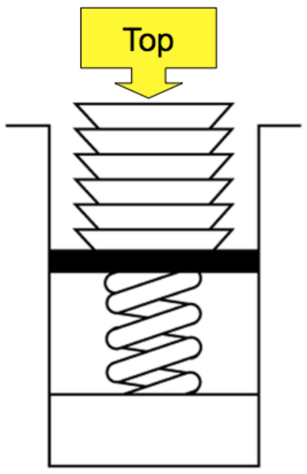
CSCI 2170 ADT Stack (C++ Container)



Characteristic: **Last In First Out (LIFO)**

Operations:

- Create an empty stack
- Destroy a stack
- Determine whether a stack is empty -- **empty()**
- Add a new item to the stack – **push(ItemType newItem)**
- Remove from the stack the item that was added most recently -- **pop()**
- Retrieve the item that was added most recently -- **top()**

How to create an empty stack using the C++ Stack Container

```
stack <string>   stringStack;
stack<int>   intStack;
```

Applications of Stack

(1) Read characters and correct with backspace: reads the input line, for each character read, either enter it into stack S, if it is '←', correct the content of S

```
ReadAndCorrect(stack <char> aStack)
{
    success = true;
    Read a new character "newChar"

    while (newChar is not the end of line symbol && success)
    {
        if (newChar is not '←')
            aStack.push(newChar, success);
        else if  (!aStack.empty())
            aStack.pop();

        Read a new character "newChar"
    }
}
```

(2) Display the content of a stack : directly popping out the content of stack will display the letters in the word in reverse order

```
void DisplayBackward(stack <char> aStack)  {
    while (!aStack.empty())   {
        aStack.pop();
        Write newChar;
    }
}
```

?? How to write out the content of the stack in the original order when they were read?

```
void DisplayForward (stack <char> aStack)        {
    aStack <char>   tmpS;        char newChar;
    while (!aStack.empty())    {
        newChar=aStack.top();
        aStack.pop();
        tmpS.push(newChar, success);
```

```
            }

            DisplayBackward(tmpS);
    }
```
?? What is the content of the stack after executing this function?

?? How to count the number of items stored in a stack and keep the content of the stack unchanged after the operation??

(3) Checking for balanced braces

- each time a '{' is encountered, push it onto the stack
- each time a '}' is entered, it is matched to an already encountered '{', pop stack
- **Balanced :** when reaching the end of the string, all the '{' has been matched against (stack is empty)
- **NOT balanced** :
  1. when a '}' is entered, there is no existing '{' to match, OR
  2. when reaching the end of the string, there are still some '{' not being matched (stack not empty)

```
void CheckBalanced(string program)  {
        int  index = 0;
        bool balanced = true, success = false;
        stack  <char> braces;
        while (balanced && index < strlen(program))   {
                ch = program [index];
                index ++;

                if (ch == '{')
                        braces.push(ch);
                else if (ch == '}')    {
                        if ( ! braces.empty())
                                braces.pop();
                        else
                                balanced = false;
                }
        }

        if (balanced && braces.empty())
                cout << "The braces in this program are balanced." << endl;
        else
                cout << "Syntax error: Braces are NOT balanced." << endl;
}
```

(4) Arithmetic Expression Evaluation

Infix notation	**2 * (3 + 4)**
Postfix notation	**2 3 4 + ***

How to evaluate postfix expressions?

Pseudocode:

```
stack <char> aStack;
for each ch in the string{
    if (ch is an operand)
        push operand onto the stack
    else if (ch is an operator) {
        // evaluate and push the result
        op2 = aStack.top()
        aStack.pop();
        op1 = aStack.top()
        aStack.pop()
        result = op1 op op2
        aStack.push(result)
    }
```

?? What are the values of these postfix expressions:

- 50 10 – 40 + 30 20 - *
- 30 20 20 10 - - * 10 +