ГУАП

КАФЕДРА №41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | А.В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| **ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 8** |
| РАБОТА С ДАННЫМИ В ФОРМАТЕ CSV |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4816 |  |  |  | В.Ф. Губайдулин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Цель работы:** Считать CSV файл соответствующий варианту во внутренние структуры данных разрабатываемой программы. Провести статистическое исследование загруженных данных, следуя дополнительным указаниям преподавателя. Результаты исследования представить в удобном для анализа виде.

**Вариант:** Был выбран вариант №8. Задание варианта представлено (см. таблица 1)

Таблица 1 – Вариант 8

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Название файла набора данных | Способ отражения стран | Дополнительная задача |
| 8 | ElectricityGeneration.csv | B | B |

**Математическая модель:**

Реализована программа для считывания CVS файлов, построения графиков для каждого показателя в столбце “SUBJECT”, а также создания новых файлов с новым временным интервалов и *k* стран, имеющих самое большое значение показателя, и *p* стран, имеющих самое низкое значение показателя. Пользователю необходимо будет ввести страну и значения параметров *k* и *p*. Были подключены библиотеки matplotlib и csv. Реализован класс для проверки с подключением модуля *unittest.*

Функция *check\_country* проверяет правильно ли была введена страна. Принимает массив стран и введённую страну, возвращает 1 или 0.

Функция *check\_third* проверяет было ли введено неотрицательное число, не превышающее лимит. Принимает значение, возвращает ошибку или 0.

В данной программе тестируется только правильность ввода, но не правильность выполнения алгоритма построения графиков и создания новых файлов. В качестве тестирования правильности построения графиков и создинии новых файлов мог выступать другой CSV файл, для которого заранее известны все графики и новые файлы. Для тестирования создан отдельный файл, с подключенным модулем *unittest*, и импортированными функциями *check\_country, check\_third.* В виде теста представлены значения *country*, *n*, а также заранее подсчитанные значения. С тестами можно ознакомится (см. таблица 2). С переменными можно ознакомится (см. таблица 3).

**Таблица тестов:**

Таблица 2 – Таблица тестов

|  |  |  |
| --- | --- | --- |
| № | Входные значения | Выходные значения |
| 1 | mas\_country = ['AUS', 'AUT', 'BEL', 'CAN', 'CZE', 'DNK', 'FIN', 'FRA', 'DEU', 'GRC', 'HUN', 'ISL', 'IRL', 'ITA','JPN', 'KOR', 'LUX', 'MEX', 'NLD', 'NZL', 'NOR', 'POL', 'PRT', 'SVK', 'ESP', 'SWE', 'CHE', 'TUR', 'GBR', 'USA', 'OEU', 'CHL', 'EST', 'ISR', 'SVN', 'OECD']  country = ' ' | 1 |
| 2 | mas\_country = ['AUS', 'AUT', 'BEL', 'CAN', 'CZE', 'DNK', 'FIN', 'FRA', 'DEU', 'GRC', 'HUN', 'ISL', 'IRL', 'ITA','JPN', 'KOR', 'LUX', 'MEX', 'NLD', 'NZL', 'NOR', 'POL', 'PRT', 'SVK', 'ESP', 'SWE', 'CHE', 'TUR', 'GBR', 'USA', 'OEU', 'CHL', 'EST', 'ISR', 'SVN', 'OECD']  country = 'AUS' | 0 |
| 3 | n = '' | 'Ничего не введено. Попробуйте ввести значение снова.' |
| 4 | n = 'a' | 'Введен символ или несколько значений. Попробуйте ввести значение снова.' |
| 5 | n = '-2' | 'Введено отрицательное значение. Попробуйте ввести значение снова.' |
| 6 | n = '37' | 'Превышен лимит. Попробуйте ввести значение снова.' |
| 7 | n = '5.7' | 'Введено вещественное число. Попробуйте ввести значение снова.' |

**Описание разработанной программы:**

Таблица 3 – Таблица переменных

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип переменной | Назначение |
| country | строка | Название страны, для которой будет построен график |
| k | число | Сколько будет строк максимальных значений в новом файле |
| p | число | Сколько будет строк минимальных значений в новом файле |

Была создана блок-схема в соответствие с рисунком 1.

Введённой страны нет в списке?

Такого местоположения не существует. Выберете местоположение из списка.

нет

Чтение файла

да

Получение страны

Вывод списка доступных стран

Создание поля построения графика

Отбор данных для графика

Построение функций

Добавление легенды

Добавление надписей

В значении есть лишние символы?

