

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**  
**INFORMATIKOS FAKULTETAS**

**Programavimo kalbų teorija (P175B124)**  
*Laboratorinių darbų ataskaita*

Atliko:

IFF-7/2 gr. studentas

Justas Milišiūnas

2020-04-01

Priėmė:

Doc. Aštrys Kirvaitis

# TURINYS

<b>1. Python (L1)</b>	<b>3</b>
1.1. Darbo užduotis	3
1.2. Programos tekstas	3
1.3. Pradiniai duomenys ir rezultatas	3
<b>2. Scala (L2)</b>	<b>3</b>
2.1. Darbo užduotis	3
2.2. Programos tekstas	3
2.3. Rezultatai	6
<b>3. Haskell (L3)</b>	<b>6</b>
<b>4. Prolog (L4)</b>	<b>6</b>

## 1. Python (L1)

### 1.1. Darbo užduotis

Dažna rašymo problema yra paspausti klaviatūros raidę per vieną į dešinę. Pvz. vietoje „Q“ paspaudžiama „W“, „J“ parašoma kaip „K“. Užduotis yra gauto teksto raides perstumti per viena poziciją į kairę pagal QWERTY klaviatūros išdėstymą.

### 1.2. Programos tekstas

```
class Converter:
    def __init__(self, _rows):
        self.rows = _rows
    def convert(self, _input):
        output = ""
        for letter in _input:
            if letter == ' ':
                output += letter
                continue
            index = self.find_index(letter)
            output += self.rows[index[0]][index[1] - 1]
        return output

    def find_index(self, letter):
        for i, rows in enumerate(self.rows):
            for j, item in enumerate(rows):
                if item == letter:
                    return [i, j]
```

### 1.3. Pradiniai duomenys ir rezultatas

Pradiniai duomenys:

input = "O S, GOMR YPFSU/"

Rezultatas:

Input text: O S, GOMR YPFSU/

Output text: I AM FINE TODAY.

## 2. Scala (L2)

### 2.1. Darbo užduotis

1. Panaudoti bent kelis master boto išleidžiamus botų padėjėjų tipus (pvz.: minos, raketos į priešus, "kamikadzės", rinkikai, masalas ir pan.)
2. Panaudoti bet kurį vieną iš kelio radimo algoritmų (DFS, BFS, A\*, Greedy, Dijkstra).

Šiame laboratoriniame darbe panaudoti padėjėjų tipai: minos, raketos į priešus. Kelio radimui naudojamas BFS algoritmas.

### 2.2. Programos tekstas

BFS kelio radimas:

```
def getPathBFS(position: XY, bot: Bot): Option[Array[XY]] = {
    val positionAbs = this.absPosFromRelPos(position)
    var visitedCells = Set[(XY, XY)]()
    var queue = mutable.Queue[XY]()
    val centerPos = this.center

    queue.enqueue(centerPos)
    while (queue.nonEmpty) {
        val current = queue.dequeue()

        val neighbours = getNeighbours(current, bot)
        for (neighbour <- neighbours) {
```

```

        // Checks if current neighbour was not visited
        if (!visitedCells.exists(c => c._1.equals(neighbour))) {
            visitedCells += Tuple2(neighbour, current)
            if (neighbour.equals(positionAbs))
                return Some(extractPath(centerPos, positionAbs, visitedCells))
        }
        queue.enqueue(neighbour)
    }
}

None
}

```

Kelio išgavimas iš aplankytų langelių:

```

def extractPath(startPos: XY, endPos: XY, visitedCells: Set[(XY, XY)]): Array[XY] = {
    var path = mutable.ArrayBuffer[XY]()
    path += endPos

    var currentPos = endPos
    while (currentPos != startPos) {
        val nextPos = visitedCells.filter(c => c._1.equals(currentPos)).toArray
        currentPos = nextPos(0)._2
        path += currentPos
    }

    path.toArray.dropRight(1).reverse.map(c => relPosFromAbsPos(c))
}

```

Šalia esančių langelių gavimas:

```

def getNeighbours(position: XY, bot: Bot): Set[XY] = {
    var neighbours = Set[XY]()

    val allCells = Array(XY.UpLeft, XY.Up, XY.RightUp, XY.Right, XY.DownRight,
        XY.Down, XY.LeftDown, XY.Left)
    for (cell <- allCells) {
        val cellAbsPos = position + cell
        if (cellAbsPos.x >= 0 && cellAbsPos.x < size &&
            cellAbsPos.y >= 0 && cellAbsPos.y < size) {
            val cellValue = this.cellAtAbsPos(position + cell)
            if (cellValue != 'W' && cellValue != 'M' && cellValue != 'S' &&
                cellValue != 's' && cellValue != 'p') {

                if (!bot.isSlave) {
                    if (cellValue != 'm' && cellValue != 'b')
                        neighbours += cell + position
                } else {
                    neighbours += cell + position
                }
            }
        }
    }

    neighbours
}

