LAPORAN PRAKTIKUM

PERTEMUAN 7

07_STACK

Nama :

Ilham Lii Assidaq (2311104068)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Memahami konsep stack.

Mengimplementasikan stack dengan menggunakan representasi pointer dan tabel.

Memahami konsep queue.

II. TOOL

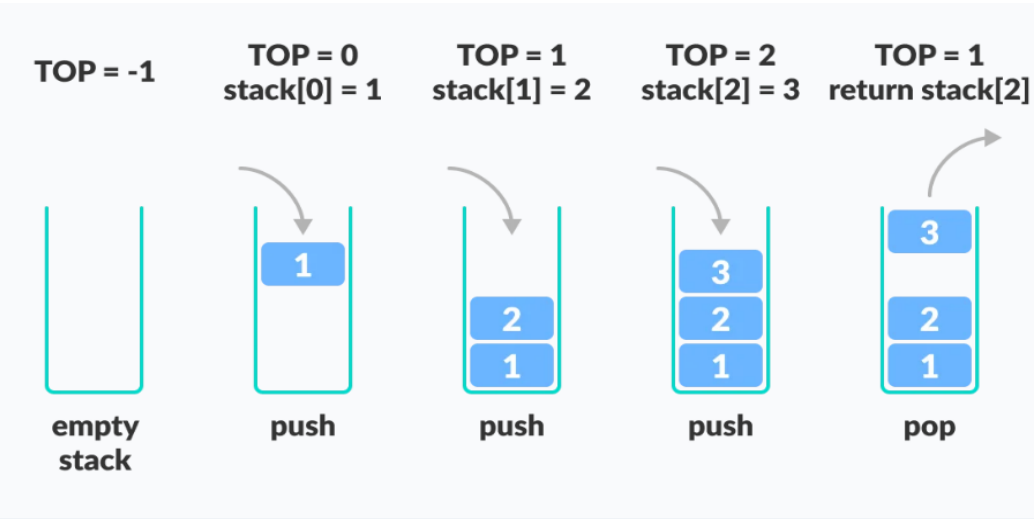
Dev C++

III. DASAR TEORI

Stack merupakan salah satu bentuk struktur data dimana prinsip operasi yang digunakan seperti tumpukan. Seperti halnya tumpukan, elemen yang bisa diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang pertama kali masuk, prinsip ini biasa disebut LIFO (Last In First Out).

LIFO (Last In, First Out) adalah prinsip dasar dalam struktur data stack yang menyatakan bahwa elemen terakhir yang dimasukkan (last in) akan menjadi elemen pertama yang dikeluarkan (first out). Dengan kata lain, elemen yang baru ditambahkan ke stack akan selalu diakses atau dihapus lebih dulu sebelum elemen-elemen yang ditambahkan sebelumnya.

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.



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:

a. Push (Tambah): Menambah elemen baru ke bagian paling atas tumpukan.

b. Pop (Hapus): Mengeluarkan elemen dari bagian paling atas tumpukan.

c. Top (Elemen Teratas): Mengakses atau melihat elemen di bagian atas tumpukan tanpa menghapusnya.

d. IsEmpty (Kosong): Mengecek apakah tumpukan tidak memiliki elemen.

e. IsFull (Penuh): Mengecek apakah tumpukan telah mencapai kapasitas maksimum (terutama pada tumpukan dengan batasan ukuran).

f. Size (Jumlah Elemen): Mengembalikan total elemen yang ada dalam tumpukan.

g. Peek (Lihat Elemen): Menampilkan elemen di posisi tertentu dalam tumpukan tanpa mengubah isinya.

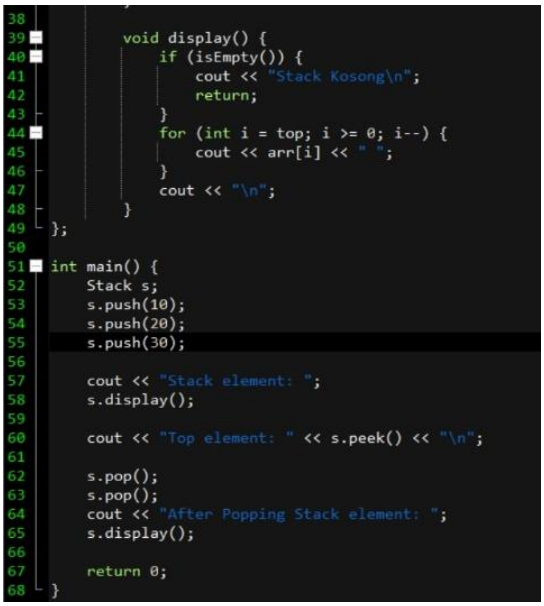
h. Clear (Kosongkan): Menghapus semua elemen dari tumpukan.

i. Search (Cari Elemen): Memeriksa apakah elemen tertentu ada di dalam tumpukan.

