Lab Report 06

Stack Using Array

Submitted By

Muhammad Awais / 2883

Submitted To

Ma'am Zanaib Malik

Department of Computer Science and Engineering

National University of modern language, Islamabad

List of Task

1. Node Class
2. Function Push
3. Pop Function
4. IsEmpty
5. IsFull
6. Top Value
7. String Reversal

1. Description of Node Class

This is the declaration of a template class Stack that represents a stack data structure. It has a private data member top that represents the index of the top element in the stack and another private data member size that represents the maximum capacity of the stack. The actual elements of the stack are stored in a dynamic array pointed to by the arr pointer.

The constructor of the Stack class takes an integer argument size that specifies the maximum capacity of the stack. It initializes the top member to -1 and assigns the value of size to the size member. It also dynamically allocates an array of type t (the template parameter) with size size using the new operator, and assigns the pointer to this array to the arr member.

The Stack class also provides several public member functions:

push: This function takes an argument of type t and adds it to the top of the stack.
pop: This function removes and returns the top element from the stack.
isEmpty: This function returns a boolean value indicating whether the stack is empty.
isFull: This function returns a boolean value indicating whether the stack is full.
topValue: This function returns the value of the top element in the stack without removing it.

Source Code

```
#include<iostream>
using namespace std;
template<class t>
class Stack
{
     private:
     int top;
     int size;
     // t arr[size]; // Static declaration/ compile time
     t *arr;
      // arr=new t(size)
      public:
```

```
        Stack(int size )
        {
            top=-1;
            this->size=size;
            arr=new t(size);
             }

             void push(t element);
             t pop();
             bool isEmpty();
             bool isFull();
             t topValue();

}; // end of class Stack
```

2. Description of Push Function

This is the implementation of the push function for the Stack class template. It takes an argument element of type t (the template parameter) and adds it to the top of the stack.

First, the function checks if the stack is already full by comparing the top member to size-1 (the index of the last element in the stack). If the stack is full, it prints an error message to the standard error output stream (cerr) indicating that a stack overflow has occurred.

If the stack is not full, the function increments the top member to point to the next empty position in the stack and assigns the value of element to the array element at this position (i.e., arr[top]). This effectively adds the new element to the top of the stack.

Note that this implementation assumes that the stack is implemented as a continuous array, and that the top member represents the index of the top element in the stack (not the number of elements currently in the stack).

Source Code

```
template<class t>
void Stack<t>::push(t element)
{
    if(top==size-1)
    {
        cerr<<"Stack overflow \n";
    }

    else
```

```
    {
        top++;
        arr[top]=element;
    }

}
```

3. Description of Void Pop

Pop function for the Stack class template. It removes and returns the top element from the stack.

First, the function checks if the stack is already empty by comparing the top member to -1 (which is the initial value of top when the stack is empty). If the stack is empty, it prints an error message to the standard error output stream (cerr) indicating that the stack is empty.

If the stack is not empty, the function assigns the value of the top element in the stack (i.e., arr[top]) to a local variable element, and then decrements the top member to remove the top element from the stack. Finally, the function returns the value of element, which represents the top element that was removed from the stack.

Note that this implementation assumes that the stack is implemented as a continuous array, and that the top member represents the index of the top element in the stack (not the number of elements currently in the stack). Also note that this implementation does not explicitly free the memory allocated for the dynamic array arr, so it may result in a memory leak if the Stack object is destroyed without calling a destructor that releases this memory.

