200650U – Thilakarathne D.L.J - CS2023 – Lab 6

Github link - https://github.com/LasithaJananjaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab6

Stack using Array

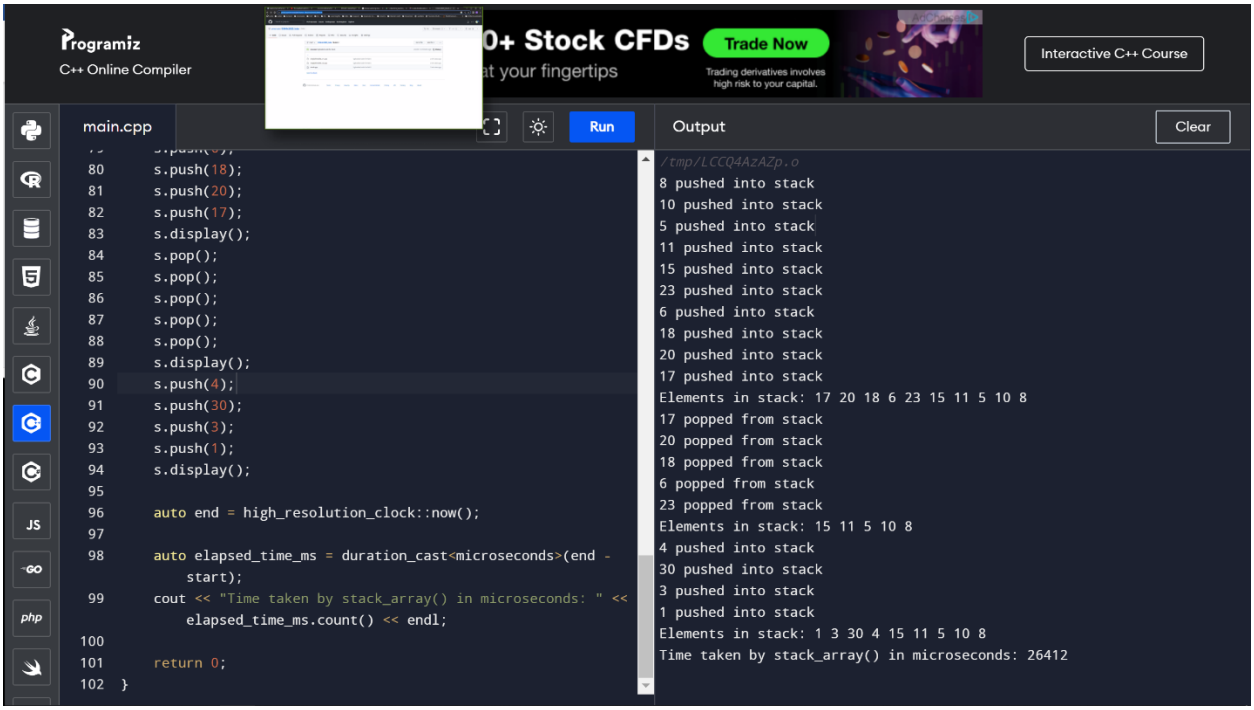
```cpp
#include <iostream>
#include <chrono>
#include <ctime>
using namespace std;
using namespace std::chrono;

#define MAX_SIZE 20 // define maximum size of stack

class Stack {
    private:
        int top;
        int arr[MAX_SIZE];

    public:
        Stack() { // constructor to initialize top to -1
            top = -1;
        }

        void push(int x) {
            if (isFull()) { // check if stack is full
                cout << "Stack Overflow!" << endl;
                return;
            }
            top++; // increment top
            arr[top] = x; // insert element
            cout << x << " pushed into stack" << endl;
        }

        void pop() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack Underflow!" << endl;
                return;
            }
            int popped = arr[top]; // get the element to be popped
            top--; // decrement top
            cout << popped << " popped from stack" << endl;
        }

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

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

        int stackTop() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack is empty!" << endl;
                return -1;
            }
            return arr[top]; // return the topmost element
        }

        void display() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack is empty!" << endl;
                return;
            }
            cout << "Elements in stack: ";
            for (int i = top; i >= 0; i--) { // loop through the stack
                cout << arr[i] << " ";
            }
            cout << endl;
        }
};

int main() {
    Stack s;

    auto start = high_resolution_clock::now();

    s.push(8);
    s.push(10);
    s.push(5);
    s.push(11);
    s.push(15);
    s.push(23);
    s.push(6);
    s.push(18);
    s.push(20);
    s.push(17);
    s.display();
    s.pop();
    s.pop();
    s.pop();
    s.pop();
    s.pop();
    s.display();
    s.push(4);
    s.push(30);
    s.push(3);
    s.push(1);
    s.display();

    auto end = high_resolution_clock::now();

    auto elapsed_time_ms = duration_cast<microseconds>(end - start);
    cout << "Time taken by stack_array() in microseconds: " << elapsed_time_ms.count() <<
endl;
    return 0;
}
```

Stack using LinkedList

```cpp
#include <iostream>
#include <chrono>
#include <ctime>
using namespace std;
using namespace std::chrono;

class Node {
    public:
        int data;
        Node* next;
};

class Stack {
    private:
        Node* top;

    public:
        Stack() { // constructor to initialize top to NULL
            top = NULL;
        }

        void push(int x) {
            Node* temp = new Node; // create a new node
            temp->data = x; // set data of node to x
            temp->next = top; // set next pointer of node to top
            top = temp; // make the new node as top
            cout << x << " pushed into stack" << endl;
        }

        void pop() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack Underflow!" << endl;
                return;
            }
            Node* temp = top; // get the node to be popped
            int popped = temp->data; // get the data of node
            top = temp->next; // make the next node as top
            delete temp; // delete the node
            cout << popped << " popped from stack" << endl;
        }

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

        bool isFull() {
            return false; // linked list implementation of stack cannot be full
        }

        int stackTop() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack is empty!" << endl;
                return -1;
            }
            return top->data; // return the data of top node
        }

        void display() {
            if (isEmpty()) { // check if stack is empty
                cout << "Stack is empty!" << endl;
                return;
            }
            cout << "Elements in stack: ";
            Node* temp = top; // start from top node
            while (temp != NULL) { // loop through the stack
                cout << temp->data << " ";
                temp = temp->next; // move to next node
            }
            cout << endl;
        }
};

int main() {
    Stack s;

    auto start = high_resolution_clock::now();

    s.push(8);
    s.push(10);
    s.push(5);
    s.push(11);
    s.push(15);
    s.push(23);
    s.push(6);
    s.push(18);
    s.push(20);
    s.push(17);
    s.display();
    s.pop();
    s.pop();
    s.pop();
    s.pop();
    s.pop();
    s.display();
    s.push(4);
    s.push(30);
    s.push(3);
    s.push(1);
    s.display();

    auto end = high_resolution_clock::now();

    auto elapsed_time_ms = duration_cast<microseconds>(end - start);
    cout << "Time taken by stack_array() in microseconds: " << elapsed_time_ms.count() <<
endl;
    return 0;
}
```

Stack implemented using LinkedList took less time than stack implemented using Array.

Stack - Array



Stack - LinkedList