МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

**технологический колледж императора петра i**

**ОТЧЁТ**

**по практическим работам**

|  |  |  |
| --- | --- | --- |
| По дисциплине/междисциплинарному курсу | | 01.01 Разработка программных модулей |
|  | | |
|  | | |
|  |
|  | | |

|  |  |
| --- | --- |
|  | Выполнил (-а) обучающийся (-аяся):  Зимин Данила Викторович |
|  | (Ф.И.О.) |
|  | Специальность:  09.02.07 Информационные системы и программирование |
|  | (код и наименование) |
|  | Курс: 3 |
|  | Группа: 404017 |
|  | Преподаватель:  Воронцов Роман Антонович |
|  | (Ф.И.О.) |

Архангельск 2022

**Практическая работа №1**

Изучение и настройка системы контроля версий. Повторение

**Цель работы**

Научиться работать и настраивать систему контроля версий. Решить приложенные задачи

**Ход работы**

Листинг 1 – Решение уравнения

|  |
| --- |
| from math import \*  x = int(input())  if(x>0):      print("y=", sin (x)\*\*2)  else:      print("y=", 1+ 2\*sin(x)\*\*2) |

Листинг 2 – Палиндром

|  |
| --- |
| n=input("Введите целое трехзначное число: ")  if n==n[2]+n[1]+n[0]:      print("Палиндром")  else:      print("Не палиндром") |

Листинг 3 – Условия

|  |
| --- |
| num = int(input('Введите число: '))  number\_1 = num // 100  number\_2 = num // 10 % 10  number\_3 = num % 10  # a)  if number\_1 == number\_2 == number\_3:      print('a) Да')  else:      print('a) Нет')  # б)  if number\_1 == number\_2 or number\_1 == number\_3 or number\_2 == number\_3:      print('б) Да')  else:      print('б) Нет') |

Листинг 4 – Перемещение первой цифры

|  |
| --- |
| print(input()[::-1]) |

Листинг 5 – Перемещение последней цифры

|  |
| --- |
| n = int(input())  nn = n % 10  n //= 10  n = nn \* 100 + n  print(n) |

Листинг 6 – Перестановка первой и последней цифры

|  |
| --- |
| x = int(input())  f = x // 100  s = x % 100 // 10  print(s \* 100 + f \* 10 + x % 10) |

Листинг 7 – Найти x

|  |
| --- |
| n=int(237)  c=n//100  n=(n%100)\*10+c  print (n, '<- исходное число Х') |

Листинг 8 – Много условий

|  |
| --- |
| a, b, c = int(input()), int(input()), int(input())  if (a>100) and (b>100):  print("YES")  else:  print("NO")  if a % 2 == 0 and b % 2 == 0:  print("NO")  else:  print("YES")  if a > 0 or b > 0:  print("YES")  else:  print("NO")  if a%3 == 0 and b%3 == 0 and c%3 == 0:  print("YES")  else:  print("NO")  if ((a<50) and (b>=50) and (c>=50)) or ((a>50) and (b<50) and (c>=50)) or ((a>=50) and (b>=50) and (c<50)):  print("YES")  else:  print("NO")  if (a<0) or (b<0) or (c<0):  print("YES")  else:  print("NO") |

Листинг 9 – Сумма фактариалов

|  |
| --- |
| from math import factorial  n = int(input())  sum = 0  for i in range(1, n+1):  sum += factorial(i)  print(sum) |

Листинг 10 – Купюры

|  |
| --- |
| n = int(input())  lst = [64, 32, 16, 8, 4, 2, 1]  ans = []  while n>0:  for i in lst:  if n >= i:  n -= i  ans.append(i)  break  print(f'Понадобится {len(ans)}шт. купюр, а именно:')  print(\*ans) |

Листинг 11 – Фермер без калькулятора

|  |
| --- |
| b, k, t = int(), int(), int()  for b in range(0, 10):  for k in range(0, 20):  for t in range(0, 200):  if 20\*b+10\*k+t==200 and b+k+t==100:  print('Быков ', b, 'коров ', k, 'телят', t)  else:  continue |

Листинг 12 – Сумма списка

|  |
| --- |
| a = [1, 2, 3, 4, 5, 6, 7, 8, 9]  print( sum(v for i,v in enumerate(a) if not i%2) - sum(v for i,v in enumerate(a) if i%2))  a = [1,3,5]  print( sum(v for i,v in enumerate(a) if not i%2) - sum(v for i,v in enumerate(a) if i%2)) |

