Федеральное агентство связи

Ордена Трудового Красного Знамени

федеральное государственное бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторные работы по дисциплине

«Структуры и алгоритмы обработки данных»

Лабораторная работа №4

«Работа со структурами данных стек и дек»

Выполнила студентка группы БСТ2002

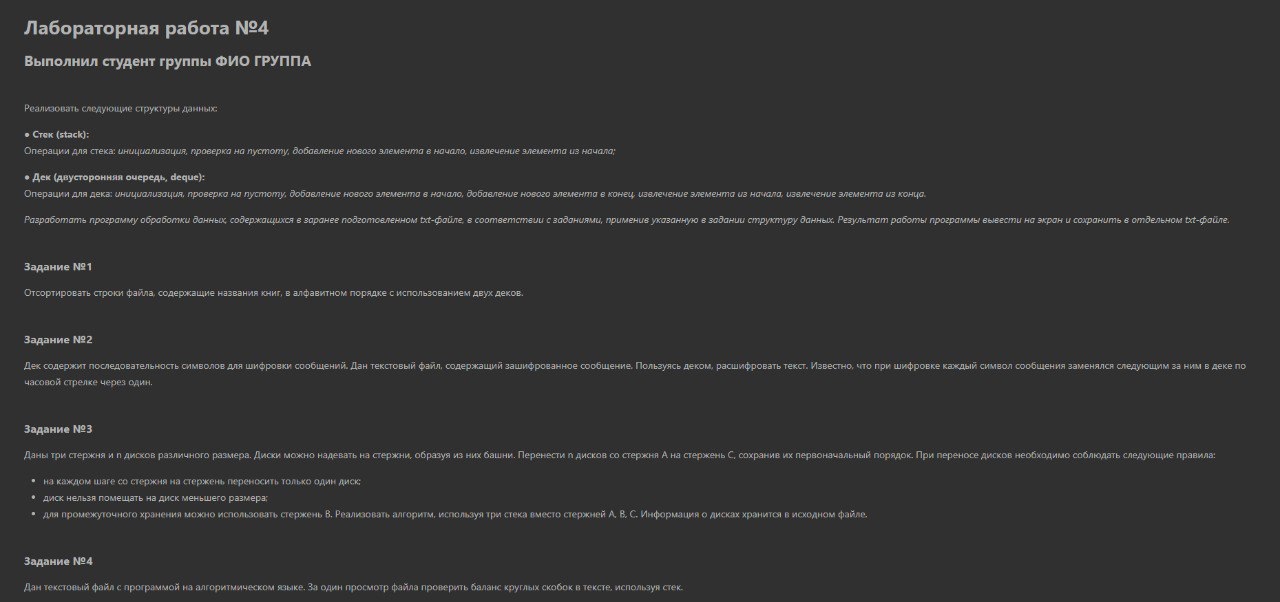
Сергеева А.А.

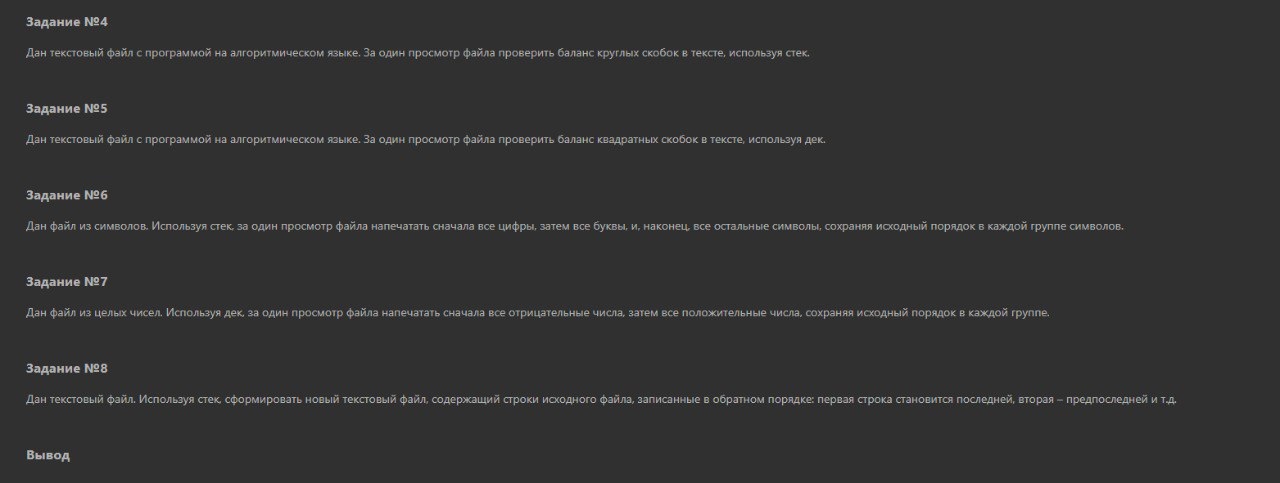
Вариант №16

Проверил: Аршинов Е.А.

