МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский

государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: Программная инженерия

**Отчет по лабораторной работе №3**

**«Разбор арифметических выражений»**

**Выполнил**:студент группы 3821Б1ПР2

Карагодин Андрей Романович

**Проверила**:

ассистент кафедры МОСТ,

Усова М.А.

Нижний Новгород

2022

**Содержание**

[Введение 3](#_Toc117849017)

[Постановка задачи 4](#_Toc117849018)

[Руководство пользователя 5](#_Toc117849019)

[Руководство программиста 5](#_Toc117849020)

[Описание структуры программы 5](#_Toc117849021)

[Описание алгоритмов 6](#_Toc117849022)

[Алгоритм перевода в обратную польскую запись 6](#_Toc117849023)

[Алгоритм вычисления на стеке 7](#_Toc117849024)

[Заключение 9](#_Toc117849032)

[Литература 9](#_Toc117849033)

[Приложения 10](#_Toc117849034)

[Приложение 1. Функция бинарного поиска 10](#_Toc117849035)

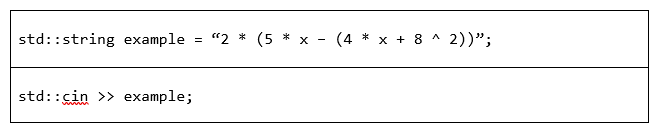
Введенние

В математике существует древняя традиция помещать оператор между операндами (x+y), а не после операндов (xy+). Форма с оператором между операндами называется инфиксной записью. Форма с оператором после операндов называется постфиксной, или обратной польской записью в честь польского логика Я. Лукасевича (1958), который изучал свойства этой записи.

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](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C#.D0.92.D1.8B.D1.87.D0.B8.D1.81.D0.BB.D0.B5.D0.BD.D0.B8.D1.8F_.D0.BD.D0.B0_.D1.81.D1.82.D0.B5.D0.BA.D0.B5). В-третьих, инфиксные операторы имеют приоритеты, которые произвольны и нежелательны. Например, мы знаем, что ab+c значит (ab)+c, а не a(b+c), поскольку произвольно было определено, что умножение имеет приоритет над сложением.

Постановка задачи

На вход программы поступает выражение, состоящее из односимвольных идентификаторов и знаков арифметических действий.



Требуется:

1. разобрать выражение на лексемы (парсер),

2. подготовить выражение к вычислению (обратная польская запись),

3. вычислить и вывести результат.

Руководство пользователя

Программа позволяет пользователю решать простые арифметические выражения. Доступный список операндов: +, -, \*, /, (, ), x.  
При запуске программы в консоли выводится текст с возможными операндами и примером выражения. Затем программа попросит ввести выражение для вычисления. Выражение нужно вводить без пробелов.

После ввода выражения, программа попросит ввести значение X. Если в выражении присутствует символ(ы) x, он(и) заменится на значение, введенное пользователем на данном шаге. Если же выражение не имело значения X, то введенное пользователем значение на этом шаге будет игнорироваться.

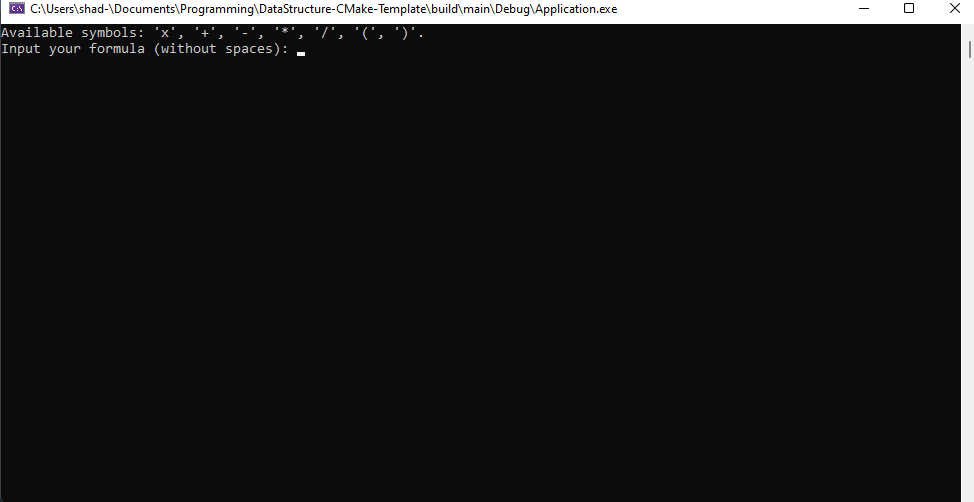


Рис. 1 Старт программы

Руководство программиста

Библиотека lib\_polish состоит из трёх главных классов: Parser, PolishConvertor, Calculator. Каждый из них выполняет свою функцию в вычислении выражения.

Описание структуры программы

Класс Calculator имеет 1 главную доступную пользователю функцию – calculate. Функция использует объекты и функции классов Parse и PolishConvertor для вычисления в своем алгоритме.

Описание функции:

float calculate(string expression, int \_x)

Функция принимает на вход выражение в виде string, заданное пользователем значение X и производит расчеты, возвращая результат типа float.