Листинг 13 – Перестановка соседей

|  |
| --- |
| a = [int(x) for x in input().split()]  for i in range(0, len(a)-1, 2):  a[i], a[i+1] = a[i+1], a[i]  print(' '.join(str(i) for i in a))  #Вводить с пробелами |

Листинг 14 – Список с генератором

|  |
| --- |
| n = int(input())  mass1 = []  mass2 = []  c = []  for i in range(1, n+1):  #print(i, end=' ')  mass1.append(i)  for i in range(1, n+1):  #print(i, end=' ')  mass2.append(i)  for i in range(0, n):  c= str(mass1[i])+str(mass2[i])  if i+1<n:  print(c, end=', ')  else:  print(c, end=".") |

Листинг 15 – Четные и нечетные

|  |
| --- |
| s=input()  a=[int(s) for s in s.split()]  for i in a:  if int(i)%2 == 0:  print(i, end=' ') |

Листинг 16 – Гости

|  |
| --- |
| from collections import Counter  name = input()  my\_list =[]  my\_list.append(name)  count = 0  while name != "":  name = input()  if name in my\_list:  count += 1  my\_list.remove(name)  continue  else:  my\_list.append(name)  print(Counter) |

Листинг 17 – Оценки Васи

|  |
| --- |
| grades = [5, 4, 5, 3, 2, 5, 4, 3, 5, 5, 4, 2, 2, 3]  print(f'Средний балл: {round((sum(grades) / len(grades)), 2)}')  print(\*grades, sep=';') |

Листинг 18 – Список с условиями

|  |
| --- |
| from random import randint  from unittest import result  numbers = []  for i in range(10):  numbers.append(randint(1, 20))  a = numbers  result = []  for i in a:  if i&1:  result.append(i + 2)  else:  result.append(i \* i)  print(result) |

Листинг 19 – Рекурсивная процедура перевода

|  |
| --- |
| def conv(n,r):  if n<r:  return "0123456789ABCDEF"[n]  else:  k=n%r  return conv(n//r,r)+"0123456789ABCDEF"[k]  print(conv(12,2)) |

Листинг 20 – Lambda-функция и функция map

|  |
| --- |
| s = [7.4, 22.6, 49.7, 701.5, 14]  n = list(map(lambda x: x % 7, s))  print(n) |

Листинг 21 – Заглавная буква

|  |
| --- |
| a = ['катя', 'маша', 'таня', 'саша']  b = list(map(lambda x: x.title(), a))  print('Исходный список')  print(a)  print('Конечный список')  print(b) |

Листинг 22 – Прозвища

|  |
| --- |
| a = ['Даня', 'Никита', 'Николай','Филипп']  b = list(map(lambda x: hash(x), a))  print('Исходный список')  print(a)  print('Конечный список')  print(b) |

Листинг 23 – Объединение множеств

|  |
| --- |
| a = set('1234')  b = set('3456')  print(a | b) |

Листинг 24 – Количество слов

|  |
| --- |
| str\_text = "Какой то текст для примера: грамматика бессмысленна как и все что связано с правописанием так что пофиг мне"  word\_count = str\_text.split(" ")  word\_count = len(word\_count) #len бкядькя бляа в дияла  #Раздели текст пизде плей всв пробел - получаем диспалами только  print("Текст для примера:")  print(str\_text)  print("Сколько в тексте слов:")  print(word\_count) |

Листинг 25 – Вывод слов

|  |
| --- |
| from collections import Counter  words = []  for \_ in range(int(input())):  words.extend(input().split())  counter = Counter(words)  pairs = [(-pair[1], pair[0]) for pair in counter.most\_common()]  words = [pair[1] for pair in sorted(pairs)]  print('\n'.join(words)) |

Листинг 26 – Самое частое слово

|  |
| --- |
| d = {}  for i in range(int(input())):  for word in input().split():  d[word] = d.get(word, 0) + 1  for i in sorted(d.items(), key=lambda x: (x[0])): #(i[0]-->keys, i[1]-->values)  if i[1] == max(d.values()):  print(i[0])  break |

Листинг 27 – Вывод суммы из файла

|  |
| --- |
| with open(r'C:\3.Programming\FileDP\Chisla.txt', encoding='utf-8') as datfile:  text = datfile.read()  print(sum(map(int, text.split(None, 2)[:2]))) |