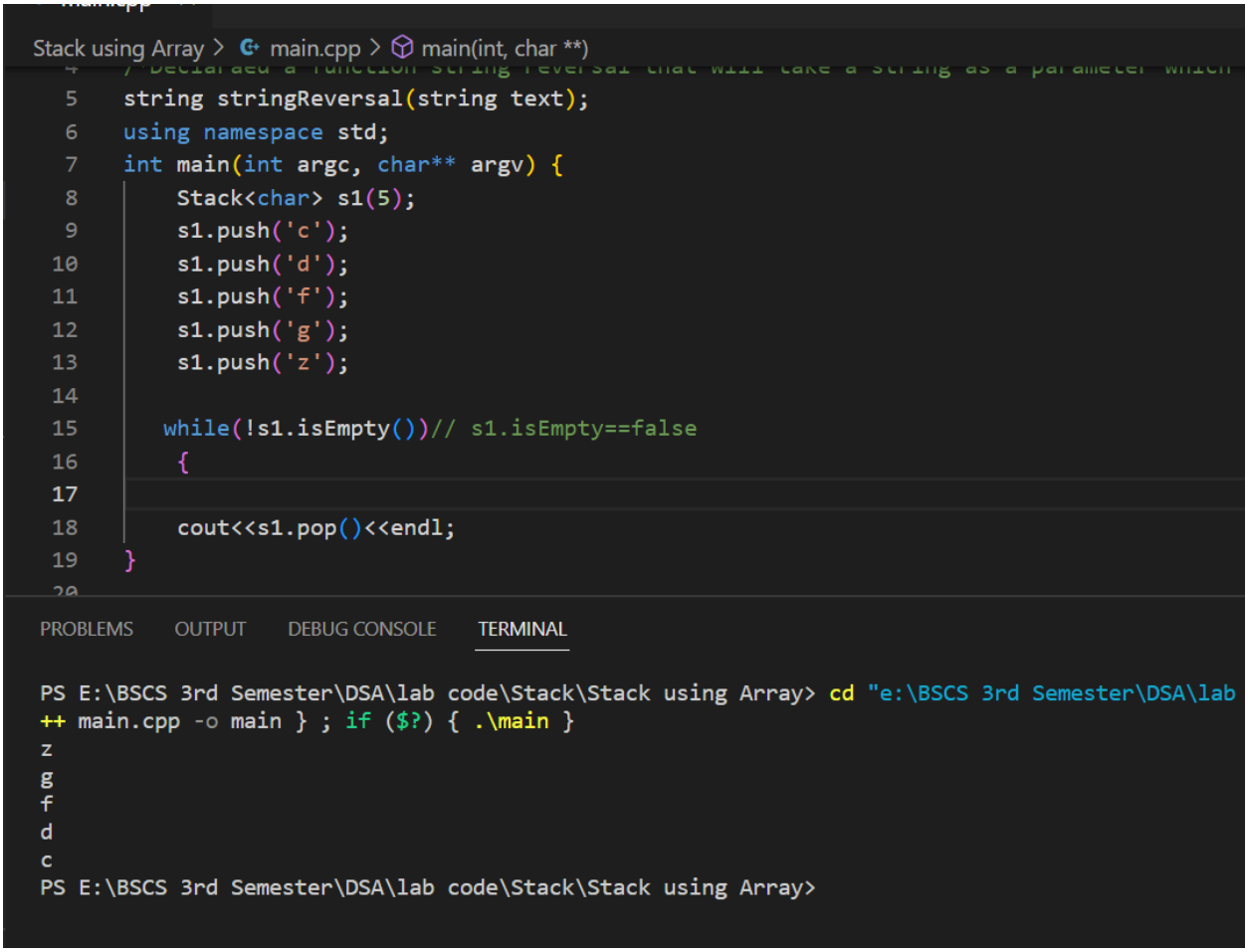
Source Code

```
template<class t>
t Stack<t>::pop()
{
    if(top<0) // ==-1
    {
        cerr<<"Stack is empty \n";
    }

    else
    {   t element=arr[top];
        top--;
        return element;
    }
```

```
}// end of Pop
```

Output



4. Description of IsEmpty

isEmpty function for the Stack class template. It returns a boolean value indicating whether the stack is empty.

The function checks if the top member is equal to -1, which is the initial value of top when the stack is empty. If top is equal to -1, the function returns true to indicate that the stack is empty. Otherwise, the function returns false to indicate that the stack is not empty.

Alternatively, the implementation could be simplified by returning the boolean expression top==-1 directly, without using an if statement. This expression evaluates to true if top is equal to -1, and to false otherwise.

