# Самостоятельная работа «Стек»

# **Вариант 1**

Реализуйте функцию, указанную в варианте

В реализации используйте шаблон «ограниченный стек» (см. Задание 2)

Выполните тестирование

# **Функция перевода арифметического выражения из инфиксной формы в постфиксную (обратную польскую запись).**

Функция должна формировать выражение в постфиксной форме и генерировать исключение, если возникает ошибка (например, при переполнении стека, неправильной последовательности операндов и операций, …)

Считать, что выражение состоит из операндов (***одноразрядных*** десятичных чисел), знаков ***бинарных*** операций ( +, -, \*, / ) и ***круглых*** скобок.

Перевод выполнить за ***один проход*** по выражению, в процессе выполнения которого проверить правильность операндов, допустимость операций, правильность скобок.

Прототип функции:

void getPostfixFromInfix (const char\* infix, char\* postfix,

const size\_t stackSize);

или

void getPostfixFromInfix (const string& infix, string& postfix,

const size\_t stackSize);

infix - выражение в инфиксной форме

postfix - выражение в постфиксной форме

stackSize - максимальный размер стека операций

# **Алгоритм получения обратной польской записи *(!!! Корректное выражение)***

1. Подготовить пустой стек.
2. Просматривать исходную строку символов слева направо.
3. Операнды переписывать в выходную строку, а знаки бинарных операций заносить в стек по следующему правилу:
4. если стек пуст, то операцию из входной строки занести в стек;
5. если стек не пуст, операцию занести в стек, предварительно вытолкнуть из стека в выходную строку все операции с большим или равным приоритетом;
6. если очередной символ из исходной строки - открывающая скобка, то занести его в стек;
7. если очередной символ из исходной строки – закрывающая скобка, то вытолкнуть из стека все операции до ближайшей открывающей скобки, сами скобки в выходную строку не переписывать.
8. В завершении выражения оставшиеся операции из стека переносятся в выходную строку.

# **Вариант 2**

Реализуйте функцию, указанную в варианте

В реализации используйте шаблон «ограниченный стек» (см. Задание 2)

Выполните тестирование

**Функция вычисления значения арифметического выражения в постфиксной форме**

Функция должна вычислять значение выражения в постфиксной форме или генерировать исключение, если возникают ошибки, например, в постфиксном выражении, при переполнении стека…

Считать, что выражение состоит из операндов (***одноразрядных*** десятичных чисел), знаков ***бинарных*** операций ( +, -, \*, / ). Деление – целочисленное.

Прототип функции:

int evaluatePostfix (const char\* infix, const size\_t stackSize);

или

int evaluatePostfix (const string& infix, const size\_t stackSize);

postfix - выражение в постфиксной форме

stackSize - максимальный размер стека операндов

# **Алгоритм обработки входного постфиксного выражения (!!!корректного, не пустого!):**

Значения операндов и операций представлены одиночными символами, причем все операции двухместные, операции одного приоритета выполняются слева направо (нет возведения в степень).

1. Подготовить пустой стек.
2. Если очередной элемент потока - операнд (число), то поместить его в стек и перейти к пункту 2
3. Если очередной элемент потока – операция (двухместная), вытащить из стека, по очереди, два верхних элемента (это числа); рассматривая их, соответственно, как 2-й и 1-й операнды двухместного выражения, вычислить его значение; поместить его в стек; перейти к пункту 2.
4. Если входной поток пуст, то вытащить элемент из стека, вывести его значение в качестве результата и закончить обработку.