Лабораторная работа

Обратная польская запись (англ. Reverse Polish notation) — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная бесскобочная запись, постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись, ПОЛИЗ.

Обратная польская запись имеет ряд преимуществ перед инфиксной записью при выражении алгебраических формул:

Во-первых, любая формула может быть выражена без скобок.

Во-вторых, она удобна для вычисления формул в машинах со стеками.

В-третьих, инфиксные операторы имеют приоритеты, которые произвольны и нежелательны.

# История создания

Обратная польская нотация (ОПН) была разработана австралийским философом и специалистом в области теории вычислительных машин Чарльзом Хэмблином в середине 1950-х на основе польской нотации, которая была предложена в 1920 году польским математиком Яном Лукасевичем. Работа Хэмблина была представлена на конференции в июне 1957, и издана в 1957 и 1962.

Первыми компьютерами, поддерживающими обратную польскую нотацию, были KDF9 от English Electric Company, который был анонсирован в 1960 и выпущен (появился в продаже) в 1963, и американский Burroughs B5000, анонсирован в 1961, выпущен в том же 1963. Один из проектировщиков B5000, Р. С. Бартон, позже написал, что разработал обратную польскую запись независимо от Хэмблина, примерно в 1958, в процессе чтения книги по символьной логике, и до того как познакомился с работой Хэмблина.

# Постановка задачи

На вход программы поступает выражение, состоящее из чисел и знаков арифметических действий. Требуется преобразовать это выражение в обратную польскую запись или же сообщить об ошибке.

Контейнер используемый для реализации задачи – **Stack**.

# Теоретическая часть Stack

**Написание контейнера для хранения исходных данных**

В контейнере типа Stack были реализованы следующие методы:

1. Top – возвращает значение элемента стека лежащего на вершине.
2. Pop – удалить элемент;

При удалении элемента из стека необходимо возвратить значение из динамического массива по индексу вершины стека, переместить указатель вершины стека и уменьшить количество элементов.

1. IsEmpty – проверить стек на пустоту;

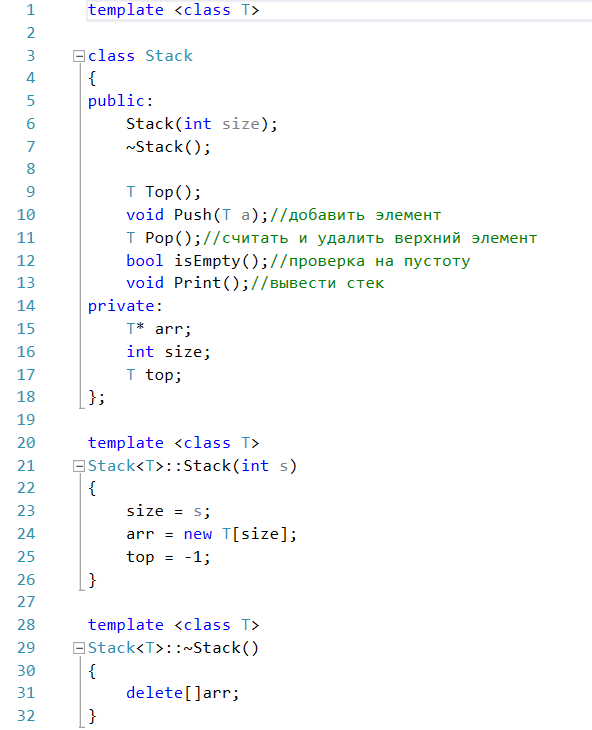
Стек пуст, если в нем нет ни одного элемента, т.е. когда количество элементов равно нулю.

1. Push – добавить элемент;

При добавлении элемента в стек необходимо переместить указатель вершины стека, записать элемент в соответствующую позицию динамического массива и увеличить количество элементов.

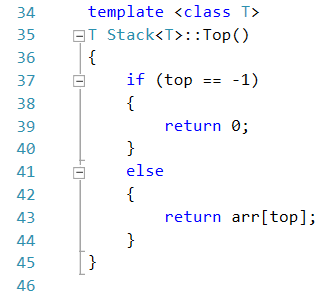
# Практическая часть Stack

Реализация Stack:

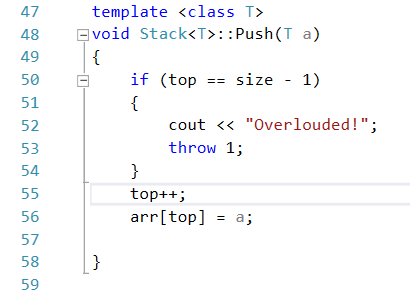
****

Реализация методов:

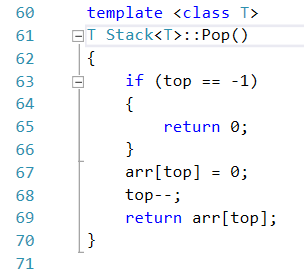
1. Top



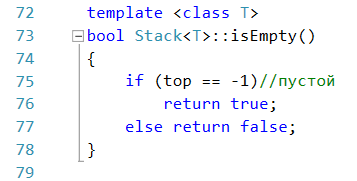
1. Push



1. Pop



1. IsEmpty



# Теоретическая часть RPN



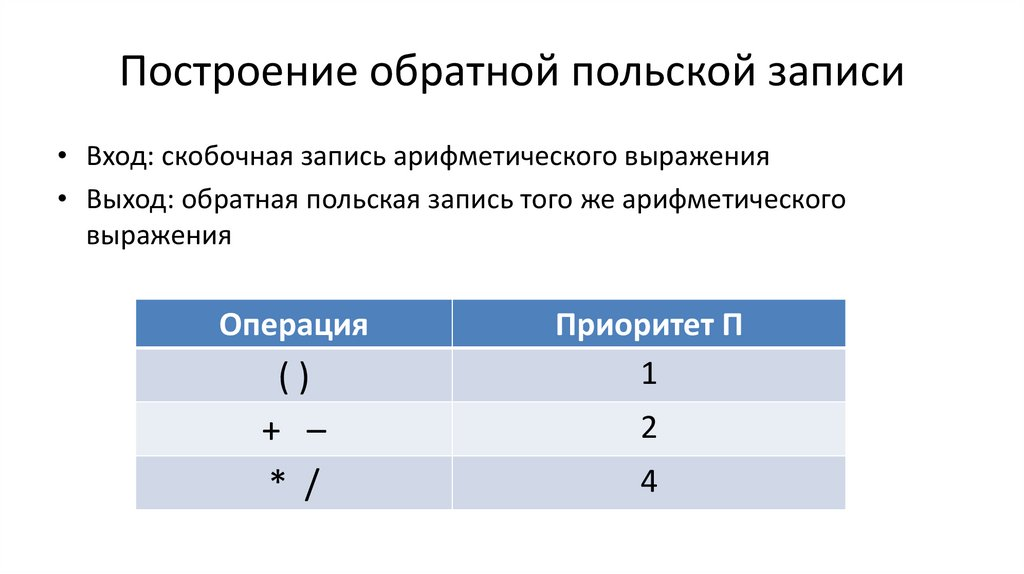
1. **Проверка правильности расстановки скобок**

На вход алгоритма поступает строка символов, на выходе должна быть выдана таблица соответствия номеров открывающихся и закрывающихся скобок и общее количество ошибок.

Идея алгоритма, решающего поставленную задачу, состоит в следующем:

1. Выражение просматривается посимвольно слева направо. Все символы, кроме скобок, игнорируются (т.е. просто производится переход к просмотру следующего символа).
2. Если очередной символ – открывающая скобка, то её порядковый номер помещается в стек. Если очередной символ – закрывающая скобка, то производится выталкивание из стека номера открывающей скобки и запись этого номера в паре с номером закрывающей скобки в результирующую таблицу.
3. Если в этой ситуации стек оказывается пустым, то вместо номера открывающей скобки записывается 0, а счетчик ошибок увеличивается на единицу.
4. Если после просмотра всего выражения стек оказывается не пустым, то выталкиваются все оставшиеся номера открывающих скобок и записываются в результирующий массив в паре с 0 на месте номера закрывающей скобки, счетчик ошибок каждый раз увеличивается на единицу.
5. **Перевод в постфиксную форму**

Данный алгоритм основан на использовании стека. На вход алгоритма поступает строка символов, на выходе должна быть получена строка с постфиксной формой. Каждой операции и скобкам приписывается приоритет.



Предполагается, что входная строка содержит синтаксически правильное выражение. Входная строка просматривается посимвольно слева направо до достижения конца строки. Операндами будем считать любую последовательность символов входной строки, не совпадающую со знаками определенных в таблице операций. Операнды по мере их появления переписываются в выходную строку. При появлении во входной строке операции, происходит вычисление приоритета данной операции.

