Пояснительная записка к домашнему заданию №3 “Архитектура вычислительных систем с динамической типизацией”

Федяев Егор Александрович, БПИ202, вариант 67

Задание

Задание 11: “Различные числа”

Различные числа	1. Комплексные (действительная и мнимая части – пара действительных чисел) 2. Простые дроби (числитель, знаменатель – пара целых чисел) 3. Полярные координаты (угол [радиан] – действительное; координаты конечной точки на плоскости)	Приведение каждого значения к действительно му числу, эквивалентно му записанному. Например, для комплексного числа осуществляется по формуле: sqrt(d^2+i^2)), а для полярных координат - расстояние.

Функция 11: ”Сортировка по убыванию с помощью прямого выбора”

Упорядочить элементы контейнера по убыванию используя сортировку Сортировка с помощью прямого выбора (Straight Selection). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Файлы:

complex.py	631 Б
container.py	3 КБ
fraction.py	738 Б
main.py	937 Б
number.py	199 Б
point.py	75 Б
polar.py	870 Б

Также в программе присутствуют тестовые файлы

Формат запуска:

Генерация значений:

- python main.py -n [number of elements] [file out] [file out sorted]

Чтение значений из файла:

- python main.py -f [file in] [file out] [file out sorted]

Измерение времени работы (Linux):

- time python main.py -n [number of elements] [file out] [file out sorted]

Время работы:

Команды для контроля:

(venv) ➜ acs_3 git:(main) ✗ time python main.py -n 1000 test_06.txt sorted_test_06.txt

- python main.py -n 1000 test_06.txt sorted_test_06.txt 0.75s user 0.02s system 97% cpu 0.787 total

(venv) ➜ acs_3 git:(main) ✗ time python main.py -n 100 test_07.txt sorted_test_07.txt

- python main.py -n 100 test_07.txt sorted_test_07.txt 0.03s user 0.02s system 63% cpu 0.077 total

(venv) ➜ acs_3 git:(main) ✗ time python main.py -n 10 test_07.txt sorted_test_07.txt

- python main.py -n 10 test_07.txt sorted_test_07.txt 0.03s user 0.01s system 95% cpu 0.038 total

Таблица результатов:

Количество элементов	Время
1000	0.787 с
100	0.077 с
10	0.038 с

Отображение на память содержимого модулей

Память программы	Таблица имен	Память данных	
main.py			
def main	argv	list	[str]
	container	Container	class
	container.py	module	container.py
container.py			
def __init__	self	ref	Container
	storage	list	[Number]
def __str__	self	ref	Container
	result	list	[str]
def generate_test	self	ref	Container
	result	list	[str]
def read	self	ref	Container
	file	FileIO	class
	line	str	str
def write	self	ref	Container
	file	FileIO	class
def add_polar **def generate_polar**	self	ref	Container

	polar	Polar	class
	storage	list	[Number]
def add_complex **def generate_complex**	self	ref	Container
	complex	Complex	class
	storage	list	[Number]
def add_fraction **def generate_fraction**	self	ref	Container
	fraction	Fraction	class
	storage	list	[Number]
def sort	self	ref	Container
	storage	list	[Number]

Сравнительный анализ

Программа с использованием динамической типизации на языке программирования Python работает значительно медленнее, чем программы с использованием статической типизации, написанные с процедурным и объектно-ориентированным подходом. Однако, читаемость, краткость кода и скорость разработки значительно выше при написании на языке Python.