Введен символ или несколько значений. Попробуйте ввести значение снова.

Значение отрицательное?

нет

'Введено отрицательное значение. Попробуйте ввести значение снова.

да

Превышен лимит?

Превышен лимит. Попробуйте ввести значение снова.

да

нет

Получение *к* и *р*

Значение пусто?

нет

да

Ничего не введено. Попробуйте ввести значение снова.

да

Введено вещ. число?

да

Введено вещественное число. Попробуйте ввести значение снова.

нет

нет

Отбор данных для новых файлов

Запись в файлы

Рисунок 1 – Блок-схема алгоритма

Листинг разработанной программы представлен в Приложение 1.

**Тестирование разработанной программы:**

Были подобраны ошибочные вводные значения и предугаданы ошибки, которые будут выводиться. До написания программы тест был провален в соответствие с рисунком 2. После написания программы тест был успешно пройден в соответствие с рисунком 3. Листинг тестов представлен в Приложение 2.

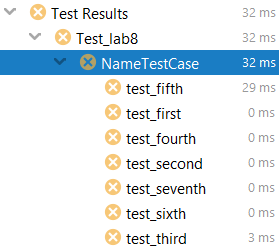


Рисунок 2 – Пример не выполнения тестов

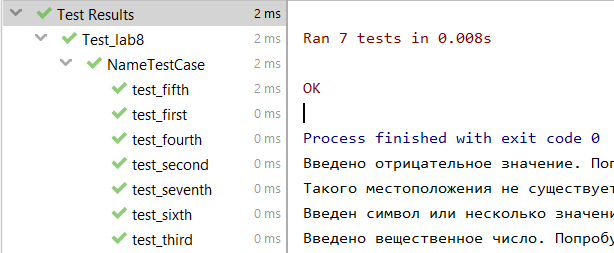


Рисунок 3 – Пример выполнения тестов

**Пример выполнения программы:**

Программа была протестирована с случайными значения в соответствие с рисунком 4, рисунком 5, рисунком 6 и рисунком 7.