Source Code

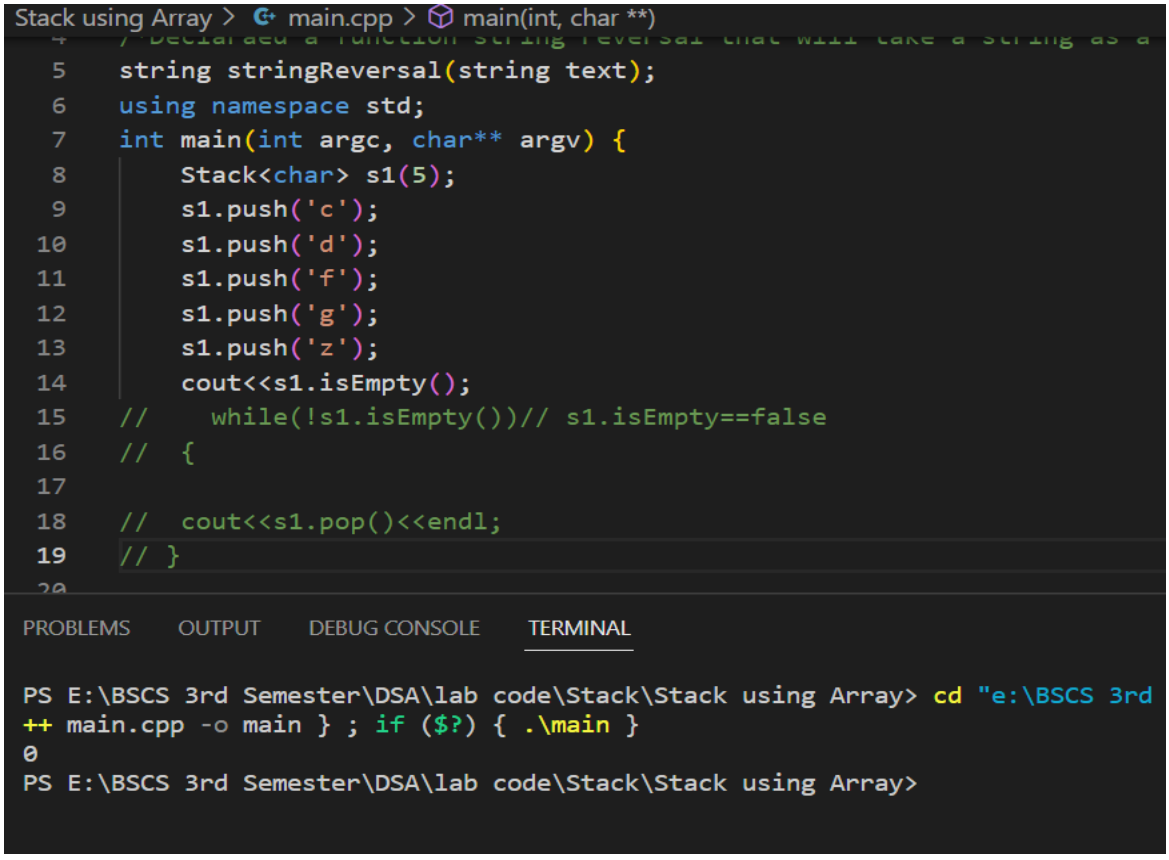
```
template<class t>
bool Stack<t>::isEmpty()
{
    if(top==-1)

        return true;


    else

        return false;
    // return top==-1;
}
```

Output



5. Description isFull

isFull function for the Stack class template. It returns a boolean value indicating whether the stack is full.

The function checks if the top member is equal to size-1, which is the index of the last element in the stack when the stack is full. If top is equal to size-1, the function returns true to indicate that the stack is full. Otherwise, the function returns false to indicate that the stack is not full.

Alternatively, the implementation could be simplified by returning the boolean expression top==size-1 directly, without using an if statement. This expression evaluates to true if top is equal to size-1, and to false otherwise

Source Code

```
template<class t>
bool Stack<t>::isFull()
{
    if(top==size-1)

        return true;


    else

        return false;
    // return top==size-1;

}// end
```

Output

```
Stack using Array > main.cpp > main(int, char **)
   5    string stringReversal(string text);
   6    using namespace std;
   7    int main(int argc, char** argv) {
   8        Stack<char> s1(5);
   9        s1.push('c');
  10        s1.push('d');
  11        s1.push('f');
  12        s1.push('g');
  13        s1.push('z');
  14        cout<<s1.isFull();
  15    //    while(!s1.isEmpty())// s1.isEmpty==false
  16    //  {
  17
  18    //  cout<<s1.pop()<<endl;
  19    // }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\BSCS 3rd Semester\DSA\lab code\Stack\Stack using Array> cd "e:\BSCS
++ main.cpp -o main } ; if ($?) { .\main }
1
PS E:\BSCS 3rd Semester\DSA\lab code\Stack\Stack using Array>
```

6. Description Top Value

TopValue function for the Stack class template. It returns the value of the top element in the stack without removing it.