```

Geriausias krypties radimas:

```
def directionToBestTarget(bot: Bot, targetPriority: Array[Char]): (XY, Char) = {
  for (target <- targetPriority) {
    bot.view.offsetToNearest(target) match {
      case Some(offset) =>
        bot.log("[Mini-Bot] Nearest offset found: " + offset)
        bot.view.getPathBFS(offset, bot) match {
          case Some(path) =>
            var pathString = "Found path:\n"
            for (shit <- path) {
              pathString += shit.toString + "\n"
            }

            bot.log(pathString)
            return (path(0), target)
          case None =>
            bot.log("[Mini-Bot] Nearest path not found: ")
        }
      case None =>
        bot.log("[Mini-Bot] Nearest offset not found: ")
    }
  }

  (XY.Zero, '_')
}
```

Master tipo boto valdymas:

```
def forMaster(bot: Bot): Unit = {
  val targetPriority = Array('B', 'P', '?')
  val direction = directionToBestTarget(bot, targetPriority)
  bot.move(direction._1)

  if (bot.energy >= 200 && bot.slaves < 4) {
    tryLaunchMissile(bot)
  }
}
```

Padejėjo tipo boto valdymas:

```
def forSlave(bot: MiniBot): Unit = {
  val targetPriority = Array('m', 'b')
  val direction = directionToBestTarget(bot, targetPriority)
  bot.move(direction._1)

  if (bot.collision.isDefined) {
    bot.explode(3)
  }
  else
    bot.view.offsetToNearest(direction._2) match {
      case Some(offset) =>
        val distance = offset.length
        if (distance < 3)
          bot.explode(6)
        case None =>
      }
    }
}
```

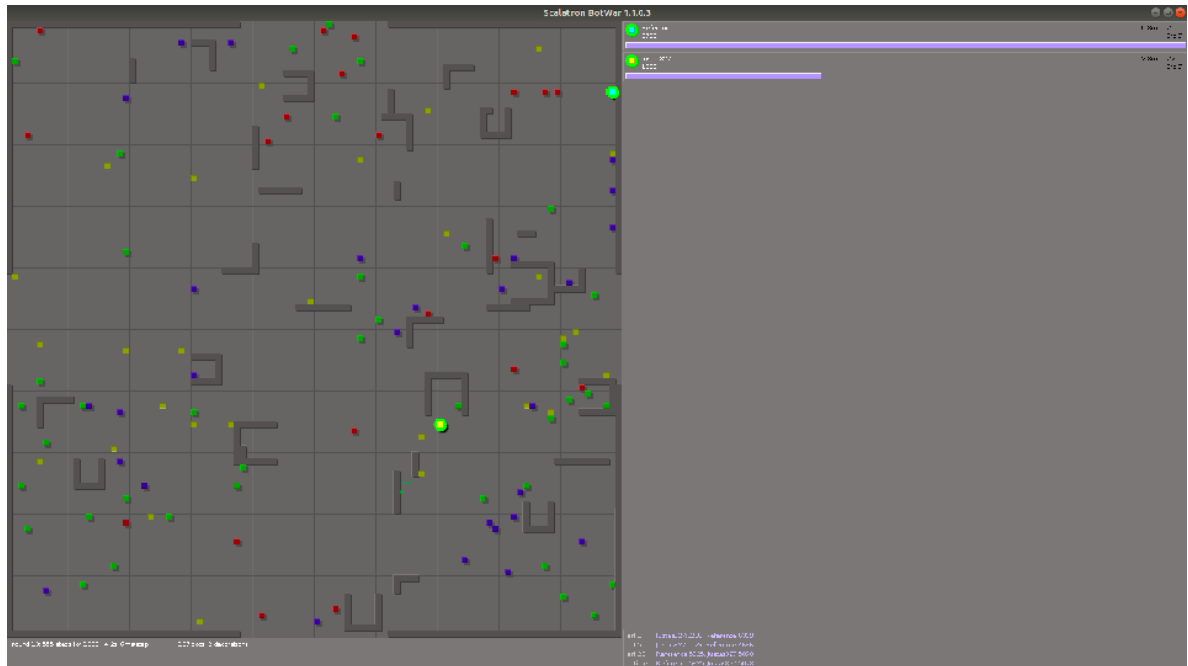
Bandymas paleisti raketą:

```
def tryLaunchMissile(bot: Bot): Unit = {
  bot.view.offsetToNearest('m') match {
    case Some(offset) =>
      bot.log("Missile launched!")
      val mine = bot.slaves == 0
      bot.spawn(offset, ("slaveDirection", offset), ("isMine", mine))
    case None =>
  }
}
```

## 2.3. Rezultatai

Po 20 rundų šis bota surinko 5078 taškus. Reference bota surinko 5325 taškus. Iš šių rezultatų matome, kad šių botų surinktas taškų skaičius labai panašūs. Taip atsitiko dėl to, kad botai atlieka beveik tas pačias funkcijas.

```
last 1: Justas327:8298, Reference:6369
last 5: Justas327:5179, Reference:4636
last 20: Reference:5325, Justas327:5078
all time: Reference:5325, Justas327:5078
```



## 3. Haskell (L3)

## 4. Prolog (L4)