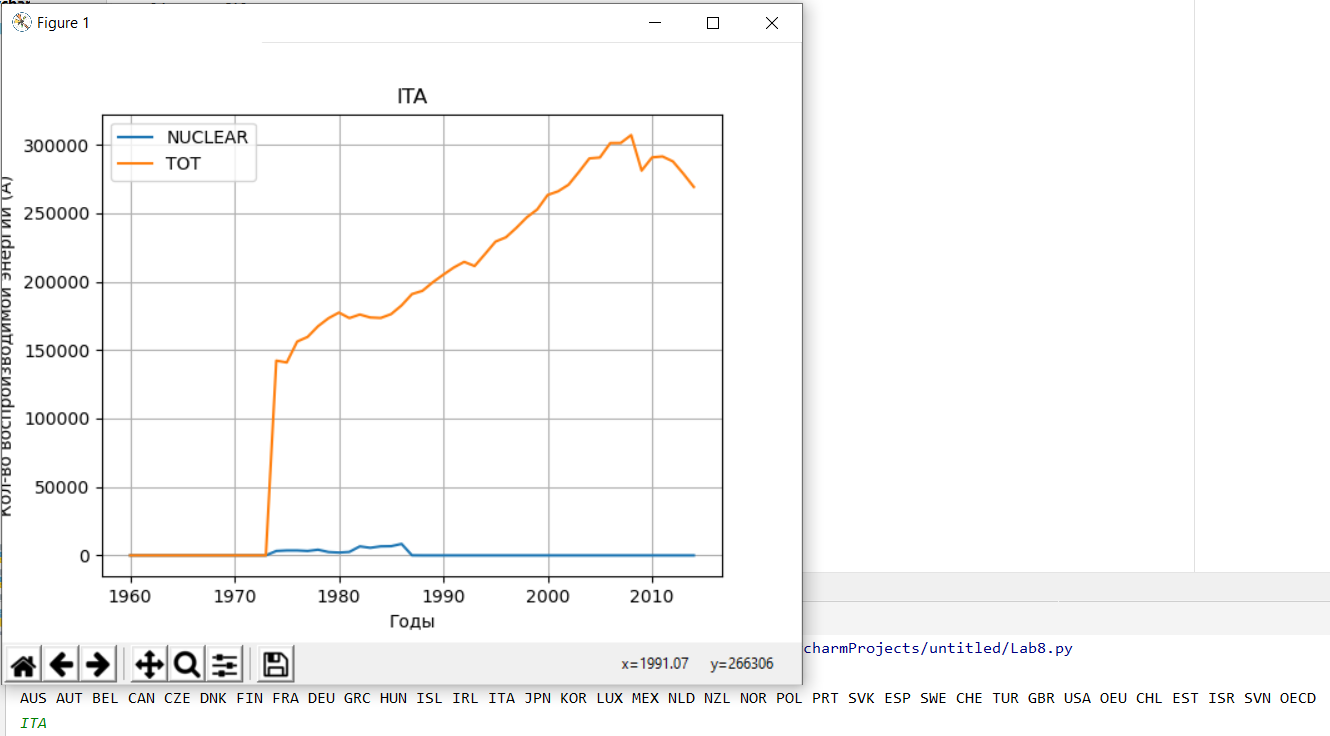


Рисунок 4 – Пример построения графика для страны ITA

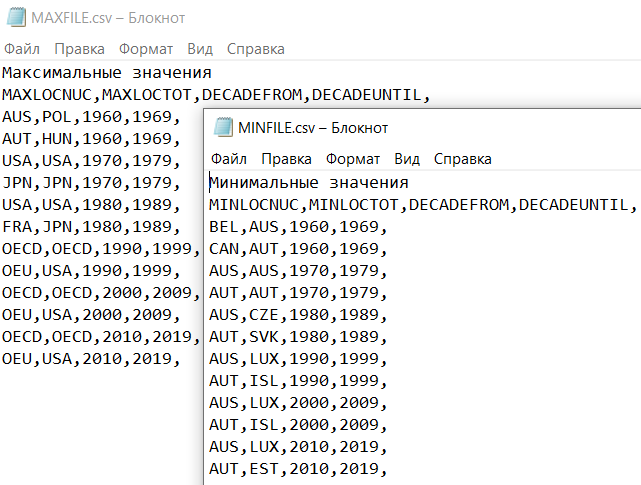


Рисунок 5 – Пример выполнения программы со значениями k=2 и p=2

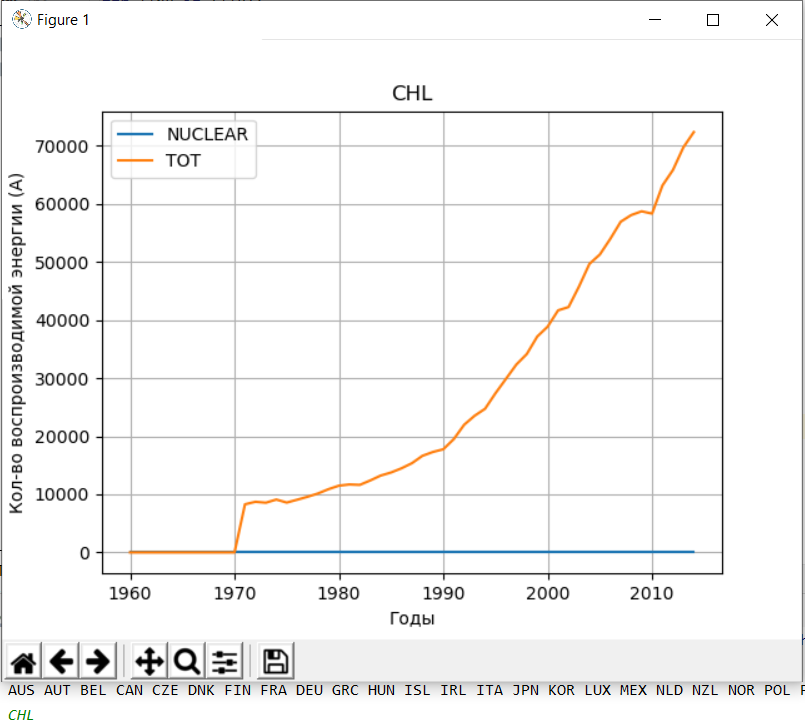


Рисунок 6 – Пример построения графика для страны CHL

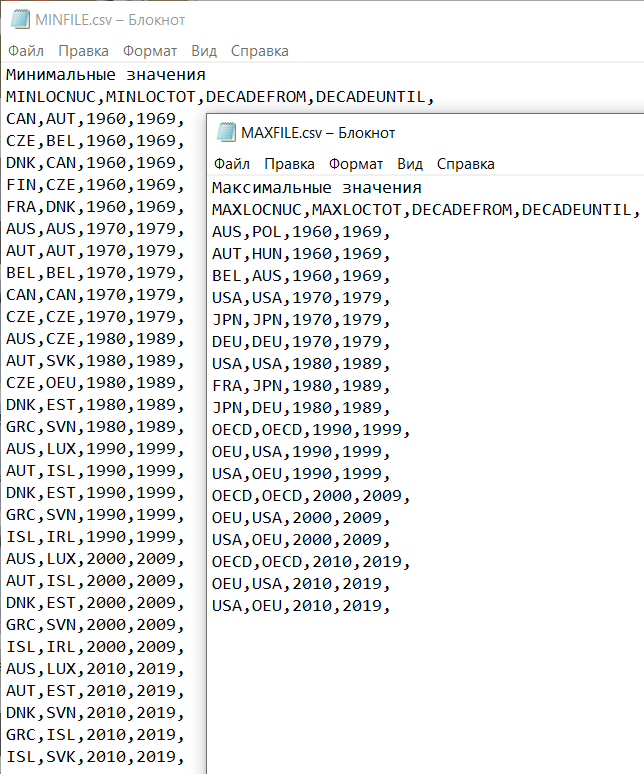


Рисунок 5 – Пример выполнения программы со значениями k=3 и p=5

**Вывод:** Считал CSV файл соответствующий варианту во внутренние структуры данных разрабатываемой программы. Провёл статистическое исследование загруженных данных, следуя дополнительным указаниям преподавателя. Результаты исследования представил в удобном для анализа виде.

**Приложение 1**

**Листинг разработанной программы**

**from** pylab **import** \*  
**import** numpy **as** np  
**import** csv  
  
  
*# проверка правильного выбора местоположения***def** check\_country(country, mas\_country):  
 **if** country **not in** mas\_country:  
 print(**'Такого местоположения не существует. Выберете местоположение из списка.'**)  
 **return** 1  
 **else**:  
 **return** 0  
  
  
*# проверка k и p***def** check\_third(n):  
 **if** n.isspace() **or** str(n) == **''**: *# Проверка на пустое значение* print(**'Ничего не введено. Попробуйте ввести значение снова.'**)  
 **return 'Ничего не введено. Попробуйте ввести значение снова.'  
 if** str(n).isalpha(): *# Проверка на численное значение* print(**'Введен символ или несколько значений. Попробуйте ввести значение снова.'**)  
 **return 'Введен символ или несколько значений. Попробуйте ввести значение снова.'  
 if** float(n) < 0: *# Проверка на <0* print(**'Введено отрицательное значение. Попробуйте ввести значение снова.'**)  