**Параметры**

expression — строка заданного пользователем выражения,

\_X – заданное X значение пользователем.

**Исключения**

Может вызывать std::logic\_error при наличии ошибки в выражении (наличие лишних символов, несоответствие количеству скобок).

Описание алгоритмов

Алгоритм перевода в обратную польскую запись

**Основная идея алгоритма.** Предположим, что формула состоит из переменных, двухоперандных операторов +,-,\*,/, а также левой и правой скобок. У каждой операции есть приоритет. Проходя по списку лексем, составленных и переданных из Parser, мы будем принимать каждый токен по очереди. Если на вход подается число – записываем в стек число. Если подается операнд или скобка – действуем в соответствии с подробностями алгоритма и приоритета действий.

**Описание алгоритма.** На вход поступает строка выражения и значение X. Парсер преобразует строку string в лист лексем с которыми далее будет работать Polish Convertor.  
Polish Convertor преобразует из обычного списка лексем в список лексем в порядке польской записи. Работа алгоритма:

Из списка подается лексема (токен) и проходит проверку:

1. Если число – добавить в выходной список.
2. Если операнд +, -, \*, \, то после проверки стека на пустоту идет сравнение:

* Если стек операндов пуст, то записываем операнд в стек.
* Если стек операндов не пуст, полученный токен имеет приоритет ниже или такой же как и последний операнд в стеке, то стек возвращает значение последнего записанного операнда в выходной список, пока приоритет токена не будет выше последнего операнда в стеке.

1. Если скобка ‘(‘, то независимо от последнего операнда, она записывается в стек.
2. Если скобка ‘)’, то начинается цикл выталкивания всех операндов из стека, пока не найдется ‘(‘. Затем ‘(‘ затирается.
3. Если стек операторов по завершению цикла остается не пустым, то все операторы из списка добавляются в конец выходного списка.

На рисунке 2 приведен схематический пример работы данного алгоритма.

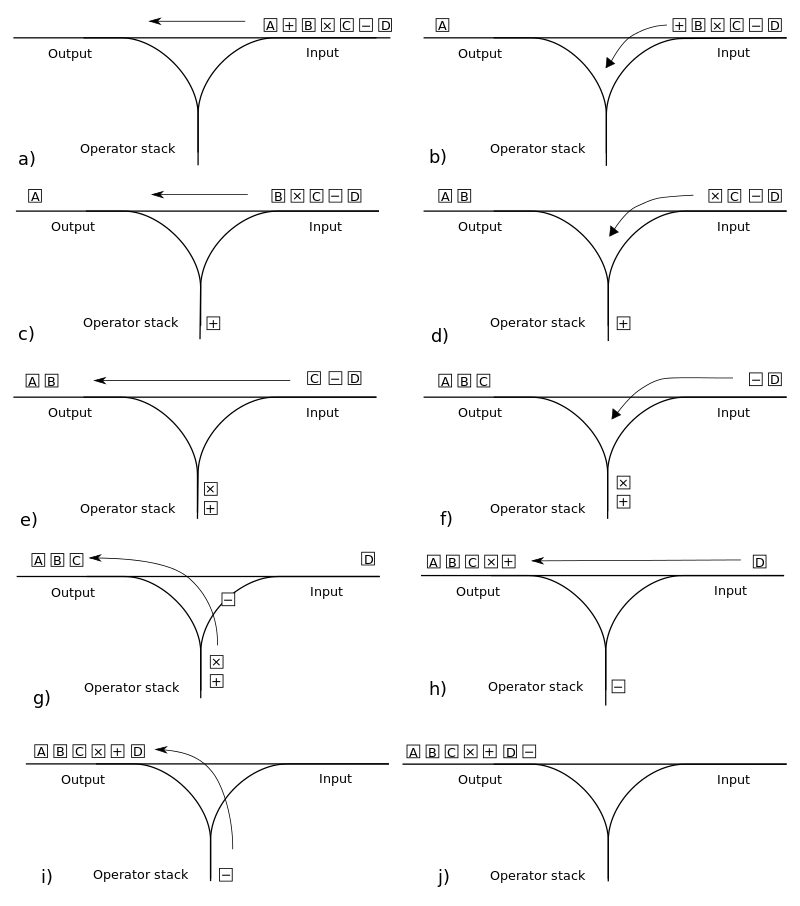


Рис 2. Алгоритм Shunting Yard для преобразования в обратную польскую запись.

Алгоритм вычисления на стеке.

После преобразования в обратную польскую запись, алгоритм считывает каждый токен из списка лексем и добавляет их в стек. Если встречается операнд, то из стека берется 2 последних числа и производится операция в соответствии с операндом. Результат добавляется в float res и сам стек для дальнейших вычислений. По завершению функции возвращается число типа float – результат.



Рис 3. Схема работы вычислений на стеке

**Замечание.** Алгоритм должен и будет работать правильно только если проверка на правильность выражения Parser прошла успешно.

