# Лабораторная работа № 2 «Основы программирования на Рефале»

3 апреля 2024 г.

Сергей Виленский, ИУ9-62Б

## Цель работы

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

## Индивидуальный вариант

#### UNIX-утилита

Программа tail, распечатывающая первые несколько строк каждого файла.

`tail [-‹число›] ‹имена файлов›…`

Если количество строк не указано, подразумевается 10.

При реализации программы на Рефале-5 ключ командной строки должен начинаться на знак «+», т.к. Arg аргументы, начинающиеся на «-», игнорирует.

#### Символьная функция

Регулярное выражение описано следующим абстрактным синтаксисом:

`RegEx → ▫ | ε | SYMBOL | RegEx ▫ RegEx | RegEx · RegEx | RegEx*`

Здесь ▫ — пустое множество, ▫ — объединение, · — конкатенация.

Написать функцию `<First t.RegEx> == { s.CHAR | EPS }*`, вычисляющую множество FIRST для регулярного выражения.

## Скрипт исполнения ref-файлов `run.sh`

```
rlmake $1 -o a.exe
./a.exe ${*:2}
rm a.exe
```

## Реализация UNIX-утилиты

```
*$FROM LibraryEx
$EXTERN LoadFile, ArgList, Map, DelAccumulator;

$ENTRY Go {
    /* пусто */ = <Main <ArgList>>;
}

Main {
    (e.ProgPath) ('-' e.LinesCount) e.Files
        , <atoi e.LinesCount> : s.LinesCount
        , e.Files : {
            /* пусто */ = (<LoadStdin>);
        e._ = <Map {(e.File) = ((e.File) <LoadFile e.File>);} e.Files>;
        } : e.FilesLines
        , e.FilesLines : {
            t.FirstFile t.SecondFile e.Other = True;
            e._ = False;
        } : s.IsMoreThanOneFile
        =
            <Map {
                ((e.File) e.FileLines)
                    , s.IsMoreThanOneFile : {
                        True = <Prout '==> ' e.File ' <==='>;
                        False = ;
                    } : e._
                  = <Map {(e.Any) = <Prout e.Any>} <DelAccumulator
                        <Last s.LinesCount e.FileLines>>>;
            } e.FilesLines>;

    (e.ProgPath) e.Files = <Main (e.ProgPath) ('-10') e.Files>;
}

LoadStdin {
    , <Card> : {
        e.Line 0 = (e.Line);
        e.Line = (e.Line) <LoadStdin>
    };
}

ctoi {
    s.Char
        , <Sub <Ord s.Char> <Ord '0'>> : s.Digit
        , <Compare s.Digit <Sub 0 1>> <Compare s.Digit 10> : {
            '+-' = s.Digit;
```

```
        };
}

atoi {
   e.Other s.PreLast s.Last = <Add <Mul <atoi e.Other s.PreLast> 10> <ctoi s.Last>>;
    s.Once                  = <ctoi s.Once>;
}
```

## Тестирование UNIX-утилиты

```
$ sh run.sh lab2/tail.ref -2 lab2/tail.ref lab2/tail.sh
*Compiling lab2/tail.ref:
+Linking C:/.../Refal-5-lambda/lib/references\Library.rasl
+Linking C:/.../Refal-5-lambda/lib/slim\exe\LibraryEx.rasl
** Compilation succeeded **
==> lab2/tail.ref <==
    s.Once                  = <ctoi s.Once>;
}
==> lab2/tail.sh <==
./a.exe $*
rm a.exe

$ sh run.sh lab2/tail.ref lab2/tail.ref lab2/tail.sh
*Compiling lab2/tail.ref:
+Linking C:/.../Refal-5-lambda/lib/references\Library.rasl
+Linking C:/.../Refal-5-lambda/lib/slim\exe\LibraryEx.rasl
** Compilation succeeded **
==> lab2/tail.ref <==
        , <Sub <Ord s.Char> <Ord '0'>> : s.Digit
        , <Compare s.Digit <Sub 0 1>> <Compare s.Digit 10> : {
            '+-' = s.Digit;
        };
}

atoi {
   e.Other s.PreLast s.Last = <Add <Mul <atoi e.Other s.PreLast> 10> <ctoi s.Last>>;
    s.Once                  = <ctoi s.Once>;
}
==> lab2/tail.sh <==
rlmake tail.ref -o a.exe
./a.exe $*
rm a.exe
```

## Реализация символьных преобразований

```
/**
    Лексическая грамматика языка регулярок
        в порядке возрастания приоритета доменов:
    t.RegEx ::=
         ()
       | "ε"
       | s.CHAR
       | (t.RegEx)
       | (t.RegEx '*')
       | (t.RegEx "·" t.RegEx)
       | (t.RegEx "␣" t.RegEx)
*/

$ENTRY Go {
    = <Prout <First ((('a' "␣" ('a' "·" 'c')) '*') "·" 'b')>>;
}

/**
  <IsIn s.CHAR (s.CHAR*)> == True | False
*/
IsIn {
    e.Element (e.Left e.Element e.Right) = True;
    e.Else = False;
}

/**
  <AddToSet s.CHAR (s.CHAR*)> == s.CHAR*
*/
AddToSet {
    "ε" (e.SetLeft "ε" e.SetRight) =
        e.SetLeft "ε" e.SetRight;
    "ε" (e.Set) = "ε" e.Set;

    s.Element (e.SetLeft s.Element e.SetRight) =
        e.SetLeft s.Element e.SetRight;
    s.Element (e.Set) = s.Element e.Set;
}

/**
  <UnionSets (s.CHAR*) (s.CHAR*)> == s.CHAR*
*/
UnionSets {
    (e.Set) () = e.Set;
    () (e.Set) = e.Set;
```

```
   (e.Set1Left s.Set2First e.Set1Right) (s.Set2First e.Set2Other) =
     <UnionSets (e.Set1Left s.Set2First e.Set1Right) (e.Set2Other)>;
    (e.Set1) (s.Set2First e.Set2Other) =
        <UnionSets (e.Set1 s.Set2First) (e.Set2Other)>;
}

/**
  <First t.RegEx> == s.CHAR*
*/
First {
    "▫" = ;
    "ε" = "ε";
    s.RegEx =  s.RegEx;
    (t.RegEx '*') = <AddToSet "ε" (<First t.RegEx>)>;

    (t.RegExL "▫" t.RegExR) =
        <UnionSets (<First t.RegExL>) (<First t.RegExR>)>;
    (t.RegExL "·" t.RegExR)
        , <First t.RegExL> : e.FirstL
        , <IsIn "ε" (e.FirstL)> : {
            True  = <UnionSets (e.FirstL) (<First t.RegExR>)>;
            False = e.FirstL;
        };
}
```

## Тестирование символьных преобразований

```
$ sh run.sh lab2/First.ref
*Compiling lab2/First.ref:
+Linking C:/.../Refal-5-lambda/lib/references\Library.rasl
** Compilation succeeded **
εab
```

## Вывод

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