Москва 2022

**Задание**





**Ход работы**

Реализация структуры стек. Стеком называется хранилище данных, в котором можно работать только с одним элементом: тем, который был добавлен в стек последним. Создаем класс Stack. В нем содержится конструктор с пустым списком. Логическая функция класса is\_empty проверяет на пустоту списка. Функция push добавляет значения в стек. Функция pop удаляет из начала списка. В этой функции также есть проверка на пустоту списка, т.е. если длина списка 0, то мы ничего не сможем добавить, а если нет, то удаляем последний элемент. Функция size возвращает длину стека. Функция print\_stack печатает стек.

Реализация структуры дек. Деком называется структура данных, в которую можно удалять и добавлять элементы как в начало, так и в конец. Создаем класс Deque. В нем содержится конструктор с пустым списком. Функции add\_first и add\_last добавляют значения в начало и конец дека соответственно. Здесь для того, чтобы добавить элементы мы присваиваем списку сумму списков из 1 элемента, который хотим добавить, и весь список. Функции remove\_first и remove\_last удаляют из начала и конца списка соответственно. Функции get\_first и get\_last возвращают из начала и конца списка соответственно, не удаляя элемент. Логическая функция класса is\_empty проверяет на пустоту списка. Функция size возвращает длину дека.

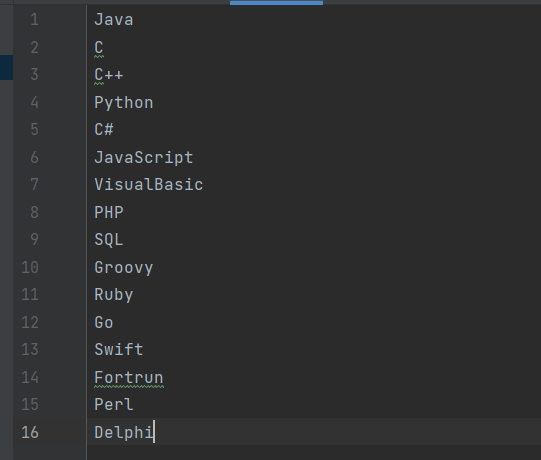
**Задание 1**

Используем в этом задании структуру дек. Будем сравнивать первый и последний элементы. Присваиваем их как 1-й и 2-й элементы. Идем от 1-го элемента до конца. Удаляем первый и последний элемент, затем сравниваем их. Наибольший элемент будет в конце. Потом берем этот элемент и присваиваем во второй дек.

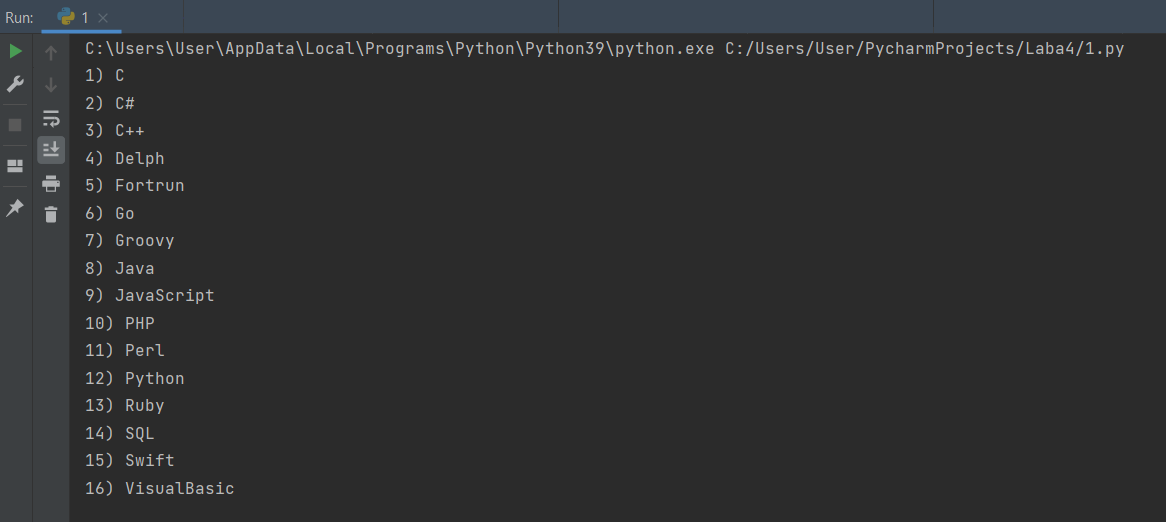
Код программы:

import Deque  
  
deque = Deque.Deque()  
res\_deque = Deque.Deque()  
with open('files/1.txt', 'r') as file:  
 # Filling the deque with content from file  
 for line in file:  
 deque.add\_last(line[:-1])  
  
 for i in range(deque.size()-1):  
 for j in range(deque.size()):  
 el1 = deque.remove\_first()  
 el2 = deque.remove\_last()  
 if el1 > el2:  
 deque.add\_last(el1)  
 deque.add\_last(el2)  
 else:  
 deque.add\_last(el2)  
 deque.add\_last(el1)  
 res\_deque.add\_last(deque.remove\_last())  
 res\_deque.add\_last(deque.remove\_last())  
  
 for i in range(res\_deque.size()):  
 print("{}) {}".format(i+1, res\_deque.remove\_first()))

Исходные данные:



Результат работы программы:



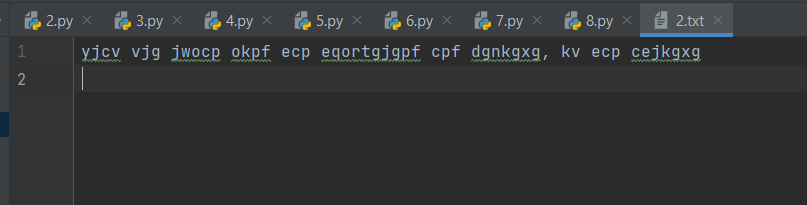
**Задание 2**

Используем в этом задании структуру дек. Есть 3 дека: с текстом, с ключом, с расшифрованным текстом. Также есть зашифрованный файл и дэк с ключом (т.е. с алфавитом). Расшифровка текста состоит в том, что мы удаляем элемент с конца и присваиваем его в начало, так через один элемент с конца и будет буква, которая шифровалась. Следовательно, нужно пройтись еще 2 раза, и мы букву, которая нам нужна. В цикле мы идем по алфавиту и присваиваем дек-ключу. Далее идем по файлу, читаем строку и каждый символ строки присваиваем в дек-текст. Так идем по всем символам в тексте. У нас есть текущий символ, который мы должны расшифровать, вытаскиваем его из дека и ищем, пока не найдем наш символ в ключах. Удаляем последний символ. Если наш текущий символ не равен текстовому, то идем дальше. Так мы идем пока не встретим наш символ. Если мы его нашли, то должны пройтись еще 2 раза. Последний раз не проходимся, а возвращаем последний символ и добавляем в расшифрованные.

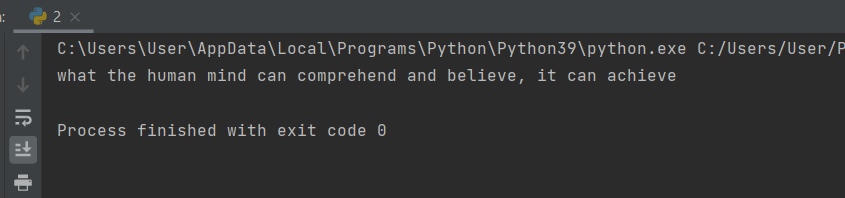
Код программы:

import Deque  
  
text = Deque.Deque()  
key = Deque.Deque()  
decoded\_text = Deque.Deque()  
  
alphabet = "abcdefghijklmnopqrstuvwxyz"  
for i in alphabet:  
 key.add\_last(i)  
  
  
with open('files/2.txt', 'r') as file:  
 for line in file:  
 text\_string = line[:-1]  
 for i in text\_string:  
 text.add\_last(i)  
  