**Пример работы алгоритма.** На вход подается выражение 1+2\*3+4.

**1 шаг.** Парсер выполняет проверку выражения на корректность. Неправильное количество скобок или символы не входящие в набор простых чисел, операндов, скобок или знака ‘x’ выдадут ошибку и программа не выдаст правильного результата. В данном случае ошибок не наблюдается.

**2 шаг.** Парсер создает список лексем, проходя по каждому символу строки и преобразуя его либо в число, либо сохраняя символ операнда, скобки или знака x в список. Затем функция parse сохраняет список лексем в объекте класса Parser. Список лексем будет выглядеть в списке как: 1 -> + -> 2 -> \* -> 3 -> + -> 4.

**3 шаг.** Объект класса PolishConvertor при помощи функции convert преобразует из списка лексем, полученных функцией getLexemList от объекта Parser, в список лексем в порядке обратной польской записи. Алгоритм приведения Shunting Yard приведен в описании выше. После преобразования к обратной польской записи, функция convert возвращает список лексем, с которым будет работать функция calculate. Преобразовав данную строку в консоль будет выведена обратная польская запись данного примера: 1 2 3 \* + 4 +.

**4 шаг.** Далее в функции calculate из списка лексем обратной польской записи происходит проверка на число или операнд в соответствии с алгоритмом вычисления на стеке. После последовательного выполнения алгоритма, результат функции, хранящийся в переменной float res, возвращает значение пользователю. Результат в данном примере будет: 11.

Заключение

В результате выполнения лабораторной работы я познакомился с алгоритмами вычисления на стеке и преобразования из инфиксной формулы в постфиксную нотацию (обратную польскую запись). Используя данные алгоритмы, мне удалось создать калькулятор, позволяющий вычислить арифметические выражения.

Литература

1. Habr - Обратная польская запись - <https://habr.com/ru/post/100869/>
2. Википедия – Обратная польская запись (рус.) - [Обратная польская запись — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C)
3. Википедия – Shunting Yard Algorithm (англ.) - [Shunting yard algorithm - Wikipedia](https://en.wikipedia.org/wiki/Shunting_yard_algorithm)
4. Фоксфорд – Обратная польская нотация - https://foxford.ru/wiki/informatika/obratnaya-polskaya-notatsiya

Приложения

Приложение 1. Функция конвертации в обратную польскую запись

CList<string> convert(CList<string> list) {

string token;

string opbuff;

int len = list.getSize();

for (int i = 0; i < len; i++) {

token = list.pop\_back();

if (isNumber(token)) {

output.push\_front(token);

token = "";

}

for (int j = 0; j < 2; j++) {

if (token == op2[j]) {

if (!operatorstack.isEmpty()) {

opbuff = operatorstack.gettop();

while (opbuff == op3[j] || opbuff == op2[j]) {

output.push\_front(operatorstack.pop());

if (!operatorstack.isEmpty()) {

opbuff = operatorstack.gettop();

}

else {

opbuff = "";

}

}

}

operatorstack.push(token);

token = "";

}

if (token == op3[j]) {

if (!operatorstack.isEmpty()) {

opbuff = operatorstack.gettop();

while (opbuff == op3[0] || opbuff == op3[1]) {

output.push\_front(operatorstack.pop());

if (!operatorstack.isEmpty()) {

opbuff = operatorstack.gettop();

}

else {

opbuff = "";

}

}

}

operatorstack.push(token);

token = "";

}

}

if (token == parenthis[0]) {

operatorstack.push(token);

}

if (token == parenthis[1]) {

opbuff = operatorstack.gettop();

while (opbuff != parenthis[0]) {

output.push\_front(operatorstack.pop());

opbuff = operatorstack.gettop();

}

operatorstack.pop();

token = "";

}

}

if (!operatorstack.isEmpty()) {

while (!operatorstack.isEmpty()) {

output.push\_front(operatorstack.pop());

}

}

CList<string> copy;

copy.copy(output);

cout << "Reverse Polish notation: ";

while (!copy.isEmpty()) {

cout << copy.pop\_back() << " ";

}

return output;

}

Приложение 2. Функция вычисления из класса Calculate

float calculate(string expression, int \_x) {

int x = \_x;

CList<string> explist;

Stack<string> stlist;

p.parse(expression, x);

explist = c.convert(p.getLexemList());

int len = explist.getSize();

string buff;

float res = 0;

for (int i = 0; i < len; i++) {

buff = explist.pop\_back();

if (isdigit(buff[0])) {

stlist.push(buff);

}

else {

if (buff == "+") {

string num1 = stlist.pop();

string num2 = stlist.pop();

res = add(num1, num2);

stlist.push(to\_string(res));

}

if (buff == "-") {

string num2 = stlist.pop();

string num1 = stlist.pop();

res = subs(num1, num2);

stlist.push(to\_string(res));

}

if (buff == "\*") {

string num1 = stlist.pop();

string num2 = stlist.pop();

res = mult(num1, num2);

stlist.push(to\_string(res));

}

if (buff == "/") {

string num2 = stlist.pop();

string num1 = stlist.pop();

res = div(num1, num2);

stlist.push(to\_string(res));

}

}

}

return res;

}