Знак данной операции помещается в стек, если:

1. Приоритет операции равен 0 (это « ( » )
2. Приоритет операции строго больше приоритета операции, лежащей на вершине стека
3. Стек пуст

В противном случае из стека извлекаются все знаки операций с приоритетом больше или равным приоритету текущей операции. Они переписываются в выходную строку, после чего знак текущей операции помещается в стек. Имеется особенность в обработке закрывающей скобки. Появление закрывающей скобки во входной строке приводит к выталкиванию и записи в выходную строку всех знаков операций до появления открывающей скобки. Открывающая скобка из стека выталкивается, но в выходную строку не записывается. Таким образом, ни открывающая, ни закрывающая скобки в выходную строку не попадают. После просмотра всей входной строки происходит последовательное извлечение всех элементов стека с одновременной записью знаков операций, извлекаемых из стека, в выходную строку.

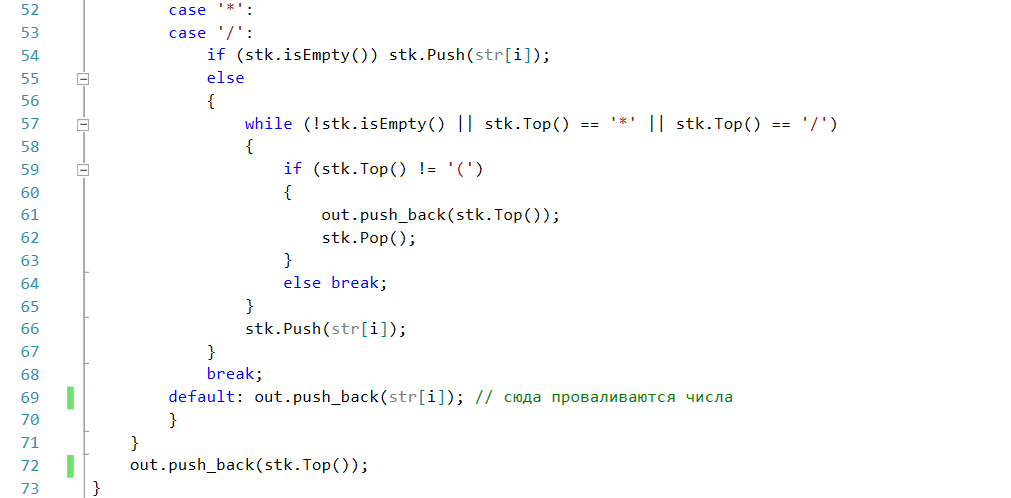
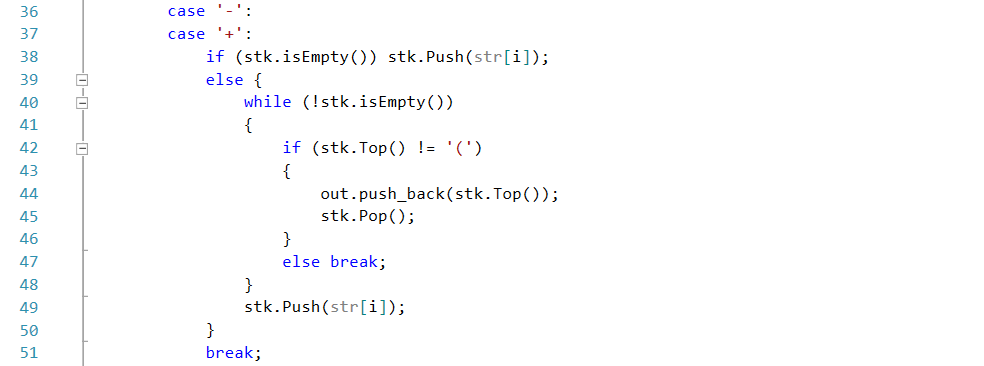
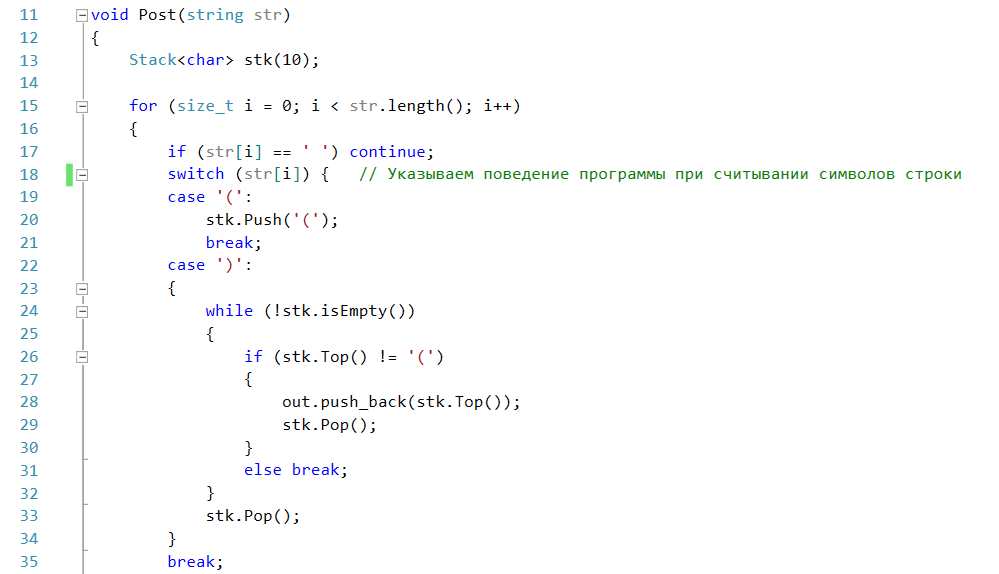
1. **Вычисление арифметического выражения**

