Отчёт по лабораторной работе

Вычисление арифметических выражений

Выполнила:
студент ИИТММ гр. 3821Б1
ПМ3

Качалин А.Г.

Нижний Новгород
2021 г

Введение

Арифметическое выражение – это запись математической формулы с использованием констант, переменных, функций, знаков арифметических операций и круглых скобок. Формулы встречаются повсеместно в экономике, физике, химии, и, конечно же, в математике. Везде, где нужно символически представить какую-нибудь функцию или соотношение. Символическое представление, конечно, не единственное. И формула является самым компактным и универсальным способом.

Некоторые простые выражения мы можем посчитать самостоятельно и быстро. Но есть и такие, для вычисления которых потребуется очень много времени. Для этого и создаются программы, выполняющие вычисления арифметических выражений.

В отчете приводится постановка задачи вычисления арифметических выражений с использованием стека, описание алгоритмов вычислений, а также дается описание программы и правил ее использования, прилагается текст программы и результаты выполнения подсчетов.

Постановка задачи

Необходимо разработать программу, выполняющую вычисление арифметического выражения с вещественными числами. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны: +, -, /, *. Допускается наличие знака "-" в начале выражения или после открывающей скобки. Опционально - наличие математических функций (sin, cos, ln, exp, и т.д.) Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки и номера символов строки, в которых были найдены ошибки.

Руководство пользователя

- Пользователь вводит выражение по следующим правилам:
    a. Не допускается использование других символов, кроме «1234567890.,+-*/»
    b. Скобки должны быть расставлены по правилам математики
    c. Нельзя ставить знак операции после знака операции, т.е. «1+/2”», исключение составляет унарный минус
- Программа выдает пользователю результат в консоль
- Пользователь может продолжить вводить выражения и получать результат, пока не выйдет из программы.

Руководство программиста

Описание структуры программы

Алгоритм программы реализован следующим образом:

1. Пользователь вводит выражение, которое записывается в строку.
2. Производится проверка строки на соответствие правилам ввода.
3. Происходит перевод строки в вектор, в котором каждый элемент – это отдельная лексема.
4. Далее, программа переводит вектор в обратную польскую запись.
5. После перевода, производится подсчет выражения в обратной польской записи.
6. Результат выводится в консоль.

Описание алгоритмов

Перевод строки в вектор

Проходим по строке от начала до конца:

1. Если встречаем пробел, то переходим на следующую итерацию
2. Если встречаем плюс, умножить или разделить, то создаем объект, в который записываем нашу операцию. Вставляем этот объект в конец нашего вектора.
3. Если встречаем минус, то мы должны определить унарный он или нет:
    a. Если минус находится в начале строки, либо символ перед минусом — символ операции, но не закрывающаяся скобка, то минус унарный и мы создаем объект, в котором будем хранить “_”, такое решение было выбрано для разделения унарного и бинарного минусов. После мы вставляем в вектор наш объект.

   b. Если предыдущие условия не верны, то минус бинарный и мы можем просто сделать вставляем в конец вектора объект, в котором храним “-“.
4. Если встречается цифра или точка, то мы идем дальше по строке, пока цифры и точки еще встречаются (также реализована проверка на количество точек, в числе не должно быть больше одной точки, иначе выбрасывается исключение). После того, как мы полностью опознали число, мы создаем объект с таким значением, далее мы вставляем этот объект в вектор.
5. Если же встретилась открывающаяся или закрывающаяся скобка, то также создаем объект, содержащим эту скобку в виде операции и вставляем в вектор.

Перевод из вектора в обратную польскую запись

Рассматриваем поочередно каждый элемент вектора:

1. Создаем дополнительный вектор, в который будем складывать готовую польскую запись.
2. Идем в цикле по элементам вектора:
    a. Если этот элемент - число (или переменная), то просто помещаем его в выходную строку.
    b. Если символ - знак операции (+, -, *, /), то проверяем приоритет данной операции. Операции умножения и деления имеют наивысший приоритет (допустим он равен 3). Операции сложения и вычитания имеют меньший приоритет (равен 2). Наименьший приоритет (равен 1) имеет открывающая скобка.
    c. Получив один из этих символов, мы должны проверить стек:
        i. Если стек все еще пуст, или находящиеся в нем символы (а находится в нем могут только знаки операций и открывающая скобка) имеют меньший приоритет, чем приоритет текущего символа, то помещаем текущий символ в стек.
        ii. Если символ, находящийся на вершине стека имеет приоритет, больший или равный приоритету текущего символа, то извлекаем символы из стека в выходную строку до тех пор, пока выполняется это условие; затем переходим к пункту а).
    d. Если текущий символ - открывающая скобка, то помещаем ее в стек.
    e. Если текущий символ - закрывающая скобка, то извлекаем символы из стека в выходную строку до тех пор, пока не встретим в стеке открывающую скобку (т.е. символ с приоритетом, равным 1), которую следует просто уничтожить. Закрывающая скобка также уничтожается.
3. Если вся входная строка разобрана, а в стеке еще остаются знаки операций, извлекаем их из стека в выходную строку.

Подсчет результата обратной польской запись

Для подсчета нам понадобится стек.

1. В цикле проходим по вектору, в которой хранится обратная польская запись
2. Если элемент – число, то вставляем в стек это число
3. Если элемент – унарный минус, то берем из стека последнее значение, умножаем его на -1 и помещаем обратно в стек
4. Если все предыдущие условия не выполнились, то
    a. Берем из стека два операнда, не забыв, что сначала из стека возвращается второй операнд, а потом первый.
    b. Если текущая операция – плюс, то складываем операнды. Если бинарный минус – то вычитаем из первого операнда второй, если знак умножения, то умножаем операнды, если знак деления, то делим первый операнд на второй, не забыв проверить, что второй операнд не равен нулю.
    c. Помещаем в стек результат операции.
5. Ответом будет являться значение, которое останется в последнем объекте в стеке.

Проверка скобок в строке

1. Если после открывающейся скобки идет знак бинарной операции или число, то выбрасываем исключение.
2. Если перед открывающейся скобкой идет число, то выбрасываем исключение.
3. Для определения правильности подсчета скобок, мы при нахождении в строке открывающейся скобки инкрементируем счетчик, а при нахождении закрывающейся скобки декрементируем его. Если когда-нибудь счетчик опустится ниже нуля, то выбрасываем исключение.

Проверка строки на запрещенные символы

Для проверки нам понадобится строка, в которой хранятся все позволенные символы.

Проходя по всей строке, мы сравниваем каждый символ строки со всеми символами строки позволенных символов. Если хотя бы один символ исходной строки не найдется в строке дозволенных символов, то строка будет признана неверной.

Общая проверка остальных случаев

Если в строке на первом месте находится бинарная операция, то выбрасываем исключение.

Также, последним символом в строке является операция, но не закрывающаяся скобка, то выбрасываем исключение.

Заключение

Лабораторная работа на тему «Вычисление арифметических выражений» дала возможность с практической точки зрения узнать, что такое стек. Также были получены базовые знания в такой теме как «разбор выражения». Получена практика проектирования классов и повторения общих знаний объектно-ориентированного программирования. Другие люди могут воспользоваться этой программой, чтобы подсчитать результат довольно простых выражений. Работа программы протестирована с помощью Google Tests