First, the function checks if the stack is empty by comparing the top member to -1. If the stack is empty, it prints an error message to the standard error output stream (cerr) indicating that the stack is underflow (i.e., there are no elements in the stack to retrieve).

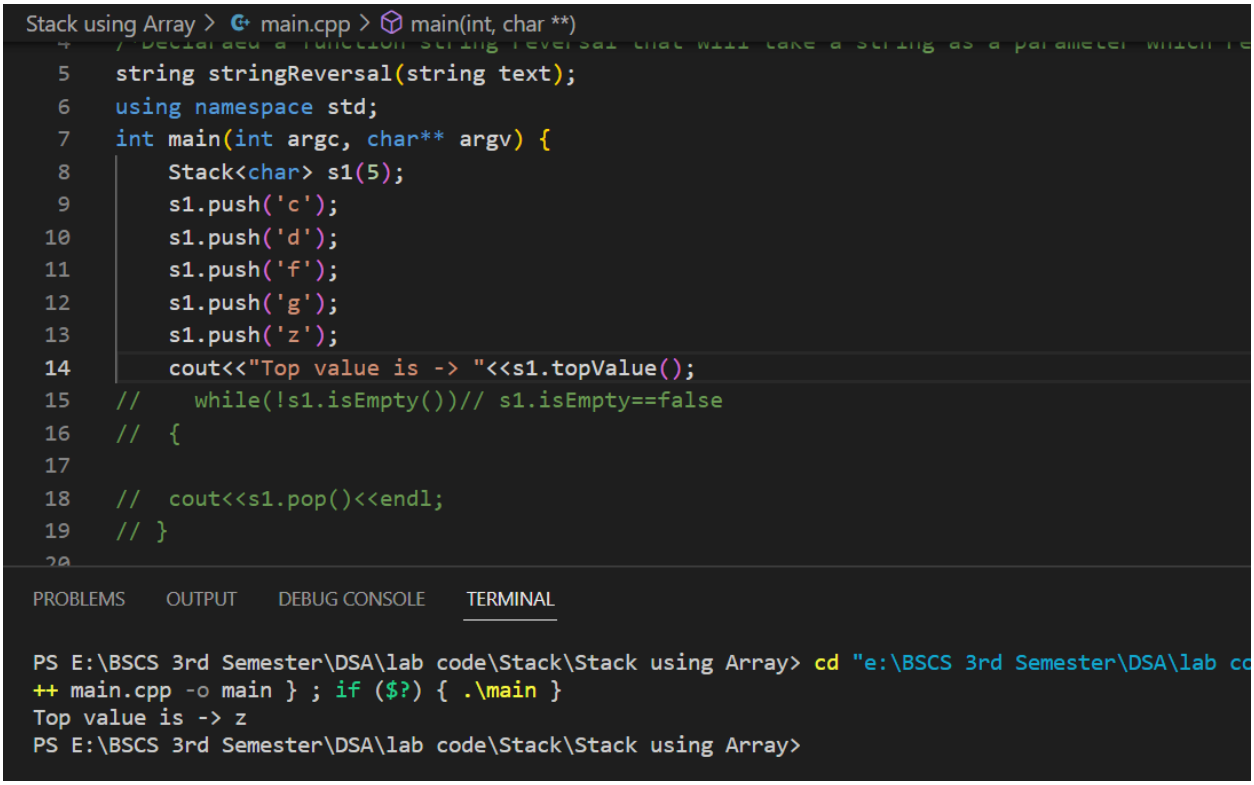
If the stack is not empty, the function returns the value of the top element in the stack (i.e., arr[top]). Note that this implementation assumes that the stack is implemented as a continuous array, and that the top member represents the index of the top element in the stack (not the number of elements currently in the stack). Also note that the implementation does not modify the stack in any way, so the stack remains unchanged after calling this function.

