ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Курсовой проект
по дисциплине «Функциональное программирование»

Выполнил студент гр. 3530904/80001: Чижов Н.В.

Руководитель ассистент ВШПИ Лукашин А. А.

Санкт-Петербург
2019

1. Задание.

Калькулятор, поддерживающий простые арифметические операции, приоритеты и скобки.

2. Ход работы.

2.1. Алгоритм решения.

Сначала выражение приводится к стандартному виду: убираются лишние пробелы/табуляции/переносы строки, заменяются запятые на точки. Приведенное выражение передается в функцию, переводящую выражение из инфиксной формы в постфиксную. Процесс выполняется при помощи стека.
Выражение в постфиксной форме переходит уже в функцию расчета. Выражение читается слева направо: постоянные помещаются в стек, выражения применяются к двум верхним постоянным в стеке.

2.2. Результат работы.

```
M:\Programming\ICST\Func\hsse-fp-2019-2\tasks\python>py calculator.py
Expression: "(1.1 + 5 * 2) * 10 + 5"
Result is: 116.0
Postfix notation is: 1.1 5 2 * + 10 * 5 +


Expression: "12852,643 / 37 * 0,32 + 31 / 6 + 52"
Result is: 168.3246601801802
Postfix notation is: 12852.643 37 / 0.32 * 31 6 / + 52 +
```

3. Выводы.

В ходе работы был изучен функциональный подход к программированию в языке программирования Python 3, который значительно
отличается от стандартного императивного подхода. Изучены некоторые основные алгоритмы, используемые в функциональном программировании и произведена работа с ними.

Приложение.

calculator.py

```
1.  def readOperator(expr: str):
2.    isNumber = expr[0].isdigit()
3.    i = 0
4.    if (isNumber):
5.      point = False
6.      for c in expr:
7.        if (c.isdigit()):
8.          i += 1
9.        elif (c == '.'):
10.         if point:
11.           raise 2
12.         i += 1
13.         point = True
14.       else:
15.         break
16.   else:
17.     if (expr[0] not in ['+', '-', '*', '/', '(', ')']):
18.       raise 2;
19.     i = 1
20.
21.   return (expr[:i], expr[i:])
22.
23. def isFloat(num: str):
24.   try:
25.     float(num)
26.     return True
27.   except ValueError:
28.     return False
29.
30. sanitize = lambda expr: "".join(expr.split()).replace(',', '.')
31.
32. def infixToPostfix(expr: str):
33.   expr = sanitize(expr)
34.   high = ['*', '/']
35.   low = ['+', '-']
36.
37.   stack = []
38.   opStack = []
39.   while (len(expr)):
40.     op, expr = readOperator(expr)
41.
```

```
    if (op.isdigit() or isFloat(op)):
      stack += [op]
    elif (op == '('):
      opStack += [op]
    elif (op == ')'):
      while (1):
        if (opStack[-1] == '('):
          opStack = opStack[:-1]
          break
        else:
          stack, opStack = stack + [opStack[-1]], opStack[:-1]
    else:
      while (1):
        if ((not len(opStack)) or opStack[-1] == '('):
          opStack += [op]
          break
        else:
          if (opStack[-1] in low) and (op in high):
            opStack += [op]
            break

          if ((opStack[-1] in high) and (op in high)) or ((opStack[-1] in low) and (op in low)):
            stack, opStack = stack + [opStack[-1]], opStack[:-1] + [op]
            break

          if (opStack[-1] in high) and (op in low):
            stack, opStack = stack + [opStack[-1]], opStack[:-1]
          else:
            break

  stack += opStack[::-1]
  return stack

def calculate(expr: str):
  expr = sanitize(expr)
  ops = ['+', '-', '*', '/']
  postfix = infixToPostfix(expr)
  stack = []
  for op in postfix:
    if (op in ops):
      rhs, lhs, stack = stack[-2], stack[-1], stack[:-2]
      stack += [eval('%s %s %s' % (rhs, op, lhs))]
    else:
      stack += [op]

```

```
87.   return stack[0]
88.
89. def outputExpression(expr: str):
90.   print("Expression: \"%s\"" % expr)
91.   print("Result is: %s" % calculate(expr))
92.   print("Postfix notation is: %s" % ' '.join(infixToPostfix(expr)))
93.
94. def main():
95.   outputExpression('(1.1 + 5 * 2) * 10 + 5')
96.   print('\n')
97.   outputExpression('12852,643 / 37 * 0,32 + 31 / 6 + 52') # Thank you, @Polykek2K!
98.
99. if __name__ == "__main__":
100.         main()
```