# Министерство образования и науки Российской Федерации

# Федеральное государственное бюджетное образовательное учреждение высшего образования «Новосибирский государственный технический университет»



## Кафедра параллельных вычислительных технологий

## Домашнее задание № 1 по дисциплине «Методы проектирования и анализа алгоритмов»

**Введение в алгоритмы**

Вариант 12

|  |  |  |
| --- | --- | --- |
|  | Факультет: | ПМИ |
| Группа: | ПМ-53 |
| Студент: | Тябин Е.А. |
| Преподаватель: | Щукин Г.А. |

Новосибирск

2016

# Условие задачи

Дано множество записей о людях. Каждая запись содержит поля: имя, фамилия, день месяца, месяц и год рождения. Требуется реализовать алгоритм из своего варианта задания (язык программирования по выбору) в виде функции, принимающей массив/список записей на вход и возвращающей требуемый результат. Алгоритм желательно по возможности построить из других примитивных/базовых алгоритмов. Правильность работы алгоритма проверить с помощью юнит-тестов. Время работы алгоритма замерить на входных данных разного размера, данные читать из файлов. Входные данные сгенерировать самостоятельно или использовать скрипт генерации (generate.py) в папке record\_gen репозитория <https://github.com/georgy-schukin/mpiaa-py>.

12. Найти всех людей, имя и фамилия которых начинаются с одной буквы.

# -Алгоритм решения

**Алг** find(file){

Начиная с первого человека, Повторять:  
 Если Первая буква Имени = Первой букве Фамилии: Записываем данные человека   
 в список  
 Пока люди не закончились

}

# Текст программы

**/\* Файл DZ.py \*/**

**from** generate **import** read\_file, write\_file

**def** sravn(data):  
 **if** (data[0] == data[data.index(**' '**) + 1]):  
 **return True  
 return False  
  
  
def** find(peoples):  
 odinakovki = []  
 **for** people **in** peoples:  
 **if** (sravn(people)==True):  
 odinakovki.append(people)  
 **return** odinakovki  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 peoples = read\_file(**"records\_1e{0}.txt"**.format(2))  
 find(peoples)

output\_file = **"result.txt"** write\_file(output\_file, odinakovki)

**/\* Файл DZtimer.py \*/**

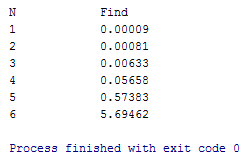
**import DZ**  
**from** timer **import** time\_us  
**from** util **import** powers\_of  
**from** generate **import** read\_file, generate\_records, main  
  
names = read\_file(**"names.txt"**)  
surnames = read\_file(**"surnames.txt"**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **"""find(file1)"""** time\_us({  
 **"Viborka"**: find,  
 }, ns=powers\_of(1, 0, 5), generator=main, repeats=10)

# Юнит-тесты

**/\* Файл Test.py \*/**

**import** unittest  
**import** DZ  
  
  
**class** MyTestCase(unittest.TestCase):  
 **def** test\_DZ(self):  
 self.assertEqual(DZ.find([]), [])  
 **def** test\_sravn(self):  
 self.assertEqual(DZ.find([**"Alexander Call 5 Nov 1979"**]), [])  
 **def** test\_sravn2(self):  
 self.assertEqual(DZ.find([**"Elene Entsminger 1 Apr 1905"**]), [**"Elene Entsminger 1 Apr 1905"**])  
 **def** test\_sravn3(self):  
 test4 = DZ.read\_file(**"test4.txt"**.format(2))  
 self.assertEqual(DZ.find(test4), [**"Elene Entsminger 1 Apr 1905"**, **"Kent Kessler 26 Dec 1952"**])  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 unittest.main()

# Результаты замера времени работы алгоритма



# Выводы

Проделав работу, мы ознакомились со средствами работы со списками в языке Python, успешно построили алгоритм для решения нашей задачи и реализовали его на языке программирования, после чего были сделаны юнит-тесты для проверки правильности работы алгоритма, а также сделаны замеры времени работы алгоритма. Программа выдала верное решение на всех тестах и, следовательно, правильно работает.