Source Code

```
template<class t>
t Stack<t>::topValue()
{
    if(top==-1)
    {
```

```
        cerr<<" Stack is underflow \n";
    }
    else
    {
        return arr[top];
    }
}// end
```

Output



7. Description of String Reversal

StringReversal that takes a string argument text and returns the reversed version of the input string.

The function first creates an instance of the Stack class template for char type, named s1, with a capacity equal to the length of the input string. Then, it iterates over each character in the input string using a for loop, and pushes each character onto the stack s1.

Next, the function enters a while loop that continues as long as the stack s1 is not empty (i.e., s1.isEmpty() returns false). Inside the loop, the function pops the top element from the stack s1

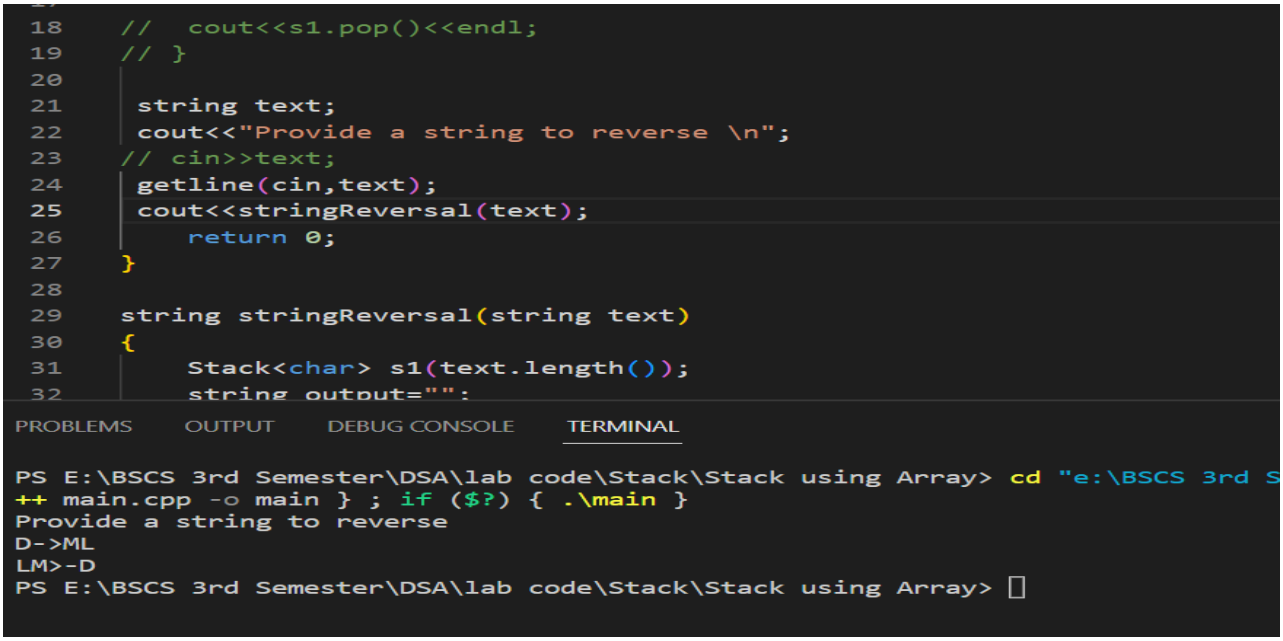
using the pop() function, and appends it to an empty string named output. This process effectively reverses the order of the characters in the input string.

Finally, the function returns the output string, which contains the reversed version of the input string.

Source Code

```
1. string stringReversal(string text)
2. {
3.     Stack<char> s1(text.length());
4.     string output="";
5.     for(int i=0; i<text.length(); i++)
6.     {
7.         s1.push(text[i]);
8.     }
9.
10.    while(!s1.isEmpty())// s1.isEmpty==false
11.    {
12.
13.    output=output+s1.pop();
14.}
15.return output;
16.}
```

Output



End of Lab 06