Алгоритм вычисления арифметического выражения за один просмотр входной строки основан на использовании постфиксной формы записи выражения и работы со стеком. Входным данным служит строка символов, полученная в результате работы алгоритма перевода в постфиксную запись, выходным – результат вычисления выражения.

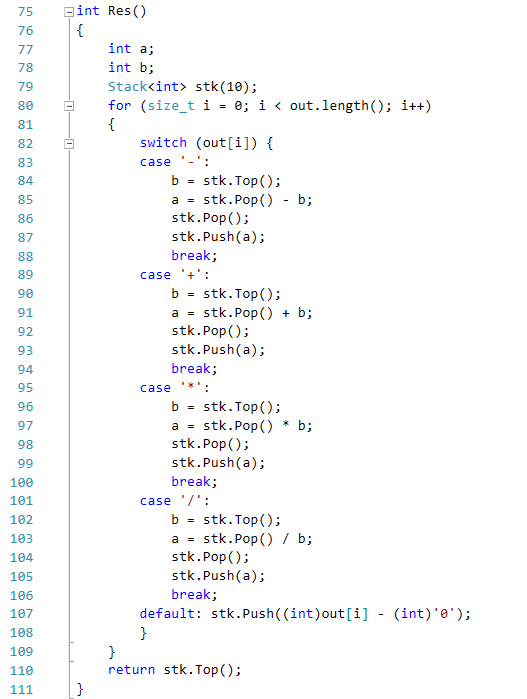
Выражение просматривается посимвольно слева направо. При обнаружении операнда производится перевод его в числовую форму и помещение в стек (если операнд не является числом, то вычисление прекращается с выдачей сообщения об ошибке.) При обнаружении знака операции происходит извлечение из стека двух значений, которые рассматриваются как операнд O2 и операнд O1 соответственно, и над ними производится обрабатываемая операция. Результат этой операции помещается в стек. По окончании просмотра всего выражения из стека извлекается окончательный результат.

# Практическая часть RPN

1. Перевод арифметического выражения в постфиксную форму



1. Вывод результата

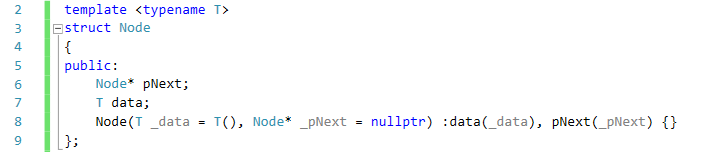


P.S. результат не выводится для чисел > 9.

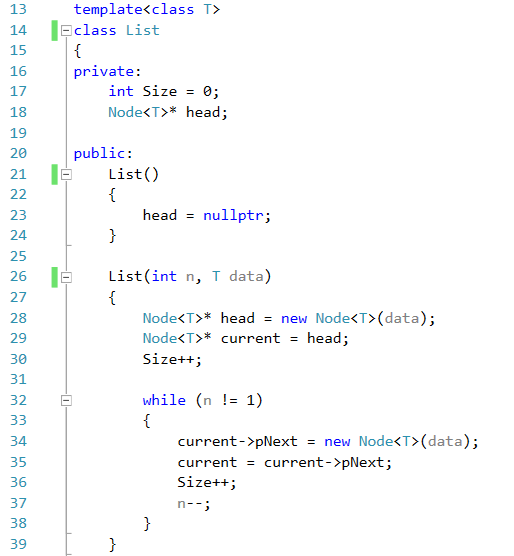
# Реализация Stack на односвязном списке (LStack)