 **return 'Введено отрицательное значение. Попробуйте ввести значение снова.'  
 if** float(n) > 36: *# Проверка на максимальное кол-во стран* print(**'Превышен лимит. Попробуйте ввести значение снова.'**)  
 **return 'Превышен лимит. Попробуйте ввести значение снова.'  
 try**: *# Проверка на вещественное число* int(n)  
 **except** ValueError:  
 print(**'Введено вещественное число. Попробуйте ввести значение снова.'**)  
 **return 'Введено вещественное число. Попробуйте ввести значение снова.'  
 return** 0  
  
mas = []  
mas\_country = []  
x = []  
y1 = []  
y2 = []  
file1 = [[**"MAXLOCNUC"**, **"MAXLOCTOT"**, **"DECADEFROM"**, **"DECADEUNTIL"**]]  
file2 = [[**"MINLOCNUC"**, **"MINLOCTOT"**, **"DECADEFROM"**, **"DECADEUNTIL"**]]  
nuct = []  
nuc = []  
tot = []  
tott = []  
  
*# чтение csv-файла***with** open(**"ElectricityGeneration.csv"**, **'r'**, encoding=**'utf-8'**) **as** rfile:  
 fileridder = csv.reader(rfile, delimiter=**","**)  
 **for** line **in** fileridder:  
 **if** line != []:

Продолжение приложения 1

**if '#' not in** line[0]:  
 mas.append(line)  
rfile.close()  
  
*# выбор страны***for** i **in** mas:  
 **if** i[0] **not in** mas\_country **and** i[0] != **'LOCATION' and** i[3] == **'GWH'**:  
 mas\_country.append(i[0])  
print(**'Выберете местоположение из списка ниже для которого Вы хотите построить график:'**)  
**for** i **in** mas\_country:  
 print(i, end=**' '**)  
print()  
country = input()  
country = country.upper()  
**while** check\_country(country, mas\_country):  
 country = input()  
 country = country.upper()  
  