 for i in range(text.size()):  
 text\_symbol = text.remove\_first()  
 if text\_symbol in [" ", "'", ",", "?"]:  
 decoded\_text.add\_last(text\_symbol)  
 continue  
 while True:  
 key\_symbol = key.remove\_last()  
 if key\_symbol != text\_symbol:  
 key.add\_first(key\_symbol)  
 else:  
 key.add\_first(key\_symbol)  
 key\_symbol = key.remove\_last()  
 key.add\_first(key\_symbol)  
 decoded\_text.add\_last(key.get\_last())  
 break  
  
 res\_string = ""  
 for i in range(decoded\_text.size()):  
 res\_string += decoded\_text.remove\_first()  
  
 print(res\_string)

Исходный файл:



Результат работы программы:



**Задание 3**

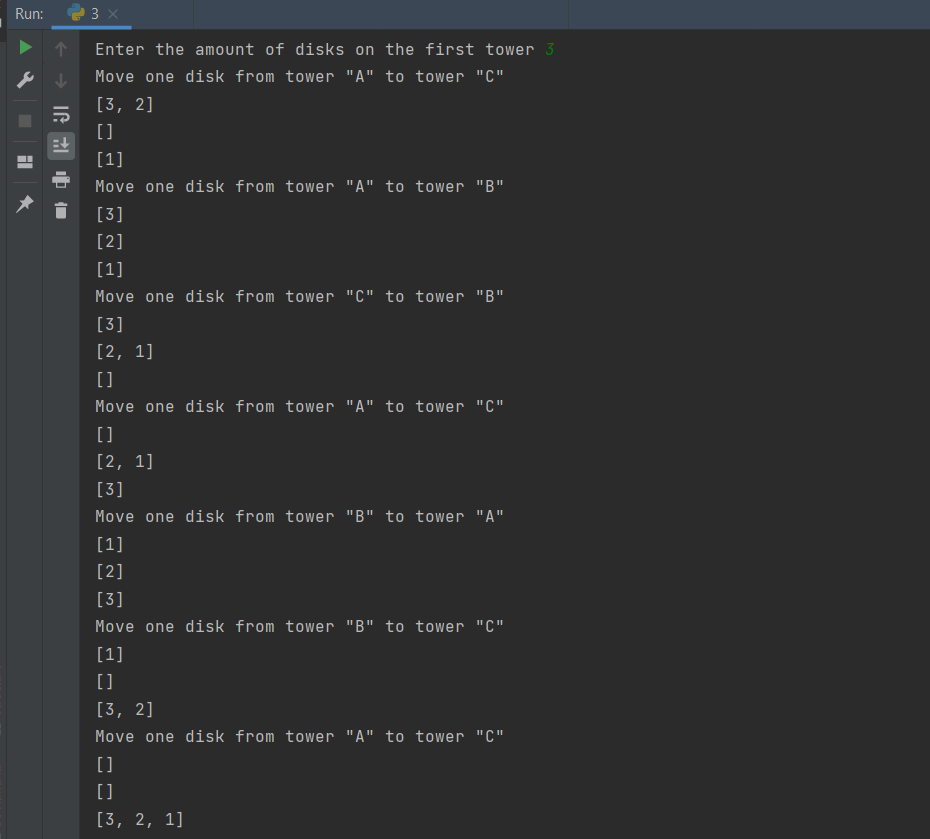
Используем в этом задании структуру стек. У нас есть функция move, переносящая 1 диск с начальной точки в конечную. Функция move\_tower используется, когда нужно перенести больше дисков (например, 3), тут уже используем 3 башни: начальную, конечную и промежуточную. Сначала мы переносим один диск с начальной на временную башню, потом последний с начальной на конечную башню. Потом вызываем функцию для переноса оставшихся дисков с временной на конечную.

Изначально создаем 3 стека с башнями. Первый заполняем количеством дисков. Потом вызываем функцию move\_tower с 3 башнями для перемещения дисков. Т.е. мы достаем элемент из башни, присваиваем его в текущий и потом добавляем в стек, который хотим переместить.

