stack untuk menyimpan karakter dari kalimat yang telah difilter agar hanya menyisakan karakter huruf dalam bentuk huruf kecil.

Penjelasan Langkah Program:

1. **Pembersihan dan Normalisasi Input:** Program memfilter karakter dalam kalimat masukan untuk hanya menyertakan huruf dengan menghilangkan spasi atau karakter lain, serta mengonversinya menjadi huruf kecil.
2. **Pengisian Stack:** Setiap karakter yang telah difilter kemudian dimasukkan ke dalam stack. Stack digunakan karena sifatnya LIFO (Last In, First Out) yang memungkinkan pemeriksaan palindrom secara efisien.
3. **Pengecekan Karakter:** Program memeriksa apakah string hasil filter memiliki karakter yang sama saat dibaca dari depan dan dari belakang dengan cara membandingkan setiap karakter dalam filteredStr dengan karakter yang diambil dari stack. Jika ada karakter yang berbeda, maka kalimat tersebut bukan palindrom.
4. **Hasil Akhir:** Jika seluruh karakter cocok, kalimat tersebut dianggap palindrom dan hasil "Kalimat tersebut adalah palindrom." akan ditampilkan. Jika tidak, hasil "Kalimat tersebut bukan palindrom." akan ditampilkan.

b. Membalik Kata

Code:

```cpp
#include <iostream>
#include <stack>
#include <sstream>
using namespace std;

void reverseWords(string sentence) {
    stack<string> wordStack;
    stringstream ss(sentence);
    string word;

    // Memasukkan setiap kata ke dalam stack
    while (ss >> word) {
        wordStack.push(word);
    }

    // Mengeluarkan kata-kata dari stack dan membalikkan urutan setiap kata
    cout << "Hasil : ";
    while (!wordStack.empty()) {
        string reversedWord = wordStack.top();
        wordStack.pop();

        // Membalikkan setiap kata
        for (int i = 0, j = reversedWord.length() - 1; i < j; i++, j--) {
            swap(reversedWord[i], reversedWord[j]);
        }

        cout << reversedWord << " ";
    }
    cout << endl;
}

int main() {
    string sentence;
    cout << "Masukkan kalimat (minimal 3 kata): ";
    getline(cin, sentence);

    reverseWords(sentence);

    return 0;
}
```

Output:

```
Masukkan kalimat (minimal 3 kata): Satria Putra
Hasil : artuP airtaS
```

Penjelasan:

Program ini digunakan untuk membalikkan setiap kata dalam kalimat yang dimasukkan oleh pengguna. Namun, urutan kata tetap dipertahankan, hanya urutan karakter dalam setiap kata yang dibalik.

Penjelasan Langkah Program:

1. **Pemecahan Kalimat menjadi Kata-Kata:** Program memanfaatkan stringstream untuk memisahkan kalimat menjadi kata-kata berdasarkan spasi. Setiap kata yang berhasil dipisahkan dimasukkan ke dalam stack wordStack.
2. **Pembalikan Urutan Karakter dalam Setiap Kata:** Setelah semua kata masuk ke stack, program mengeluarkan satu per satu kata dari stack (dimulai dari kata terakhir yang dimasukkan karena sifat LIFO) dan kemudian membalikkan urutan karakter dalam kata tersebut. Pembalikan dilakukan dengan menukar karakter menggunakan dua indeks, yaitu indeks awal dan akhir yang bergerak menuju tengah.
3. **Penampilan Hasil:** Program menampilkan setiap kata yang telah dibalik urutannya di layar dengan menggunakan spasi untuk memisahkan antar kata. Hasil akhir adalah kalimat yang memiliki urutan kata yang sama dengan kalimat asli, tetapi setiap kata dalam kalimat tersebut dibalik urutan karakternya.

E. Kesimpulan

Dalam praktikum ini, kita telah mempelajari konsep dasar dan penerapan stack dalam struktur data. Stack adalah struktur data yang mengikuti prinsip Last-In-First-Out (LIFO), yang memungkinkan elemen terakhir yang dimasukkan menjadi elemen pertama yang dikeluarkan. Implementasi stack dapat dilakukan menggunakan array atau linked list, masing-masing memiliki kelebihan dan kekurangan, seperti keterbatasan ukuran pada array dan fleksibilitas memori pada linked list. Selain itu, penggunaan stack juga dapat diterapkan dalam berbagai permasalahan komputasi, seperti memeriksa palindrom pada kalimat dan membalikkan urutan kata dalam sebuah kalimat. Melalui praktikum ini, kita dapat memahami bagaimana stack digunakan untuk memecahkan masalah-masalah tersebut secara efisien dan fleksibel. Dengan memanfaatkan operasi dasar stack, seperti push, pop, dan peek, kita dapat melakukan manipulasi data yang lebih kompleks dengan mudah.