Лабораторная работа № 2. Основы программирования на Рефале

Коновалов А.В.

18 февраля 2023 г.

1 Цель работы

Целью данной работы является ознакомление с языком программирования Рефал-5 и библиотекой LibraryEx из refal-5-framework.

2 Исходные данные

Библиотека LibraryEx доступна в git-репозитории

https://github.com/mazdaywik/refal-5-framework

Кроме того, эта библиотека автоматически устанавливается вместе с Рефалом-5λ.

3 Задание

3.1 Утилита `grep`

Требуется написать аналог POSIX-утилиты `grep`, осуществляющей поиск подстроки в файле. Синтаксис командной строки:

```
refgo grep ‹искомая строка› ‹файл›…
```

или

```
refgo grep+LibraryEx ‹искомая строка› ‹файл›…
```

Если имена файлов не указаны, должен читаться стандартный ввод.

Если указано 2 файла и более, в начале каждой строки должно распечатываться имя файла, в котором найдена искомая подстрока.

Ключи командной строки реализовывать не нужно.

3.2 Декомпилятор обратной польской записи

Функция принимает выражение в обратной польской записи и строит традиционное инфиксное выражение:

```
<PostfixDecompile s.Operation+> == t.Expr
s.BinOp ::= "+" | "-" | "*" | "/"
s.VarName ::= s.WORD \ s.BinOp
s.Operation ::= s.NUMBER | s.VarName | s.BinOp
t.Expr ::= s.VarName | s.NUMBER | (t.Expr s.BinOp t.Expr)
```

Пример работы:

```
<PostfixDecompile 5 X "-" Y 4 "+" "*"> == ((5 "-" X) "*" (Y "+" 4))
```

Тестовая программа должна загружать исходные данные из файла при помощи функции `LoadExpr` из библиотеки `LibraryEx`. Имя файла задаётся в командной строке.

3.3 Функция символьного дифференцирования

Функция должна иметь следующий тип:

```
<Diff t.Expr s.VarName> == t.Expr

s.VarName ::= s.WORD
t.Expr ::=
    s.VarName
  | s.NUMBER
  | ("-" t.Expr)
  | (t.Expr s.BinOp t.Expr)
s.BinOp ::= "+" | "-" | "*" | "/"
```

Функция принимает выражение, производную которого надо построить и имя переменной, по которой нужно построить производную. Например:

```
<Diff ((X "*" Y) "+" ((2 "*" X) "+" (3 "*" Y))) X> == (Y "+" 2)
<Diff ((X "*" Y) "+" ((2 "*" X) "+" (3 "*" Y))) Y> == (X "+" 3)
```

Функция не должна оставлять выражения, которые можно упростить: сложение с нулём, умножение на ноль или единицу и т.д.

Тестовая программа должна загружать исходные данные из файла при помощи функции `LoadExpr` из библиотеки `LibraryEx`. Имя файла задаётся в командной строке.

Язык реализации: Markdown

Код решения

Из файла

Отправить