*# отбор значений для графика***for** i **in** mas:  
 **if** i[0] == country **and** i[2] == **'NUCLEAR' and** i[7] == **'' and** i[3] == **'GWH'**:  
 x.append(int(i[5]))  
 y1.append(float(i[6]))  
 **if** i[0] == country **and** i[2] == **'TOT' and** i[7] == **'' and** i[3] == **'GWH'**:  
 y2.append(float(i[6]))  
  
  
*# построение графика*fig, ax = plt.subplots()  
ax.plot(x, y1, label=**'NUCLEAR'**)  
ax.plot(x, y2, label=**'TOT'**)  
ax.grid() *# Сетка*ax.legend(loc=2) *# Легенда*ax.set\_title(country) *# Название страны графика*ax.set\_xlabel(**'Годы'**) *# Подпись к оси x*ax.set\_ylabel(**'Кол-во воспроизводимой энергии (А)'**) *# Подпись к оси y*show()  
  
*# доп задание*print()  
  
k = input(**'Введите k:'**)  
**while** check\_third(k):  
 k = input(**'Введите k:'**)  
  
p = input(**'Введите p:'**)  
**while** check\_third(p):  
 p = input(**'Введите p:'**)  
  
year = 1960  
**while** year <= 2015:  
 nuct = []  
 nuc = []  
 tot = []  
 tott = []  
  
 **for** i **in** mas:

Продолжение приложения 1

**if** i[3] == **'GWH' and** i[2] == **'NUCLEAR' and** year <= int(i[5]) **and** int(i[5])<= int(year) + 9 **and** int(i[5]) != 2015:  
 **try**:  
 float(i[6])  
 **except** ValueError:  
 i[6] = 0  
 nuct.append(i[0])  
 nuc.append(float(i[6]))  
 **elif** i[3] == **'GWH' and** i[2] == **'TOT' and** year <= int(i[5]) **and** int(i[5]) <= year + 9 **and** int(i[5]) != 2015:  
 **try**:  
 float(i[6])  
 **except** ValueError:  
 i[6] = 0  
 tott.append(i[0])  
 tot.append(float(i[6]))  
  
 **for** j **in** range(int(k)):  
 maxindexnuc = nuct[nuc.index(max(nuc))]  
 maxindextot = tott[tot.index(max(tot))]  
 file1.append([maxindexnuc, maxindextot, int(year), year+9])  
 indexsave = [z **for** z, ltr **in** enumerate(nuct) **if** ltr == maxindexnuc]  
 nuct[:] = [x **for** z,x **in** enumerate(nuct) **if** z **not in** indexsave]  
 nuc[:] = [x **for** z,x **in** enumerate(nuc) **if** z **not in** indexsave]  
 indexsave = [z **for** z, ltr **in** enumerate(tott) **if** ltr == maxindextot]  
 tott[:] = [x **for** z, x **in** enumerate(tott) **if** z **not in** indexsave]  
 tot[:] = [x **for** z, x **in** enumerate(tot) **if** z **not in** indexsave]  
  
 **for** i **in** mas:  
 **if** i[3] == **'GWH' and** i[2] == **'NUCLEAR' and** year <= int(i[5]) **and** int(i[5])<= int(year) + 9 **and** int(i[5]) != 2015:  
 **try**:  
 float(i[6])  
 **except** ValueError:  
 i[6] = 0  
 nuct.append(i[0])  
 nuc.append(float(i[6]))  
 **elif** i[3] == **'GWH' and** i[2] == **'TOT' and** year <= int(i[5]) **and** int(i[5]) <= year + 9 **and** int(i[5]) != 2015:  
 **try**:  
 float(i[6])  
 **except** ValueError:  
 i[6] = 0  
 tott.append(i[0])  
 tot.append(float(i[6]))  
  
 **for** j **in** range(int(p)):  
 minindexnuc = nuct[nuc.index(min(nuc))]  
 minindextot = tott[tot.index(min(tot))]  
 file2.append([minindexnuc, minindextot, int(year), year+9])  
 indexsave = [z **for** z, ltr **in** enumerate(nuct) **if** ltr == minindexnuc]  
 nuct[:] = [x **for** z,x **in** enumerate(nuct) **if** z **not in** indexsave]  
 nuc[:] = [x **for** z,x **in** enumerate(nuc) **if** z **not in** indexsave]  
 indexsave = [z **for** z, ltr **in** enumerate(tott) **if** ltr == minindextot]  
 tott[:] = [x **for** z, x **in** enumerate(tott) **if** z **not in** indexsave]  
 tot[:] = [x **for** z, x **in** enumerate(tot) **if** z **not in** indexsave]