Листинг 28 – Есть или нет

|  |
| --- |
| with open(r'C:\3.Programming\FileDP\Znak.txt', encoding='utf-8') as f:  print('yes' if input() in f.read() else 'no') |

Листинг 29 – Сумма в строке

|  |
| --- |
| with open(r'C:\3.Programming\FileDP\ChislaMany.txt', encoding='utf-8') as t:  print(\*(sum(map(int, line.split())) for line in t.readlines()), sep='\n') |

Листинг 30 – Количество букв, слов, строк

|  |
| --- |
| import re  res = """Input file contains:  {} letters  {} words  {} lines  """  with open(r'C:\3.Programming\FileDP\Opredelenie.txt', encoding='utf-8') as fh:  f = fh.read().rstrip()  count\_lines = len(f.split('\n'))  count\_words = len(re.findall(r"\w+", f))  count\_letters = sum([1 for x in f if x.isalpha()])  print(res.format(count\_letters, count\_words, count\_lines)) |

**Вывод**

На данной практической работе я научился работать и настраивать систему контроля версий. Решил приложенные задачи.

**Практическая работа №2**

Разработка, оценка сложности и оформление алгоритмов линейной структуры

**Цель работы**

Научиться разрабатывать, проводить оценку сложности и оформлять линейные алгоритмы.

**Ход работы**

**Задание 1.** Разработать линейный алгоритм и оценить его сложность.

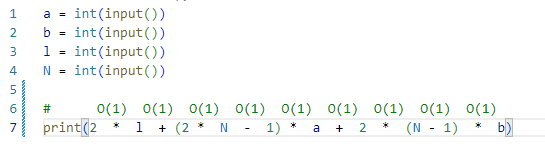


Рисунок 1 – Линейный алгоритм

Алгоритм приведенный на рисунке 1 имеет сложность О(1). В первых четырех строках происходит ввод переменных. В шестой строке приводится объяснения сложности выполнения строки семь.

**Контрольные вопросы**

1. Что такое сложность алгоритма?

Сложность алгоритма – это количественная характеристика, которая говорит о том, сколько времени, либо какой объём памяти потребуется для выполнения алгоритма.

Бывают 5 разных типов алгоритмов:

* O(1)
* O(n)
* O(n^2)
* O(log n)
* O(n\*log n)

1. O(1)

Время выполнения будет одним и тем же для любого количества данных. То есть алгоритм выполняется за постоянное/константное время.

2.O(n)

Чем больше массив, тем больше операций нам потребуется. Данный тип алгоритмов называют линейными или что он линейно масштабируется.

3.O(n^2)

Допустим, у нас есть алгоритм для поиска какого-то значения в матрице 100 на 100 ячеек.

Чтобы обойти цикл все строки и столбца будем использовать вложенный цикл.

4.O(log n)

В случае, если все значения массива будут отсортирован, то можно воспользоваться бинарным поиском: делим массив на две половины, отбрасываем не нужную, оставшуюся опять делим на две части и так пока не найдём нужно значение. Такой тип алгоритмов называется “разделяй и властвуй” Divide and Conquer.

5.O(n\*log n)

Вернёмся, к примеру к матрицам. Если возможность заменить вложенный цикл на бинарный поиск. Таким образом у нас останется перебор всех элементов O(n) и внутри будет O(log n).

**Вывод**

На данной практической работе я научился разрабатывать, проводить оценку сложности и оформлять линейные алгоритмы.

**Практическая работа №3**

Оценка сложности алгоритмов сортировки. Оценка сложности алгоритма поиска

**Цель работы**

Научиться разрабатывать алгоритмы сортировки и поиска, а также оценивать их сложность.

**Ход работы**

**Задание 1.** Разработать алгоритм сортировки и оценить его сложность

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Алгоритм сортировки

При помощи данного примера можно узнать сложность алгоритма указанного на рисунке 1. Сложность этого алгоритма равна O(1).

**Задание 2.** Разработать алгоритм поиска и оценить его сложность

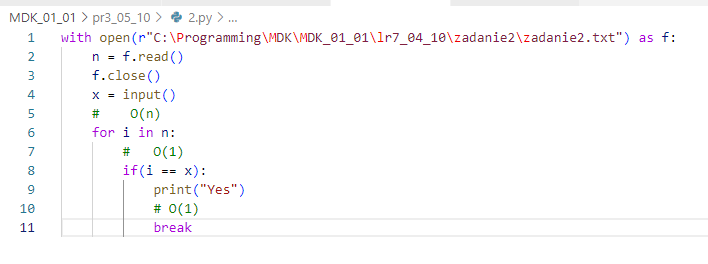


Рисунок 2 – Алгоритм поиска

Алгоритм, приведенный на рисунке 2 имеет сложность О(n). В первой строке происходит открытие файла. Во второй строке происходит чтение и присваивание содержимого файла. В третьей строке происходит закрытие файла. В четвертой строке присваивается переменная. В строках с шестой по одиннадцатую происходит поиск элемента.

