**Read Me для 3 домашнего задания по предмету «Архитектура вычислительных систем»**

**Выполнил: Сорокин Александр Игоревич БПИ202**

* **Вариант задания: 63**
* **Номер задачи: 7**
* **Номер функции: 5**

**Изображение выглядит как текст

Автоматически созданное описание**

Функция: 5. Упорядочить элементы контейнера по возрастанию используя сортировку Шелла (Shell Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

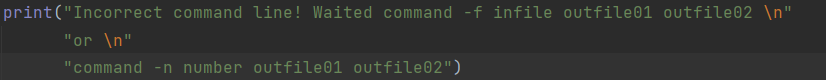
Общий вид изложения данных в тесте: (вес – целое число!)

1. <тип фильма> <имя> <год выхода> <доп. поле>

(Тип указывается цифрой: <Game> = 1 <Cartoon> = 2 <Documentary> 3)

1. Доп.поле для Game: 1 (Producer) – строка символов
2. Доп.поле для Cartoon: 1(DRAW), 2(PUPPET), 3(PLASTICINE)
3. Доп.поле для Documentary: 1 (Duration) – длительность в минутах

**Пример ввода:**



Структура программы с образными ассоциациями альтернативных фильмов

1. Отображение на память содержимого модуля main

**Память данных**

**Таблица имён**

**Память программы**

main.py

int: <number>

string: “…”

counter

str

file: fileName

ifile

string: “…”

inputFileName

string: “…”

string: “…”

outputFileName\_02

outputFileName\_01

float: <number>

module: container.py

container

startTime

1. Отображение на память остальных модулей и их методов

|  |  |  |  |
| --- | --- | --- | --- |
| Память программы | Таблица имён | Память данных | |
| container.py |  |  |  |
| \_\_init\_\_ |  | func | def… |
|  |  |  |  |
| def write() |  | func | def… |
|  | movie | module | Game, cartoon, documentary.py |
| def random\_write() |  | func | def… |
|  | movie | module | Game, cartoon, documentary.py |
| def shell() |  | func | def… |
|  | \_\_length | int | <number> |
|  | temp | list | <list> |
|  | s | int | <number> |
|  | j | int | <number> |
|  | I | int | <number> |
| game.py |  |  |  |
|  |  |  |  |
| def write() |  | func | def… |
|  |  |  |  |
| def random\_write() |  | func | def… |
|  |  |  |  |
| def task() |  | func | def… |
|  |  |  |  |
| def random\_print() |  | func | def… |
|  |  |  |  |
| readStrArray.py |  |  |  |
| read\_str\_array() |  | func | def… |
|  | array\_len | int | <number> |
|  | i | int | <number> |
|  | key | int | <number> |
|  | str | string | “…” |
|  | figNum | int | <number> |
|  | movie | module | Game, cartoon, documentary.py |
| game.py |  |  |  |
| \_\_init\_\_ |  | func | def… |
|  |  |  |  |
| read\_str\_array() |  | func | def… |
|  |  |  |  |
| write() |  | func | def… |
|  |  |  |  |
| random\_print() |  | func | def… |
|  |  |  |  |
| task() |  | func | def… |
|  | ans | int | <number> |
| Random\_write() |  | func | def… |
|  |  |  |  |
| cartoon.py |  |  |  |
| \_\_init\_\_ |  | func | def… |
|  |  |  |  |
| read\_str\_array() |  | func | def… |
|  |  |  |  |
| Random\_write() |  | func | def… |
|  |  |  |  |
| random\_print() |  | func | def… |
|  |  |  |  |
| task() |  | func | def… |
|  | ans | int | <number> |
| write() |  | func | def… |
|  |  |  |  |
| documentary.py |  |  |  |
| \_\_init\_\_ |  | func | def… |
|  |  |  |  |
| read\_str\_array() |  | func | def… |
|  |  |  |  |
| write() |  | func | def… |
|  |  |  |  |
| random\_write() |  | func | def… |
|  |  |  |  |
| task() |  | func | def… |
|  | ans | int | <number> |
| Random\_print() |  | func | def… |
|  |  |  |  |

**Характеристики программы**

Размер проекта вместе с тестами:



Число программных файлов проекта: 8

**# Sources**

* **game.py**
* **cartoon.py**
* **documentary.py**
* **rnd.py**
* **container.py**
* **readStrArray.py**
* **extender.py**
* **main.py**

**Особенности реализации проекта**

**Проект написан в объектно-ориентированной парадигме. Общие поля заполняются в контейнере сразу через конструктов, а доп. поля уже внутри специальных методов этих классов.**

**Сравнение с предыдущими проектами**

Замеры времени выполнения показали:

|  |  |
| --- | --- |
| **Количество фильмов** | **Время, сек** |
| 10 | Время, затраченное на выполнение данного кода = 0.0009965896606445312 |
| 100 | Время, затраченное на выполнение данного кода = 0.058377981185913086 |
| 1000 | Время, затраченное на выполнение данного кода = 51.061447858810425 |
| 10000 | Время, затраченное на выполнение данного кода = 339,495432 |

Замеры времени относительно предыдущих заданий:

|  |  |  |
| --- | --- | --- |
| **Количество фильмов** | **Время, сек**  **ООП python** | **Время, сек**  **ООП C++** |
| 10 | 0.0009965896606445312 | 0.000004432 |
| 100 | 0.058377981185913086 | 0.011000 |
| 1000 | 51.061447858810425 | 0.013000 |
| 10000 | 339,495432 | 0.359000 |

**Вывод**: программа, написанная на python 3.10 конечно же, уступает программе, написанной на с++ довольно существенно. Особенно сильно это начинает ощущаться, когда количество фильмов начинает превышать тысячу элементов. Программа на питоне слишком долго сортирует элементы, что доходит вплоть до нескольких минут, в то время как программа на с++ справляется практически моментально.

Зато объём занимаемой постоянной памяти у данной реализации намного меньше.