IV. GUIDED

1. Main.cpp

```cpp
#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 Kosong\n";
            return -1;
        }
```



```cpp
        void display() {
            if (isEmpty()) {
                cout << "Stack Kosong\n";
                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 element: ";
    s.display();

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

    s.pop();
    s.pop();
    cout << "After Popping Stack element: ";
    s.display();

    return 0;
}
```

2. MainPointer.cpp

```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;
        }

        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 element: ";
    s.display();

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

    s.pop();
    s.pop();
    cout << "After Popping Stack element: ";
    s.display();

    return 0;
```

3. Output

```
D:\ITT SM 3\07_STACK\mainP
Stack element: 30 20 10
Top element: 30
After Popping Stack element: 10

--------------------------------
Process exited after 0.6669 seconds with return value 0
Press any key to continue . . .
```

V. UNGUIDED

1. Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

```cpp
#include <iostream>
#include <stack>
#include <string>
#include <cctype>

using namespace std;


bool isPalindrome(const string& str) {
    stack<char> s;
    string original;
    string reversed;

for (size_t i = 0; i < str.length(); ++i) {
    char c = str[i];
    if (isalnum(c)) {
        char lowerC = tolower(c);
        original += lowerC;
        s.push(lowerC);
    }
}

    while (!s.empty()) {
        reversed += s.top();
        s.pop();
    }

    return original == reversed;
}

int main() {
    string input;

    cout << "Masukan kalimat: ";
    getline(cin, input);

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

    return 0;
}
```

```
D:\ITT SM 3\07_STACK\ung1.e
Masukan kalimat: Ilham
Kalimat ini bukan palindrom.

--------------------------------
Process exited after 3.112 seconds with return value 0
Press any key to continue . . .
```

```
D:\ITT SM 3\07_STACK\ung1.e
Masukan kalimat: ini
Kalimat tersebut adalah palindrom.

--------------------------------
Process exited after 47.18 seconds with return value 0
Press any key to continue . . .
```

2. Buatlah program untuk melakukan pembalikan terhadap kalimat menggunakan stack dengan minimal 3 kata. Jelaskan output program dan source codenya beserta operasi/fungsi yang dibuat?

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

// Kelas Stack berbasis array
class Stack {
private:
    static const int MAX = 100; // Kapasitas maksimum stack
    char data[MAX];             // Array untuk menyimpan data stack
    int top;                    // Indeks elemen teratas di stack

public:
    Stack() : top(-1) {} // Konstruktor: inisialisasi stack kosong

    // Fungsi untuk menambahkan elemen ke stack
    void push(char c) {
        if (top >= MAX - 1) {
            cout << "Stack overflow!" << endl;
            return;
        }
        data[++top] = c; // Tambahkan elemen dan geser top
    }

    // Fungsi untuk menghapus elemen dari stack
    char pop() {
        if (isEmpty()) {
            cout << "Stack underflow!" << endl;
            return '\0';
        }
        return data[top--]; // Kembalikan elemen dan kurangi top
    }

    // Fungsi untuk memeriksa apakah stack kosong
    bool isEmpty() {
        return top == -1;
    }
};

int main() {
    string input;

    // Input dari pengguna
    cout << "Masukkan kata: ";
    getline(cin, input);

    Stack s;
    string reversed = "";

    // Masukkan setiap karakter ke dalam stack
    for (size_t i = 0; i < input.length(); ++i) {
        s.push(input[i]);
    }

    // Pop elemen dari stack untuk membuat string terbalik
    while (!s.isEmpty()) {
        reversed += s.pop();
    }

    // Tampilkan hasil
    cout << "Datastack Array: " << endl;
    cout << "Data: " << reversed << endl;

    return 0;
}
```

Output

```
D:\ITT SM 3\07_STACK\2ung.e
Masukkan kata: Telkom University Purwokerto
Datastack Array:
Data: otrekowruP ytisrevinU mokleT

--------------------------------
Process exited after 16.5 seconds with return value 0
Press any key to continue . . .
```

VI. KESIMPULAN

Pada praktikum modul 7 ini, diajarkan modul Stack yang memiliki prinsip dasar LIFO, yang mana elemen yang terakhir dimasukkan akan pertama kali dikeluarkan. selain itu, pada praktikum ini berhasil mengimplementasikan operasi-operasi dasar stack seperti push, pop, top, dan isEmpty dalam program, serta mengaplikasikan stack untuk menyelesaikan masalah

seperti menentukan palindrom dan membalik kalimat. Praktikum ini memberikan pemahaman yang lebih dalam tentang struktur data stack dan bagaimana cara kerjanya dalam berbagai aplikasi pemrograman. Serta saya siap untuk melanjutkan praktikum yang lain