

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)
Laboratorinių darbų ataskaita

Atliko:

IFF-6/11 gr. studentas

Nerijus Dulkė

2019 m. balandžio 18 d.

Priėmė:

lekt. **Evaldas Guogis**

TURINYS

1. Python (L1)	3
1.1. Darbo užduotis	3
1.2. Programos tekstas	3
1.3. Pradiniai duomenys ir rezultatai	7
2. Scala (L2)	7
3. F# (L3)	7
3.1. Darbo užduotis	7
3.2. Programos tekstas	8
3.3. Pradiniai duomenys ir rezultatai	9
4. Prolog (L4)	9
4.1. Darbo užduotis	9
4.2. Programos tekstas	9
4.3. Pradiniai duomenys ir rezultatai	10

1. Python (L1)

1.1. Darbo užduotis

Nuoroda į užduotį:

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=9&page=show_problem&problem=725

Trumpas aprašymas:

Labirintas sudarytas iš stačiakampių kambarių pavaizduotas plokštumoje, naudojant simbolius. Užduoties tikslas yra pažymėti kambarius, kuriuos galima aplankyti iš nurodytos startinės pozicijos.

```
XXXXXXXXXXXXXXXXXXXXX
X  X  X  X  X  X
X      X  X  X
X  X  X  X  X  X
XXXXXX XXX XXXXXXXXX
X  X  X  X  X  X
X  X      *      X
X  X  X  X  X  X
XXXXXXXXXXXXXXXXXXXXX
```

a) Initial maze

```
XXXXXXXXXXXXXXXXXXXXX
X###X###X###X  X  X
X#####X      X  X
X###X###X###X  X  X
XXXXXX###X#####X
X  X###X###X###X
X  X#####X
X  X###X###X###X
XXXXXXXXXXXXXXXXXXXXX
```

b) Painted maze

1.2. Programos tekstas

Nerijus.Dulke.IFF.6.11.Lab.1.py

```
# IFF-6/11 Nerijus Dulke Lab1
#
https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=9&page=show_problem&problem=725

from Maze import Maze

mazes = []

duom = open('duom.txt')

mazeCount = int(duom.readline())

i = 0
iterations = 0
while i < mazeCount:
    maze = Maze()
    line = duom.readline()

    while line and not line.startswith('_'):
        maze.addline(line)
        iterations = iterations + 1
        line = duom.readline()

    mazes.append(maze)
    i = i + 1

duom.close()

rez = open('rez.txt', 'w+')
```

```

for maze in mazes:
    maze.paint()
    maze.printresult(rez)
    rez.write('____\n')

rez.close()

```

Maze.py

```

WALL = 'X'
EMPTY = ' '
MARKED = '#'
START = '*'
directions = ['u', 'd', 'l', 'r', 'ul', 'ur', 'dl', 'dr']

class Maze:
    def __init__(self):
        self.lines = []
        self.startX = -1
        self.startY = -1
        self.maxX = -1
        self.maxY = -1

    def addline(self, line):
        self.lines.append(line)

        if self.startX == -1:
            index = line.find(START)
            if index != -1:
                self.startX = index
                self.startY = self.count() - 1

        self.maxY = self.count() - 1
        maxX = len(line) - 1
        if maxX > self.maxX:
            self.maxX = maxX
        return

    def printlines(self):
        for line in self.lines:
            print line
        return

    def count(self):
        return len(self.lines)

    def printresult(self, file):
        file.writelines(self.lines)
        return

    def getvalue(self, coord):
        return self.lines[coord.y][coord.x]

```

```

def markvalue(self, coord):
    if self.getvalue(coord) is EMPTY:
        self.lines[coord.y] = self.lines[coord.y][:coord.x] + MARKED +
self.lines[coord.y][coord.x + 1:]
        return True
    return False

def paint(self):
    self.max = Coord(self.maxX, self.maxY)
    current = Coord(self.startX, self.startY)
    coordsToSearch = [current]
    visited = []

    while len(coordsToSearch) > 0:
        current = Coord(coordsToSearch[0].x, coordsToSearch[0].y)
        del coordsToSearch[0]

        for direction in directions:
            neighbour = self.getNeighbour(current, direction)
            isVisited = filter(lambda x: neighbour.equals(x), visited)

            if neighbour is None or len(isVisited) > 0:
                continue

            success = self.markvalue(neighbour)
            if success:
                coordsToSearch.append(neighbour)

        visited.append(current)
    return

def getNeighbour(self, current, direction):
    coord = Coord(current.x, current.y)
    if not coord.canmove(direction, self.max):
        return None

    if direction is 'u':
        coord.up()
    elif direction is 'd':
        coord.down()
    elif direction is 'l':
        coord.left()
    elif direction is 'r':
        coord.right()
    elif direction is 'ul':
        coord.up()
        coord.left()
    elif direction is 'ur':
        coord.up()
        coord.right()
    elif direction is 'dl':
        coord.down()
        coord.left()