Продолжение приложения 1

year += 10  
  
*# Запись в первый файл*my\_file1 = open(**'MAXFILE.csv'**, **'w'**)  
my\_file1.write(**'Максимальные значения\n'**)  
**for** line **in** file1:  
 **for** j **in** line:  
 a = str(j)+**','** my\_file1.write(str(a))  
 my\_file1.write(**'\n'**)  
my\_file1.close()  
  
  
*# Запись во второй файл*my\_file1 = open(**'MINFILE.csv'**, **'w'**)  
my\_file1.write(**'Минимальные значения\n'**)  
**for** line **in** file2:  
 **for** j **in** line:  
 a = str(j) + **','** my\_file1.write(str(a))  
 my\_file1.write(**'\n'**)  
my\_file1.close()

**Приложение 2**

**Листинг теста**

**import** unittest  
**from** Lab8 **import** check\_country  
**from** Lab8 **import** check\_third  
  
  
**class** NameTestCase(unittest.TestCase):  
 *# Тесты для Lab8#* **def** test\_first(self):  
 *# Тест на страну -#* mas\_country = [**'AUS'**, **'AUT'**, **'BEL'**, **'CAN'**, **'CZE'**, **'DNK'**, **'FIN'**, **'FRA'**, **'DEU'**, **'GRC'**, **'HUN'**, **'ISL'**, **'IRL'**,  
 **'ITA'**,**'JPN'**, **'KOR'**, **'LUX'**, **'MEX'**, **'NLD'**, **'NZL'**, **'NOR'**, **'POL'**, **'PRT'**, **'SVK'**, **'ESP'**, **'SWE'**, **'CHE'**,  
 **'TUR'**, **'GBR'**, **'USA'**, **'OEU'**, **'CHL'**, **'EST'**, **'ISR'**, **'SVN'**, **'OECD'**]  
 result = check\_country(**''**, mas\_country)  
 self.assertEqual(result, 1)  
  
 **def** test\_second(self):  
 *# Тест на страну +* mas\_country = [**'AUS'**, **'AUT'**, **'BEL'**, **'CAN'**, **'CZE'**, **'DNK'**, **'FIN'**, **'FRA'**, **'DEU'**, **'GRC'**, **'HUN'**, **'ISL'**, **'IRL'**,  
 **'ITA'**, **'JPN'**, **'KOR'**, **'LUX'**, **'MEX'**, **'NLD'**, **'NZL'**, **'NOR'**, **'POL'**, **'PRT'**, **'SVK'**, **'ESP'**, **'SWE'**, **'CHE'**,  
 **'TUR'**, **'GBR'**, **'USA'**, **'OEU'**, **'CHL'**, **'EST'**, **'ISR'**, **'SVN'**, **'OECD'**]  
 result = check\_country(**'AUS'**, mas\_country)  
 self.assertEqual(result, 0)  
  
 **def** test\_third(self):  
 *# Тест на пустой ввод* result = check\_third(**''**)  
 self.assertEqual(result, **'Ничего не введено. Попробуйте ввести значение снова.'**)  
  
 **def** test\_fourth(self):  
 *# Тест на сивол* result = check\_third(**'a'**)  
 self.assertEqual(result, **'Введен символ или несколько значений. Попробуйте ввести значение снова.'**)  
  
 **def** test\_fifth(self):  
 *# Тест на отрицательнрое значение* result = check\_third(**'-2'**)  
 self.assertEqual(result, **'Введено отрицательное значение. Попробуйте ввести значение снова.'**)  
  
 **def** test\_sixth(self):  
 *# Тест на лимит* result = check\_third(**'37'**)  
 self.assertEqual(result, **'Превышен лимит. Попробуйте ввести значение снова.'**)  
  
 **def** test\_seventh(self):  
 *# Тест на вещественное число* result = check\_third(**'5.7'**)  
 self.assertEqual(result, **'Введено вещественное число. Попробуйте ввести**

Продолжение приложения 2

**значение снова.'**)  
  
**if** \_\_name\_\_ == **"\_\_name\_\_"**:  
 unittest.main()