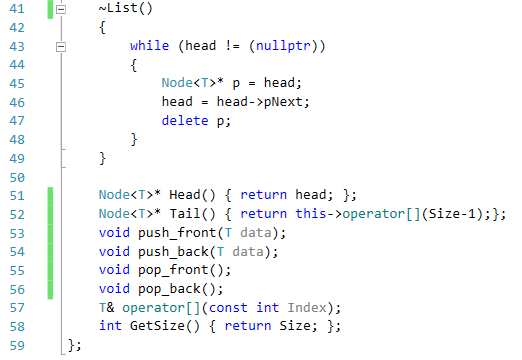
Для начала опишу класс List на основе которого будет реализован LStack.

Определяем содержимое узла (Node) :

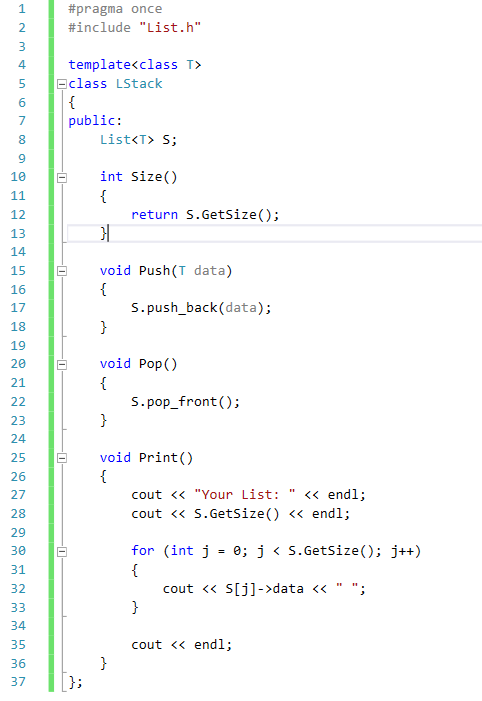


Тело List:



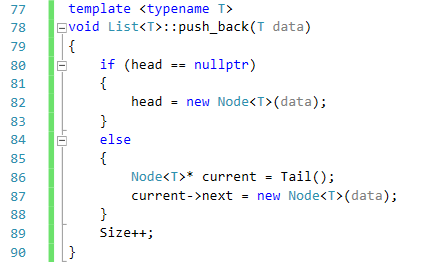


LStack:

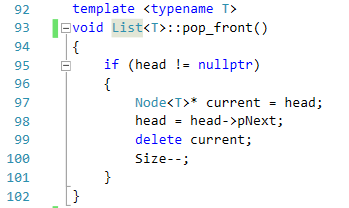


Не забываем, что стек — это структура данных, которая работает по принципу FILO, что обязывает нас использовать следующие методы для реализации:

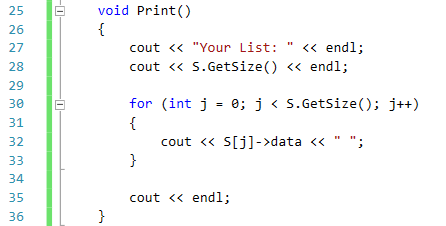
1. Push (push\_back из List) (Добавляет элемент в начало)



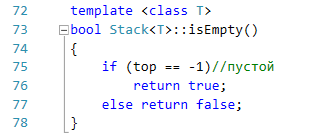
1. Pop (Pop\_front из List) (Удаляет элемент из начала)



1. Print (для вывода стека)



1. isEmpty (проверка на наличие элементов)



# Вывод

Я познакомился с таким способом записи арифметических выражений как постфиксная запись. Выделил ее плюсы и минусы, построил алгоритм действия и реализовал на практике, используя контейнер хранения Stack.

# Использованный материал:

Д. Э. Кнут “Искусство программирования”

wikipedia.org

[https://ru.wikipedia.org/wiki/Обратная\_польская\_запись](https://ru.wikipedia.org/wiki/)

studytonight.com

<https://www.studytonight.com/data-structures/stack-data-structure>

codelessons.ru

<https://codelessons.ru/cplusplus/realizaciya-steka-stack-v-c.html>

habr.com

<https://habr.com/ru/post/100869/>

Ссылка на репозиторий:

<https://github.com/ITrickStar/Deus-Vult/tree/master/Test>