Калимулин Н.Ф., Павлова А.С., Борисова А.Д., Петрова Н.К., Хамитов Р.М.
Кафедра информатики и информационных управляющих систем
Казанский государственный энергетический университет

РАЗРАБОТКА ПРИЛОЖЕНИЯ, ОБЕСПЕЧИВАЮЩЕГО РАСЧЁТ АРИФМЕТИЧЕСКОГО ВЫРАЖЕНИЯ, ЗАДАННОГО В ВИДЕ СИМВОЛЬНОЙ СТРОКИ

Анотация

В статье представлено описание разработанного алгоритма и соответствующей ему программы на С++, которая позволяет вычислять результат арифметического выражения, заданного в виде символьной строки. Строковое выражение включает в себя 4 арифметических действия с числами, двумя переменными x,y, инициализируемыми при запуске программы, а также скобки с разной степенью вложения. Программа способна определить приоритеты арифметических операций, включая анализ выражения в скобках, выделить многозначные числовые константы и именна переменных, входящих в выражение. Ограничения на длину выражения редактируются пользователем. Работа программы организована с применением двух стеков по принципу «последний пришел – первый ушел». С помощью методов происходит считывание, при анализе выражения определяется значимость символа, оценка приоритета для операции, подстановка числового значения для переменной, сборка многозначного числа. Программа написана на языке С++ и функционирует в операционной системе Windows.

Ключевые слова: арифметическое выражение, калькулятор, стековый калькулятор, арифметическое операции, приоритет операций.

Введение

Программирование задач с символами и строками на языке «С++» представляет собой достаточно интересную и непростую работу не только для студентов, но и для опытных программистов. Ведь символьная строка на Си – это, с одной стороны, массив символов, а с другой стороны, при условии использования объектов класса string, - обычная строка. Такая двойственность порой вызывает сложности в грамотной работе с символьными данными [1-2].

В рамках поставленного исследования – создать арифметический калькулятор на основе введённой строки символов – работа усложняется и нетривиальностью алгоритма. Для его реализации нами была рассмотрена идея применения польской нотации, что и позволило получить красивое и функциональное приложение. На примере приведенного в работе тестового решения детали алгоритма становятся понятными и позволяют выбрать и алгоритмические приёмы, и метод решения задачи. Блок-схемы ключевых программ позволяют оценить стратегию решения. Коды некоторых функций демонстрируют тактику в применении отдельных алгоритмических методов.

Суть решаемой задачи заключалась в следующем: строка содержит переменную X и Y, константы (целые и вещественные), операции (+, -, *, /, :) и скобки. Подпрограмма-функция возвращает значение арифметического выражения при заданном значении X и Y.

Разработка контрольного примера

Для понимания сути разрабатываемого приложения, формирования алгоритма и реалилизации тестирования получаемых программой результатов разработаем контрольный тестовый пример, позволяющий решить поставленную задачу [3].

Контрольный пример: S(x,y) = 1+10*(x*2+7)/3-y

Программа запрашивает у пользователя ввести «X» (X=4).

Программа запрашивает у пользователя ввести «Y» (Y=5).

Программа запрашивает у пользователя ввести выражение «S» 1+10*(x*2+7)/3-y

Примечание: Есть 2 стека, один предназначен для хранения чисел (stack <int> score) другой для операций (stack <char> operation) «-, +, *, /, (,)» «+», «-» имеют приоритет 1; «*», «/» имеют приоритет 2;

Через цикл (for) программа считывает выражение.

Программа встречает число «1» и помещает в стек чисел.

Программа встречает «+» и проверяет, что в стеке с операциями ничего нет, поэтому операцию «+» помещает в стек с операциями.

Программа встречает число «10» и помещает в стек чисел.

Программа встречает «*» и проверяет на прошлую добавленную операцию, то есть «+». У него приоритет 1, а у умножении 2, так что программа просто кладет «*» в стек с операциями.

Дальше программа встречает «открывающаяся скобки» и помещает в стек с операциями.

Программа встречает число «4» и помещает в стек чисел.

Программа встречает «*» и проверяет стек с операциями, там скобки, а скобки и плюс никак не взаимодействует, так что «*» программа кладет в стек.

Программа встречает «2» и помещает в стек чисел.