```

```

        elif direction is 'dr':
            coord.down()
            coord.right()

        return coord

class Coord:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def canmove(self, direction, maxcoord):
        if direction is 'u':
            return self.y != 0
        elif direction is 'd':
            return self.y != maxcoord.y
        elif direction is 'l':
            return self.x != 0
        elif direction is 'r':
            return self.x != maxcoord.x
        elif direction is 'ul':
            return self.y != 0 and self.x != 0
        elif direction is 'ur':
            return self.y != 0 and self.x != maxcoord.x
        elif direction is 'dl':
            return self.y != maxcoord.y and self.x != 0
        elif direction is 'dr':
            return self.y != maxcoord.y and self.x != maxcoord.x
        return False

    def up(self):
        self.y = self.y - 1

    def down(self):
        self.y = self.y + 1

    def left(self):
        self.x = self.x - 1

    def right(self):
        self.x = self.x + 1

    def equals(self, other):
        return self.x == other.x and self.y == other.y

```

1.3. Pradiniai duomenys ir rezultatai

duom.txt

```
2
XXXXXXXXXX
X  X  X
X *   X
X  X  X
XXXXXXXXXX
X  X
X  X
X  X
XXXXXX
```

```
XXXXX
X  X
X * X
X  X
XXXXX
```

rez.txt

```
XXXXXXXXXX
X###X###X
X#*#####X
X###X###X
XXXXXXXXXX
X  X
X  X
X  X
XXXXXX
```

```
XXXXX
X###X
X#*#X
X###X
XXXXX
```

2. Scala (L2)

3. F# (L3)

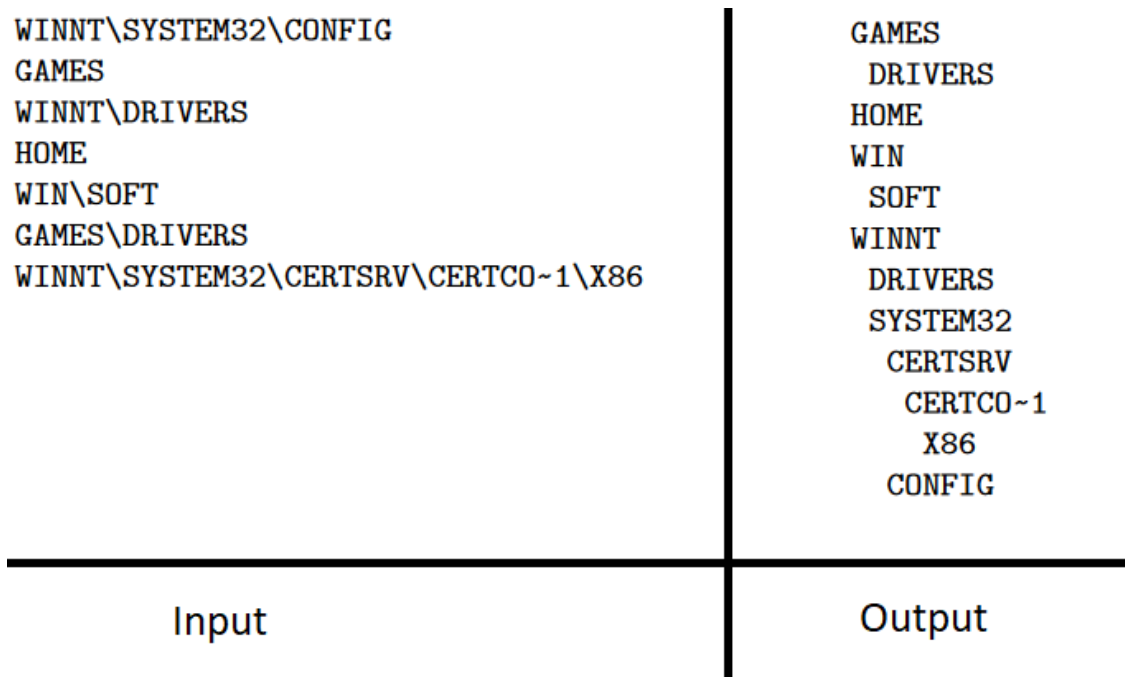
3.1. Darbo užduotis

Nuoroda į užduotį:

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=448&page=show_problem&problem=4331

Trumpas aprašymas:

Duotos nuorodos į aplankus esančius failų sistemoje, sudaryti failų sistemos medį.



3.2. Programas tekstas

```
//
https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&category=448&page=show\_p
roblem&problem=4331
```

```
open System
open System.IO
```

```
let readLines filePath = File.ReadLines(filePath)
```

```
let print (line: string) = printf "%s\n" line
```

```
let startsWithCount (path: string, lines: seq<string>) =
    lines
    |> Seq.filter(fun (y: string) -> y.StartsWith(path))
    |> Seq.length
```

```
let replaceFirst (text: string, search: string, replace: string) =
    let pos = text.IndexOf(search)
    if pos < 0 then
        text
    else
        text.Substring(0, pos) + replace + text.Substring(pos + search.Length)
```

```
let hasDepth (lines: seq<string>) = Seq.exists(fun x -> String.exists(fun c -> c.Equals '\\')
x) lines
```

```
let rec handleThings (lines: seq<string>, spaces: string) : (seq<string>) =
    if hasDepth lines then
        let transformedLines =
            lines
            |> Seq.filter(fun x -> startsWithCount(x, lines).Equals 1)
            |> Seq.map(fun x -> replaceFirst(x, "\\ ", "\n" + spaces))

        handleThings(transformedLines, (spaces + " "))
    else
        lines
```

```
[<EntryPoint>]
let main argv =
    let lines = readLines "data.txt" |> Seq.sort
    let result =
        handleThings(lines, " ")
```



```
|> Seq.map(fun x -> x.Split('\n'))
|> Seq.concat
|> Seq.distinct

Seq.iter print result

Console.ReadKey() |> ignore
0 // return an integer exit code
```

3.3. Pradiniai duomenys ir rezultatai

duom.txt

```
WINNT\SYSTEM32\CONFIG
GAMES
HOME
WIN\SOFT
GAMES\DRIVERS
WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86
```

Rezultatai

```
GAMES
  DRIVERS
HOME
WINNT
  SYSTEM32
    CERTSRV
      CERTCO~1
        X86
  CONFIG
WIN
  SOFT
```

4. Prolog (L4)

4.1. Darbo užduotis

Gautos užduotys iš sąrašo – 8 ir 9:

8. Rekursiškai suskaičiuokite sąrašų (bet kokio gylio) sveikų skaičių sumą
9. Dviejų dimensijų sąrašė raskite palindromus (žodžiai, iš abiejų pusių skaitomi vienodai)

4.2. Programos tekstas

```
uzd8([], Sum) :- format("~a~n", Sum).
uzd8([H|T], Sum) :-
    integer(H),
    NewSum is Sum + H,
    uzd8(T, NewSum)
;
    not(integer(H)),
    uzd8(T, Sum).

isReverse(List) :-
    reverse(List, List),
    string_codes(Str, List),
    format("~a~n", Str).
isReverse(_).
```

```

row([]).
row([H|T]) :-
    string_codes(H, Chars),
    isReverse(Chars),
    row(T).

uzd9([]).
uzd9([H|T]) :-
    row(H),
    uzd9(T).

start :-
    writeln('8 užd atsakymas:'),
    Numbers = [1, 2, 4, 1.3, 4.5, 3],
    uzd8(Numbers, 0),
    writeln('9 užd atsakymas:'),
    Words = [["aba", "bbb", "ca"], ["ds", "eegee", "fa"]],
    uzd9(Words).

```

4.3. Pradiniai duomenys ir rezultatai

8 užduoties duomenys: [1, 2, 4, 1.3, 4.5, 3]

8 užduoties rezultatai: 10

9 užduoties duomenys:

["aba", "bbb", "ca"],
["ds", "eegee", "fa"]

9 užduoties rezultatai:

aba
bbb
eegee