Abdul Munim

  55470

Data structures & algorithms lab

**Lab 04**

* Implement Following Operations

1. Stack (int ignored = 0)

Requirements: None

Results: Constructor. Creates an empty stack.

1. ~Stack ()

Requirements: None

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

1. void push (const DataItem)

Requirements: None

Results: Push the element at top of the stack.

1. Void pop ()

Requirements: Stack is not empty Results: Returns the

element from the top of the stack.

1. element Peek ()

return element at the top of stack

1. void clear ()

Requirements: None

Results: Removes all the elements from a stack.

1. 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**

A screenshot of a black screen

Description automatically generated