Код программы:

import Stack  
  
  
def move(tower\_from, tower\_to):  
 cur\_disk = None  
 if tower\_from == 'A':  
 cur\_disk = tower1.pop()  
 elif tower\_from == 'B':  
 cur\_disk = tower2.pop()  
 else:  
 cur\_disk = tower3.pop()  
  
 if tower\_to == 'A':  
 tower1.push(cur\_disk)  
 elif tower\_to == 'B':  
 tower2.push(cur\_disk)  
 else:  
 tower3.push(cur\_disk)  
  
 print("Move one disk from tower \"{}\" to tower \"{}\"".format(tower\_from, tower\_to))  
 tower1.print\_stack()  
 tower2.print\_stack()  
 tower3.print\_stack()  
  
  
def move\_tower(disks\_amount, tower1, tower2, temp\_tower):  
 if disks\_amount == 0:  
 return  
  
 move\_tower(disks\_amount - 1, tower1, temp\_tower, tower2)  
 move(tower1, tower2)  
 move\_tower(disks\_amount - 1, temp\_tower, tower2, tower1)  
  
  
tower1 = Stack.Stack()  
tower2 = Stack.Stack()  
tower3 = Stack.Stack()  
  
for disk in range(int(input("Enter the amount of disks on the first tower ")), 0, -1):  
 tower1.push(disk)  
  
move\_tower(tower1.size(), 'A', 'C', 'B')

Результат работы программы:



**Задание 4**

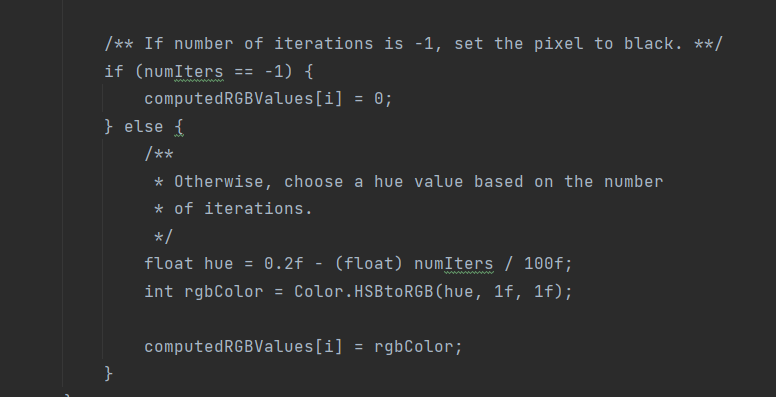
Используем в этом задании структуру стек. Идем по всем строкам, и когда встретим открывающуюся скобку, кладем ее в стек. Идем дальше, если нам встретилась закрывающаяся скобка, то идем в стек и смотрим, есть ли у нас там что-то (а там у нас должны быть только открывающиеся скобки). Если в стеке есть открывающаяся скобочка, то удаляем ее оттуда. В конце делаем проверку на баланс скобочек.

Код программы:

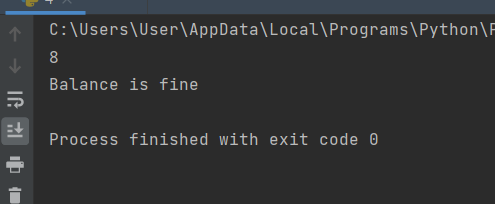
import Stack  
  
with open('files/4-5.txt', 'r') as file:  
  
 for line in file:  
 text\_string = line[:-1]  
  
 balance = Stack.Stack()  
 balance\_broken = False  
  
 for i in text\_string:  
 if i == "(":  
 balance.push(i)  
 elif i == ")":  
 cur\_symbol = balance.pop()  
 if cur\_symbol is None:  
 balance\_broken = True  
 break  
  
 if balance.size() != 0:  
 balance\_broken = True  
 break  
  
 if balance\_broken:  
 print("Balance is broken")  
 else:

print("Balance is fine")

Исходный файл:



Результат работы программы:



**Задание 5**

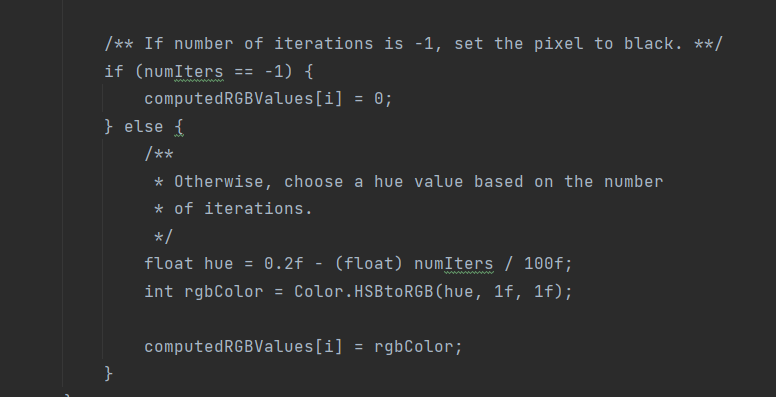
Такой же алгоритм, как в предыдущем задании, но тут работа с квадратными скобками.

