Задача

В рамках тестового задания предлагается реализовать интерпретатор примитивного функционального языка программирования.

Общие требования:

- задание должно быть выполнено на языке C++ **с использованием ООП и STL**
- использование сторонних библиотек не разрешено
- приложение должно собираться под Linux
- сборка должна быть организована через CMake
- исходный код задания нужно опубликовать на GitHub

Рекомендации:

- не пытайтесь гнаться за эффективностью или производительностью вашего решения в ущерб читаемости кода. помните, что преждевременная оптимизация - это корень всех зол
- назначайте функциям и переменным осмысленные имена, соблюдайте отступы
- избегайте дублирования кода
- избегайте больших или очень сложных функций - выполняйте разумную декомпозицию
- разделите проект на несколько файлов - не нужно размещать весь код в main.cpp
- не стоит игнорировать возможности C++11

CLI-интерпретатор

Реализуйте интерпретатор примитивного функционального языка программирования. Интерпретатор должен иметь несколько встроенных функций:

- add - вычислить сумму всех аргументов
- sub - вычислить разность аргументов
- mult - вычислить произведение всех аргументов
- concat - склеить все аргументы в одну строку

Синтаксис для вызова функции имеет вид

```
functionName arg1 arg2 ... argN
```

Аргументом может быть:

- строка без пробелов, состоящая только из символов [-][a-z][A-Z][0-9], например 123, -33, abc, -abc
- строка, заключённая в двойные кавычки '"', и состоящая из любых символов кроме самих двойных кавычек (экранирование как в C не требуется). например: "123", "2 * 2 = ", "(by the way)"
- вызов другой функции, заключённый в круглые сборки; например: (add 5 7)

Интерпретатор должен постоянно ожидать ввода выражения с клавиатуры, вычислять его и выводить результат. Если в качестве очередного выражения пользователь ввёл "quit", то программа должна завершить работу.

Примеры работы с интерпретатором в режиме CLI

```
$ add 5 7 3
> 15

$ sub -7 "3"
> -10

$ mult (add 5 3) (sub 5 3)
> 16

$ concat "(2 + 2) * 2 = " (mult 2 (add 2 2))
> (2 + 2) * 2 = 8

$ quit
```

Допущения и рекомендации:

- ради упрощения, будем считать, что программе будут подаваться только корректные выражения, поэтому не обязательно реализовывать обработку синтаксических ошибок (но если очень хочется, то можно!)
- постарайтесь построить приложение таким образом, чтобы добавление новых функций требовало минимум изменений в коде.