Программа встречает «-» и проверяет предыдущую операцию в стеке с операциями, а там «*», поэтому со стека с операциями извлекается «*», а со стека с числами извлекается последние два числа, то есть 4 и 2 и вместо этих чисел помещается 8 (4+2).

Программа встречает «6» и помещает в стек чисел.

Программа встречает «закрытую скобку», поэтому выполняет все операции находящиеся между скобками, то есть «+» => 8 + 7 = 15.

Программа считывает «/» и проверяет предыдущую операцию в стеке, там «*», поэтому сначала пройдет операция «умножение» то есть 15*10=150 – записывается в стек вместо «15 и 10», а в стек с операциями помещается «/»

Программа считывает «3» и помещает в стек чисел.

Программа считывает «-» и проверяет предыдущую операцию, там «/», поэтому сначала пройдет «/», то есть 150/3 = 50, а «-» в стек с операциями.

Программа считывает «5» и помещает в стек чисел.

Программа считывает следующий символ, которого уже нет. Получается программа делает все оставшиеся операции, которые остались в стеке, то есть «-» и «+», то есть 50 – 5 = 45 и затем 45 + 1 = 46, что и является результатом.

Для реализации описанного процесса в средствах языка C++ имеется достаточно эффективных методов, позволяющих решить эту задачу [4; 5].

Выбор и обоснование метода решения

На основе анализа контрольного примера, алгоритм главной программы будет состоять из следующих шагов:

1. Заполняем переменную X и Y.
2. Заполняем строковую переменную S, содержащее переменные X и Y.
3. Обозначаем приоритет операций *,/, :, +,-,(,).
4. Считываем строку, который разбит на некие сущности (числа и знаки) и выполняем следующие действия:
   a. Если это число (или переменная X, Y), то помещаем в стек чисел.
   b. Если это операция «*,/, +,-,(,)», то проверяем приоритет предыдущей операцией с последней добавленной. «+,-» - имеет приоритет 1. «*,/» - имеет приоритет 2.

c. Если приоритет предыдущей операции больше чем последней добавленной, то мы выполняем вычисления. Иначе кладем его в стек с операциями.
d. Если стек операций 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 операций.
e. Если символ, находящийся на вершине стека, имеет приоритет, больший или равный приоритету текущего символа, то извлекаем символы из стека операций операции до тех пор, пока выполняется это условие; затем переходим к пункту (а).
f. Если текущий символ - открывающая скобка, то помещаем ее в стек операций.
g. Если текущий символ - закрывающая скобка, то извлекаем символы из стека операций символы 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.
h) Если вся входная строка разобрана, а в стеке операций еще остаются знаки операций, извлекаем их из стека.
h. j) Если в примере ошибка, то в консоль выйдет «предупреждение об ошибке»

5. После выполнения всех операций в стеке чисел останется единственное число, что и будет являться ответом.
6. Выводим результат.

Стратегия решения ключевых функций [6] продемонстрирована на соответствующих блок-схемах (рис. 1- рис. 3).