Только используем в этом задании структуру дек. А вместо функций push и pop используем add\_first и remove\_first.

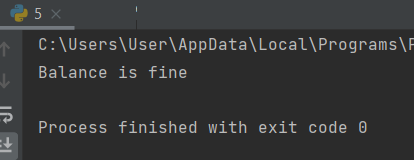
Код программы:

import Deque  
  
with open('files/4-5.txt', 'r') as file:  
  
 for line in file:  
 text\_string = line[:-1]  
  
 balance = Deque.Deque()  
 balance\_broken = False  
  
 for i in text\_string:  
 if i == "[":  
 balance.add\_first(i)  
 elif i == "]":  
 cur\_symbol = balance.remove\_first()  
 if cur\_symbol is None:  
 balance\_broken = True  
 break  
  
 if balance.size() != 0:  
 balance\_broken = True  
 break  
  
 if balance\_broken:  
 print("Balance is broken")  
 else:  
 print("Balance is fine")

Исходный файл:



Результат работы программы:



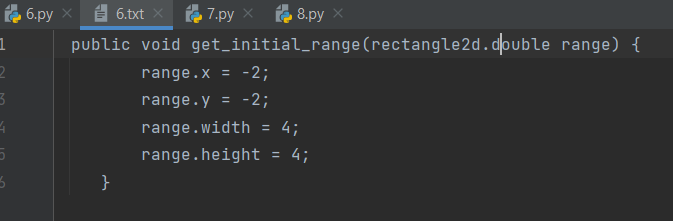
**Задание 6**

Используем в этом задании структуру стек. У нас есть общий стек (general\_stack) со всеми символами файла. А также 3 стека: для чисел, для букв и для символов. Сначала все символы присваиваем в general\_stack. Теперь идем по стеку, берем символ, присваиваем его в текущий и смотрим к чему он относится (к буквам, числам или символам). Когда определили, туда его и присваиваем. Мы постепенно идем по стеку, присваиваем все, что там есть и выводим это. В конце делаем 3 вывода.

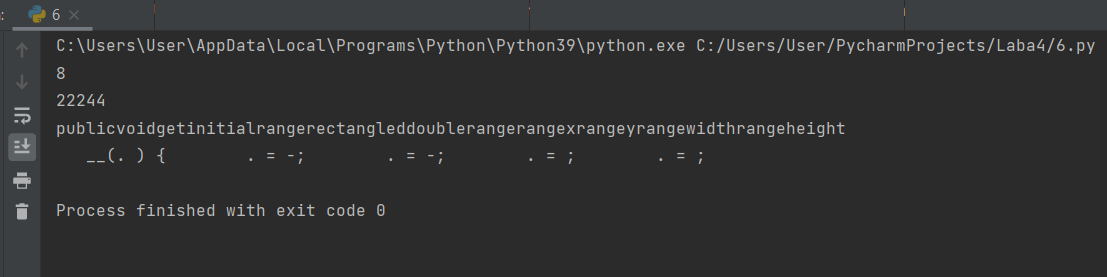
Код программы:

import Stack  
  
def isdigit(string):  
 for i in string:  
 if i not in "1234567890":  
 return False  
 return True  
  
def isletter(string):  
 string.lower()  
 for i in string:  
 if i not in "abcdefghijklmnopqrstuvwxyz":  
 return False  
 return True  
  
  
with open('files/6.txt', 'r') as file:  
  
 general\_stack = Stack.Stack()  
  
 numbers\_stack = Stack.Stack()  
 letters\_stack = Stack.Stack()  
 symbols\_stack = Stack.Stack()  
  
 for line in file:  
 text\_string = line[:-1]  
  
 for symbol in text\_string:  
 general\_stack.push(symbol)  
  
 for i in range(general\_stack.size()):  
 cur\_symbol = general\_stack.pop()  
  
 if isdigit(cur\_symbol):  
 numbers\_stack.push(cur\_symbol)  
 elif isletter(cur\_symbol):  
 letters\_stack.push(cur\_symbol)  
 else:  
 symbols\_stack.push(cur\_symbol)

