Ульяненков Ярослав БПИ204

**Домашнее задание №3**

Архитектура ВС с динамической типизацией

1. Описание задания:

|  |  |  |  |
| --- | --- | --- | --- |
| 11. 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 действительному числу, эквивалентному записанному. Например, для комплексного числа осуществляется по формуле: sqrt(d^2+i^2)), а для полярных координат - расстояние. |

9. Упорядочить элементы контейнера по возрастанию используя сортировку с помощью прямого слияния (Straight Merge). В качестве ключей для

сортировки и других действий используются результаты функции, общей для

всех альтернатив.

1. Структурная схема программы:

|  |  |  |
| --- | --- | --- |
| **Память программы** | **Таблица Имён** | **Типы данных** |
| main.py | start\_time | float |
| arguments | list[string] |
| container | Container |
| Container | cont | list[Number] |
| random\_fill | def |
| read\_numbers | def |
| print\_container | def |
| \_merge | def |
| \_sort | def |
| straight\_merge | def |
| Container.random\_fill | n | integer |
| i | integer |
| Container.read\_numbers | file | TextIO |
| n | integer |
| Container.print\_container | file | TextIO |
| index | integer |
| number | Number |
| Container.\_merge | a | list[Number] |
| left\_border | integer |
| divider | integer |
| right\_border | integer |
| left | integer |
| middle | integer |
| temp | list[Number] |
| i | integer |
| Container.\_sort | a | list[Number] |
| left\_border | integer |
| right\_border | integer |
| divider | integer |
| Container.straight\_merge | file | TextIO |
| numbers.\_\_init\_\_.py | get\_number | def |
| get\_random\_number | def |
| numbers.get\_number | file | TextIO |
| n | integer |
| x | integer |
| y | integer |
| numbers.get\_random\_number | number\_type | integer |
| min\_int | integer |
| max\_int | integer |
| x | integer |
| y | integer |
| Number | x | integer |
| y | integer |
| \_\_init\_\_ | def |
| decimal | def |
| \_\_str\_\_ | def |
| Complex | x | integer |
| y | integer |
| \_\_init\_\_ | def |
| decimal | def |
| \_\_str\_\_ | def |
| Complex.\_\_init\_\_ | a | integer |
| b | integer |
| Fraction | x | integer |
| y | integer |
| \_\_init\_\_ | def |
| decimal | def |
| \_\_str\_\_ | def |
| Fraction.\_\_init\_\_ | a | integer |
| b | integer |
| Polar | x | integer |
| y | integer |
| \_\_init\_\_ | def |
| decimal | def |
| \_\_str\_\_ | def |
| Polar.\_\_init\_\_ | a | integer |
| b | integer |

1. Основные характеристики программы

Размер исходных текстов – 44кб.

Число модулей реализации – 7.

Запуск программы с использованием созданных тестов производится следующей командой:

~$ python main.py -f <test> <output>

Где <executable> - имя исполняемого файла, <test> - путь до тестового файла, <output> - путь до файла, куда программа запишет результат своей работы.

Запуск программы с генерацией тестов производится следующей командой:

~$ python main.py -n <amount> <output>

Где <executable> - имя исполняемого файла, <amount> - количество элементов в контейнере, <output> - путь до файла, куда программа запишет результат своей работы.

Тестовые наборы являются текстовыми файлами, имеющими следующий формат: на первой строке размещено число n – количество элементов, после чего идёт n строк по три числа на каждой. Первое число – тип (1 – комплексное число, 2 – дробь, 3 – полярное число), второе и третье числа – числа, задающие требуемый тип.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тестового набора | Количество элементов | Используемая память, мегабайт | Время работы, миллисекунды |
| 0 | 843 | 5.1 | 16 |
| 1 | 4556 | 5.7 | 72 |
| 2 | 8281 | 6.6 | 130 |
| 3 | 4219 | 5.6 | 65.3 |
| 4 | 8341 | 6.7 | 132.6 |
| 5 | 3160 | 5.4 | 49.9 |
| 6 | 6206 | 6.1 | 100.1 |
| 7 | 9518 | 7.0 | 157.8 |
| 8 | 8247 | 6.7 | 131.58 |
| 9 | 2988 | 5.3 | 47.36 |
| 10 | 0 | 4.7 | 1.6 |

Для замера времени работы использовалась библиотека time. Для замера памяти использовалась стандартная утилита macOS «Мониторинг системы».

1. Сравнение с предыдущими реализациями
   1. Сравнение по памяти (килобайт):

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Первая реализация | Вторая реализация | Третья реализация |
| 0 | 368 | 480 | 5100 |
| 1 | 488 | 624 | 5700 |
| 2 | 608 | 680 | 6600 |
| 3 | 476 | 568 | 5600 |
| 4 | 600 | 716 | 6700 |
| 5 | 456 | 588 | 5400 |
| 6 | 536 | 676 | 6100 |
| 7 | 616 | 752 | 7000 |
| 8 | 600 | 712 | 6700 |
| 9 | 432 | 564 | 5300 |
| 10 | 324 | 432 | 4700 |

* 1. Сравнение по времени (мс):

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Первая реализация | Вторая реализация | Третья реализация |
| 0 | 4.6 | 5.31 | 16 |
| 1 | 18.3 | 21.7 | 72 |
| 2 | 29.6 | 32.4 | 130 |
| 3 | 13.7 | 19.7 | 65.3 |
| 4 | 28.3 | 35.7 | 132.6 |
| 5 | 15.8 | 20.2 | 49.9 |
| 6 | 20.5 | 27.6 | 100.1 |
| 7 | 44.1 | 37.1 | 157.8 |
| 8 | 26.6 | 33.7 | 131.58 |
| 9 | 9.6 | 16.18 | 47.36 |
| 10 | 1 | 1.2 | 1.6 |

Из таблиц видно, что последнее решение медленнее своих предшественников приблизительно в 2 - 4 раза и занимает в 10 - 14 раз больше памяти компьютера.