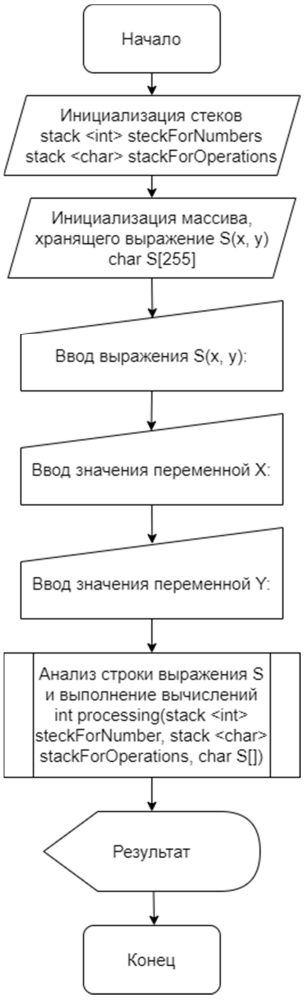

Рис. 1 Блок-схема главной программы

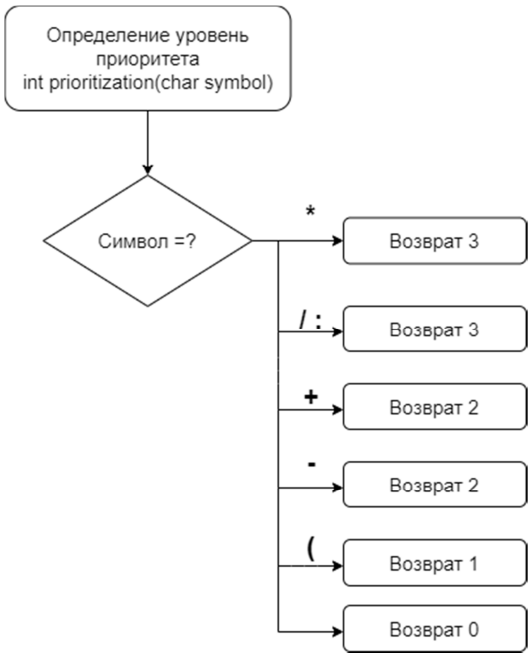

Рис. 2 Блок-схема функции определения приоритетов

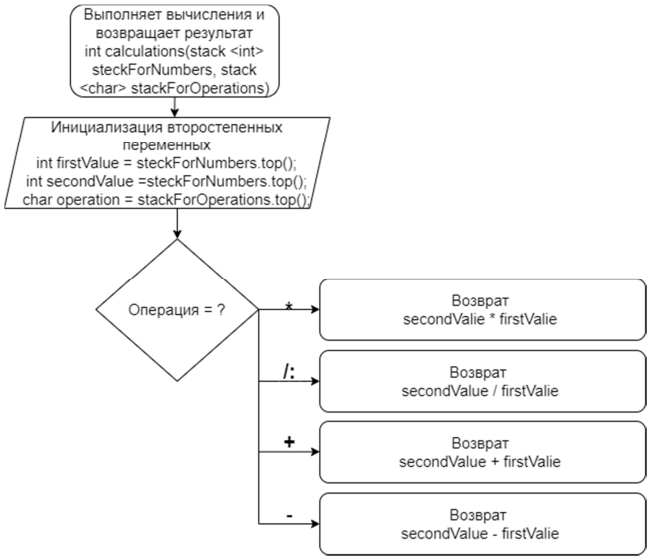

Рис. 3 Блок схема метода вычислений

Интересным решением для выделения числовых констант было применение кодовых адресов цифр и реализация метода перевода символа в число [7], суть которого понятна из демонстрации приведённого ниже фрагмента кода программы:

```
for (int i = 0; S[i] != '\0'; i++) { //перебирающий цикл всех элементов, \0 означает конец строки.
   int number = 0;
   if (S[i] >= '0' && S[i] <= '9') { //если элемент у массива число, то ...
      while (S[i] >= 0x30 && S[i] <= 0x39) { //выполняется алгоритм преобразования чисел
         number = number + (S[i] & 0x0F);
         number = number * 10;
         i++;
      }
      number = number / 10;
      steckForNumber.push(number); // после преобразования числа кладем в стек с числами
   }
   if (S[i] == 'x' || S[i] == 'X') { //если встречаем X, кладем в стек с числами
      steckForNumber.push(x);
```

Тестирование программ

Тестирование программы проведено путем сравнения результатов её работы программы с приведёнными контрольными примерами. Контрольный пример разработан с учётом проверки полноты программы – рассмотрены множество возможных исходов при работе с разными значениями переменных *x, y*.

Пользовательский интерфейс программы отображает необходимые шаги, которые пользователь/оператор должен предпринять для того, чтобы корректно решить поставленную задачу.

Тестирование 1. Дано арифметическое выражение, содержащее переменные X и Y и все операции. Результат его вычисления:

```
Введите выражение S, содержащее переменную X и Y:
2+5*88+(300/30-41*2)-102+y*3/3-x
Введите X:
14
Введите Y:
6
Результат:260
```

Рис.4. Тестирование с переменными X и Y

Тестирование 2. Дано арифметическое выражение с четырёхзначными числами и содержащую только Y. Результат его вычисления:

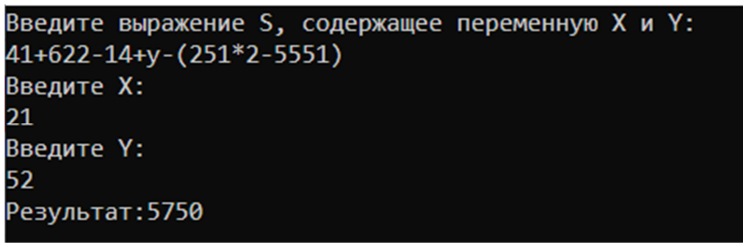

Рис. 5 Тестирование с одной переменной Y

Тестирование 3. Дано арифметическое выражение с двойными скобками и не содержащее переменные. Результат его вычисления:

