DATA STRUCTURES & ALGORITHMS LAB

Muhammad Anas
55412

Lab 04

- Implement Following Operations

1) Stack (int ignored = 0)

Requirements: None

Results: Constructor. Creates an empty stack.

2) ~Stack ()

Requirements: None

Results: Destructor. Deallocates (frees) the memory used to store a stack.

3) void push (const DataItem)

Requirements: None

Results: Push the element at top of the stack.

4) Void pop ()

Requirements: Stack is not empty Results: Returns the

element from the top of the stack.

5) element Peek ()

return element at the top of stack

6) void clear ()

Requirements: None

Results: Removes all the elements from a stack.

7) Bool isEmpty ()

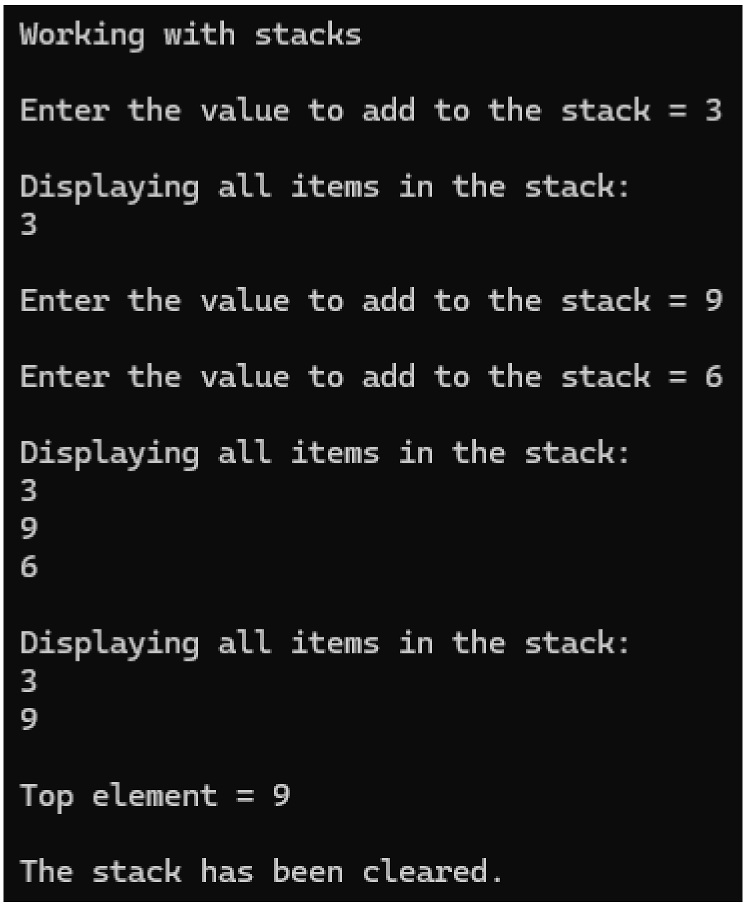
Requirements: None

Results: Returns true if a stack is empty. Otherwise, returns false.

- Write a program in C++ to reverse a string (Data Structures) using stack.

```
#include <iostream>
using namespace std;
class Stack
{
private:
   static const int MAX_SIZE = 10;
   int top;
   int array[MAX_SIZE];
public:
   Stack() : top(-1) {}
   void push()
   {
      int value;
      cout << "\n Enter the value to add to the stack = ";
      if (top == MAX_SIZE - 1)
      {
         cout << " Stack overflow" << endl;
      }
      else
      {
        cin >> value;
        array[++top] = value;
    }
}
void pop()
{
    if (top < 0)
    {
        cout << " The stack is empty" << endl;
    }
    else
    {
        --top;
    }
}
void show() const
{
    cout << "\n Displaying all items in the stack:" << endl;
    for (int i = 0; i <= top; i++)
    {
        cout << " ";
        cout << array[i] << endl;
    }
}
void peek() const
{
    if (top < 0)
    {
        cout << " The stack is empty" << endl;
        }
        else
        {
            cout << "\n Top element = " << array[top] << endl;
        }
    }
    void clear()
    {
        top = -1;
        cout << "\n The stack has been cleared." << endl;
    }
};
int main()
{
    cout << " Working with stacks" << endl;
    Stack stack;
    stack.push();
    stack.show();
    stack.push();
    stack.push();
    stack.show();
    stack.pop();
    stack.show();
    stack.peek();
    stack.clear();
    return 0;
}
```

Output



```
Working with stacks

Enter the value to add to the stack = 3

Displaying all items in the stack:
3

Enter the value to add to the stack = 9

Enter the value to add to the stack = 6

Displaying all items in the stack:
3
9
6

Displaying all items in the stack:
3
9

Top element = 9

The stack has been cleared.
```