Исходный файл:



Результат работы программы:



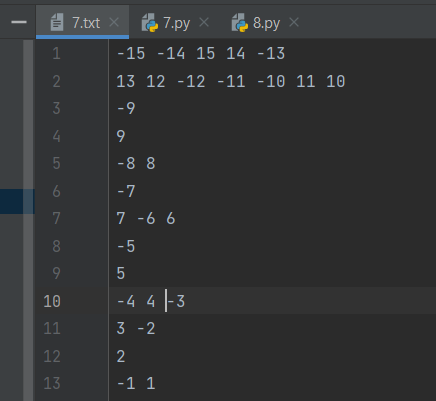
**Задание 7**

Используем в этом задании структуру дек. У нас есть начальный дек со всеми числами. Есть деки с положительными и отрицательными числами. Берем строчку, разделяем пробелами и присваиваем в text\_string. Далее идем и каждое число добавляем в general\_stack. Вытаскиваем оттуда по 1 числу, присваиваем его в текущий элемент и проверяем на знак. В конце выводим.

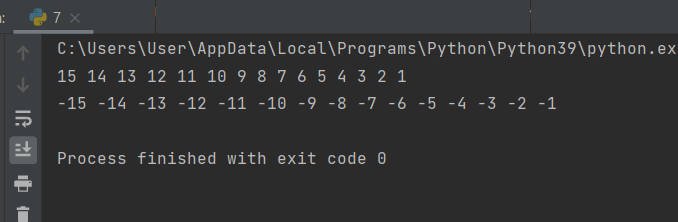
Код программы:

import Deque  
  
with open('files/7.txt', 'r') as file:  
  
 general\_stack = Deque.Deque()  
  
 positive\_numbers\_stack = Deque.Deque()  
 negative\_numbers\_stack = Deque.Deque()  
  
 for line in file:  
 text\_string = line[:-1].split(' ')  
  
 for number in text\_string:  
 general\_stack.add\_first(number)  
  
 for i in range(general\_stack.size()):  
 cur\_number = general\_stack.remove\_first()  
  
 if int(cur\_number) >= 0:  
 positive\_numbers\_stack.add\_first(cur\_number)  
 else:  
 negative\_numbers\_stack.add\_first(cur\_number)  
  
 positive\_numbers\_string = ""  
 for i in range(positive\_numbers\_stack.size()):  
 positive\_numbers\_string += positive\_numbers\_stack.remove\_first()  
 positive\_numbers\_string += " "  
 print(positive\_numbers\_string)  
  
 negative\_numbers\_string = ""  
 for i in range(negative\_numbers\_stack.size()):  
 negative\_numbers\_string += negative\_numbers\_stack.remove\_first()  
 negative\_numbers\_string += " "  
 print(negative\_numbers\_string)

Исходный файл:



Результат работы программы:



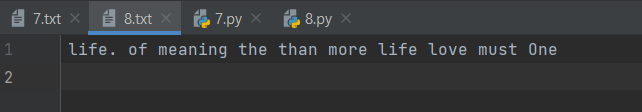
**Задание 8**

Используем в этом задании структуру стек. Открываем файл. Проходимся по всем строкам. В строку мы присваиваем все, что разделено пробелами (за исключение символа перехода на новую строку). Дальше каждое слово добавляем в general\_stack (заполняем из файла). У нас есть пустая строка. Идем по general\_stack и в эту пустую строку добавляем слово из файла (т.е. из general\_stack) и пробелы. Потом записываем в файл, указывая к нему путь.

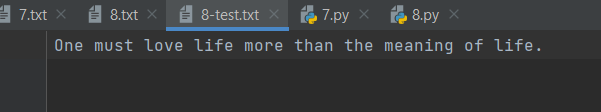
Код программы:

import Stack  
  
with open('files/8.txt', 'r') as file:  
  
 general\_stack = Stack.Stack()  
  
 for line in file:  
 text\_string = line[:-1].split(' ')  
  
 for word in text\_string:  
 general\_stack.push(word)  
  
 string = ''  
 for i in range(general\_stack.size()):  
 string += general\_stack.pop()  
 string += ' '  
  
 with open('files/8-test.txt', 'w') as f:  
 f.write(string)

Исходный файл:



Результат работы программы:



**Вывод**

В данной лабораторной работе научились работать с такими структурами данных как стек и дек.