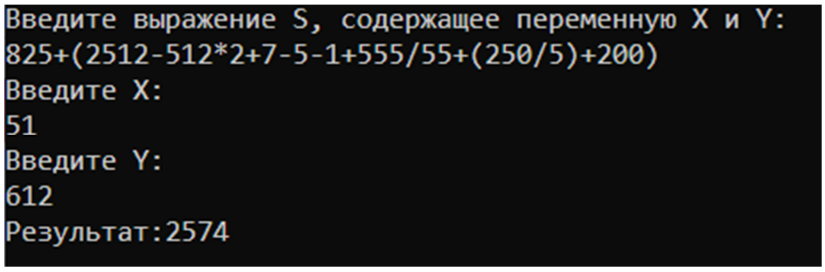

Рис.6 Тестирование с двумя скобками

Тестирование 4. Дано арифметическое выражение, содержащее только операции умножения и деления. Результат его вычисления:

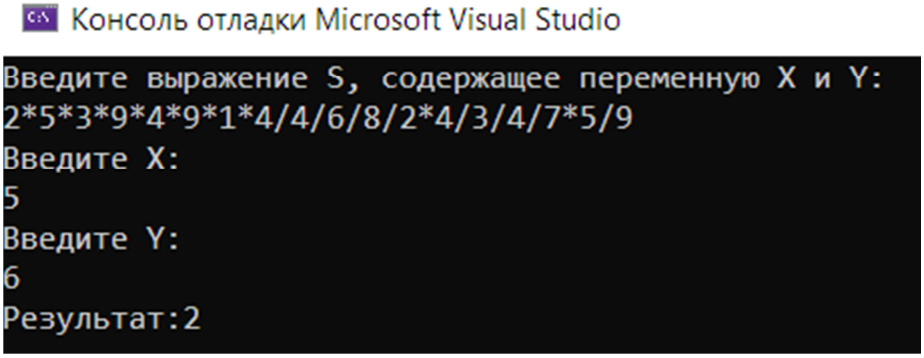

Рис.7 Тестирование с приоритетами операций 2

Заключение

В данной работе была создана программа, позволяющая расставлять приоритеты операций в выражениях и грамотно выполнять вычисления. В процессе выполнения работы были осуществлены следующие этапы исследования:

Разработка контрольного примера

Построение словесного алгоритма с учётом разбиения его на функциональные модули.

Формирование алгоритма работы для каждого модуля

Запись алгоритма на языке программирования

Тестирование программы показало, что все возможные исходы, заложенные в контрольном примере, реализуются правильно. Значит, программа работоспособна и может быть применена в массовых расчётах.

Литература

1. Петрова Н.К. Конспект лекций «Алгоритмизация и программирование на С++».[Электронный ресурс]- Режим доступа: https://lms.kgeu.ru/course/view.php?id=3751
2. Губаев Т.О., Петрова Н.К. Разработка синтаксического анализатора арифметических выражений на языке С++ - Вестник Казанского государственного энергетического университета. 2018. Т. 10. № 2 (38). С. 32-40.
3. Курсовая работа по дисциплине «Алгоритмизация и программирование»: методические указания.Петрова Н.К., Куценко С.М. – Казань: Казан. гос. энерг. ун-т, 2019. – 51 с.
4. Бьёрн Страуструп "Язык программирования С++».- Изд-во Бином. 2015 – 1166с.
5. Павловская Т.А. С/С++. Программирование на языке высокого уровня: учебник для вузов. - СПб.: Питер, 2013. - 461 с.
6. Ишмуратов Р.А., Ситников С.Ю. Применение визуальных сред разработки приложений для создания обучающих программ - Ученые записки ИСГЗ, Казань. - 2018. - Т.16. - №2, С.111-117.
7. Белавин В.А., Голицына И.Н., Куценко С.М. Эффективность использования моделирующих учебных систем в техническом вузе - Образовательные технологии и общество. 2000. Т. 3. № 2. С. 161-173.