**Контрольные вопросы**

*1.Что такое алгоритм сортировки?*

Это алгоритм для упорядочивания элементов в массиве. В случае, когда элемент в массиве имеет несколько полей, поле, служащее критериям порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

*2.Что такое алгоритм поиска?*

Алгоритмы поиска предназначены для проверки элемента или извлечения элемента из любой структуры данных, где он хранится. В зависимости от типа операции поиска эти алгоритмы обычно подразделяются на две категории:

* Последовательный поиск: при этом список или массив просматриваются последовательно, и проверяется каждый элемент.

Например: линейный поиск.

* Интервальный поиск: это алгоритмы специально разработаны для поиска в отсортированных структурах данных. Алгоритмы поиска такого типа намного эффективнее, чем линейный поиск, поскольку они многократного нацеливаются на центр структуры поиска и делят пространство поиска пополам.

Например: бинарный поиск.

**Вывод**

На данной практической работе я научился разрабатывать алгоритмы сортировки и поиска, а также оценивать их сложность.

**Практическая работа №4**

Оценка сложности эвристических алгоритмов

**Цель работы**

Научиться оценивать сложность эвристических алгоритмов.

**Ход работы**

**Задание 1.** Оценить сложность эвристического алгоритма.

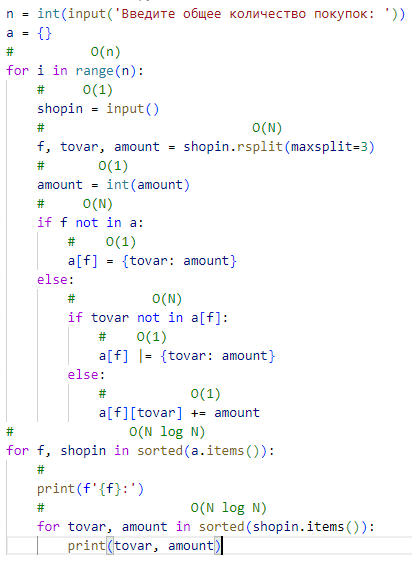


Рисунок 1 – Эвристический алгоритм

При помощи данного примера можно узнать сложность алгоритма указанного на рисунке 1. Сложность этого алгоритма равна .

**Контрольные вопросы**

1. *Что такое эвристический алгоритм?*

Эвристический алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев. Иногда термин «эвристика» используется также для обозначения методов ускорения заведомо точных методов (например, полного перебора).

2. *Что такое формулы оценки сложности алгоритмов?*

Сложность алгоритмов обычно оценивают по времени выполнения или по используемой памяти. В обоих случаях сложность зависит от размеров входных данных: массив из 100 элементов будет обработан быстрее, чем аналогичный из 1000.

**Вывод**

На данной практической работе я научился оценивать сложность эвристических алгоритмов.

**Практическая работа №5**

Оценка сложности рекурсивных алгоритмов

**Цель работы**

Научиться разрабатывать рекурсивные алгоритмы и оценить их сложность.

**Ход работы**

**Задание 1.** Разработать рекурсивный алгоритм вычисления факториала и оценить его сложность.

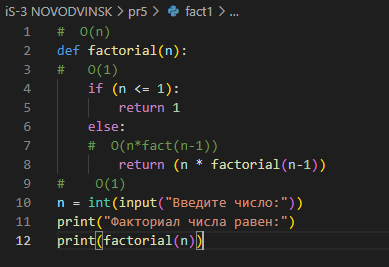


Рисунок 1 – Рекурсивный алгоритм

При помощи данного примера можно узнать сложность алгоритма указанного на рисунке 1. Сложность этого алгоритма равна O (n \* fact(fact(n-1)).

**Контрольные вопросы**

*1.Что такое рекурсивный алгоритм?*

Рекурсивный алгоритм – это алгоритм, в описании которого прямо или косвенно содержится обращение к самому себе. В технике процедурного программирования данное понятие распространяется на функцию, которая реализует решение отдельного блока задачи посредством вызова из своего тела других функций, в том числе и себя самой. Если при этом на очередном этапе работы функция организует обращение к самой себе, то такая функция является рекурсивной.

