Тема: наследование и полиморфизм подтипов

Вариант: 2.1.1

Задача: Реализовать иерархию классов для работы с математическими выражениями.

Реализовать абстрактный класс выражений `Expression`, являющийся корнем иерархии.

На базе класса `Expression` реализовать производные классы `Number`, `Variable`, `Add`, `Sub`, `Mul`, `Div`, соответствующие различным выражениям: константе; переменной; сумме, разности, произведению и частному двух других выражений.

Пример:

```
Expression* e = new Add(
                    new Number(3),
                    new Mul(
                        new Number(2),
                        new Variable("x")));
```

Здесь e - это указатель на выражение вида `(3+(2*x))`

Для указанной иерархии реализовать следующую функциональность:

1) Создание выражения по заданной в файле/консоли строке. Считать, что каждое выражение кроме переменных и констант всегда пишется в скобках.

2) Печать выражения в файл/на экран по аналогичным правилам.

   Пример (здесь и ниже *e* - выражение из примера выше):

```
e->print(); // (3+(2*x))
```

3) Символьное дифференцирование выражений по заданной переменной.

```
Expression* de = e->derivative("x");
de->print(); // (0+((0*x)+(2*1)))
```

   При этом изначальное выражение не меняется, но создается новое соответствующее результату дифференцирования.

   Обратите внимание, что названия переменных могут быть многобуквенными.

4) Вычисление значения выражения при задании означивания всех переменных:

```
int res = e->eval("x <- 10; y <- 13");
// res = (3+(2*10)) = 23
```

   Функция eval в качестве аргумента принимает строку, в которой через разделитель ; указаны означивания переменных.

5) Корректное управление динамической памятью для всех классов иерархии. В классах не должно быть утечек памяти, некорректных указателей и т. д.

В качестве демонстрационного примера необходимо:

1. Считать из входного файла выражение по описанным выше правилам.

2. Символьно продифференцировать его по переменной `"x"`
3. Записать получившееся выражение в выходной файл.

Входные данные:

Во входном файле находится строка с выражением. Каждое подвыражение кроме констант и переменных заключено в скобки, пробелов в выражении нет.

Выходные данные:

Выражение, являющееся результатом символьного дифференцирования по переменной `"x"`. Правила записи выражения аналогичны.

Пример входных и выходных данных:

input.txt	output.txt
(((2*x)+(x*x))-3)	((((0*x)+(2*1))+((1*x)+(x*1)))-0)

Дополнительные задания:

Реализовать:

1. Считывание выражения со строки, работающее со строкой без скобок вокруг каждого выражения.

2. Функцию для упрощения выражений по следующим правилам:
    a. Если выражение можно вычислить без какого-либо означивания (т.е. в нем нет переменных), то выражение заменяется на результат вычисления.

b. Если выражение представляет из себя умножение на 0, то оно заменяется на константу 0.
c. Если выражение представляет из себя умножение на 1, то оно заменяется на второй множитель.
d. Если выражение представляет из себя вычитание двух одинаковых подвыражений, то оно заменяется на константу 0.

Упрощение реализовать, как функцию, создающую новое (упрощенное) выражение по заданному выражению.