**Вывод**

На данной практической работе я научился разрабатывать рекурсивные алгоритмы и оценить их сложность.

**Практическая работа №6**

Оформление документации на программные средства

**Цель работы**

Научиться оформлять документацию на программные средства.

**Ход работы**

**Задание.** Оформить документацию на программный код в соответствии с основными форматами документирования: Строки документации Google, reStructuredText, NumPy/SciPy, Epytext.

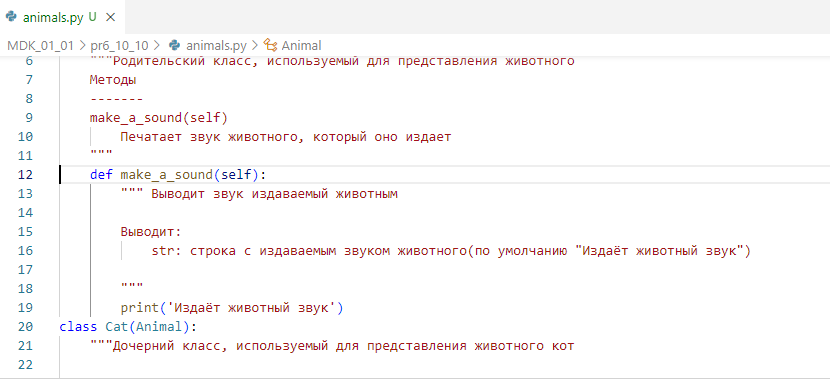


Рисунок 1 – Форматирование Google

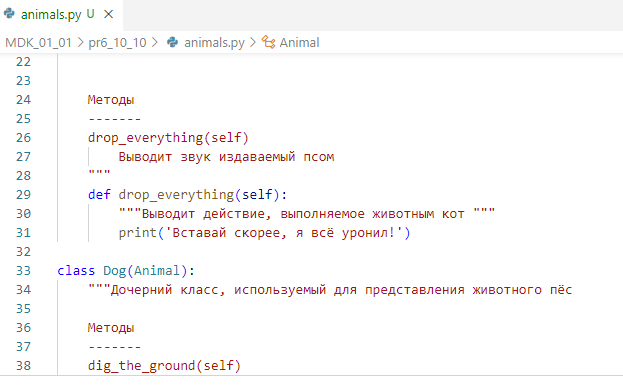


Рисунок 2 – Форматирование reStructuredText

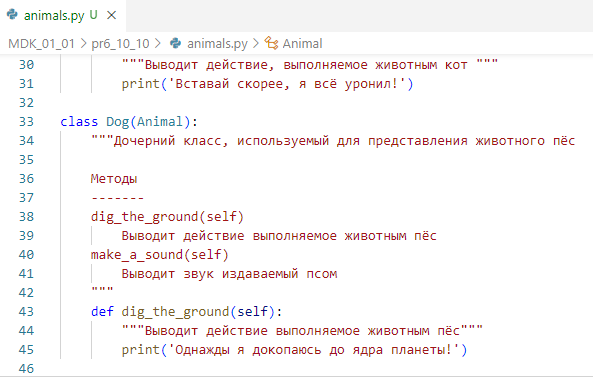


Рисунок 3 – Форматирование NumPy/SciPy

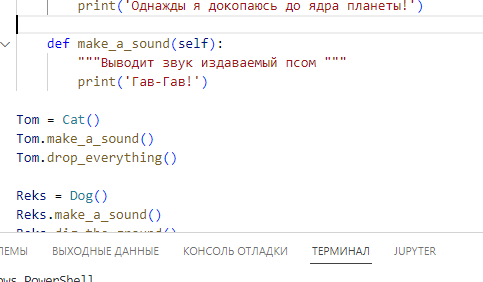


Рисунок 4 – Форматирование Epytext

**Контрольные вопросы**

*1. Что такое стандарт PEP8 и PEP257?*

PEP8 – этот документ описывает соглашение о том, как писать код языка python, включая стандартную библиотеку, входящую в состав python.

PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код.

*2. Перечислите основные форматы документирования кода?*

Форматы документирования кода:

* Google styleguide -> Comments and Docstrings.
* Numpydoc docstring guide.
* Epydoc.
* reStructuredText (reST)

**Вывод**

На данной практической работе я научился оформлять документацию на программные средства.