

Отчет

Пасилецкий Даниил Олегович

ФКН ПИ БПИ202. Вариант 55. Вариант задания 13. Вариант функции 4.

Программа выполнена на динамически типизированном языке Python 3.8.

Программа содержит следующие классы:

1. `tree` - дерево содержащие имя и возраст.
2. `bush` - кустарник содержит имя и месяц цветения.
3. `flowers` - цветок содержит имя и вид.
4. `plants` - базовый класс, в котором используется ключ для определения типа растения. 1 - `tree`; 2 - `bush`; 3 - `flowers`
5. `container` - место для хранения массива растений.

Согласно варианту 13 задания. Так же реализована шейкерная сортировка (Shaker Sort).

Число интерфейсных модулей: 0

Число модулей реализации: 6

Размер файлов исходного кода : 37 КБ

Размер скомпилируемого кода: 2800 КБ

Запуск программы

Запуск программы производится через консоль с указанием специальных аргументов:

`-f infile outfile01 outfile02` - ввод из файла *infile*, а вывод в *outfile01* - содержимое контейнера, а в *outfile02* - контейнер после сортировки.

`-n number outfile01 outfile02` - создание растений рандомом, `number` - число созданных растений. Вывод в `outfile01` - содержимое контейнера, а в `outfile02` - контейнер после сортировки.

Ввод в программу через файл осуществляется следующим способом:

- В начале вводится код элемента 1 - tree; 2 - bush; 3 - flowers.
- Затем через пробел (перенос строки) имя.
- Также через пробел индивидуальная характеристика в следующем формате:
 - для дерева количество лет (целое число)
 - для кустарника месяц цветения (целое число от 1 до 12 включительно)
 - для цветка его вид, так же целое число где:
 - 1 - домашний
 - 2 - садовый
 - 3 - дикий

После всего пустая строка НЕ допускается

Вывод программы

Программа выводит файлы в 2 файла указанные при запуске, в первом файле находится не отсортированный массив, где указаны характеристики каждого растений. Во втором файле находится уже отсортированный массив.

С целью демонстрации двух способов вывода перечисления, в `flowers` оно выводится словами, а в `bush` номером месяца.

Так как Python обладает динамической типизацией, размер который переменная занимает в памяти точно определить нельзя. В главе тип написан предполагаемый тип.

Таблица типов

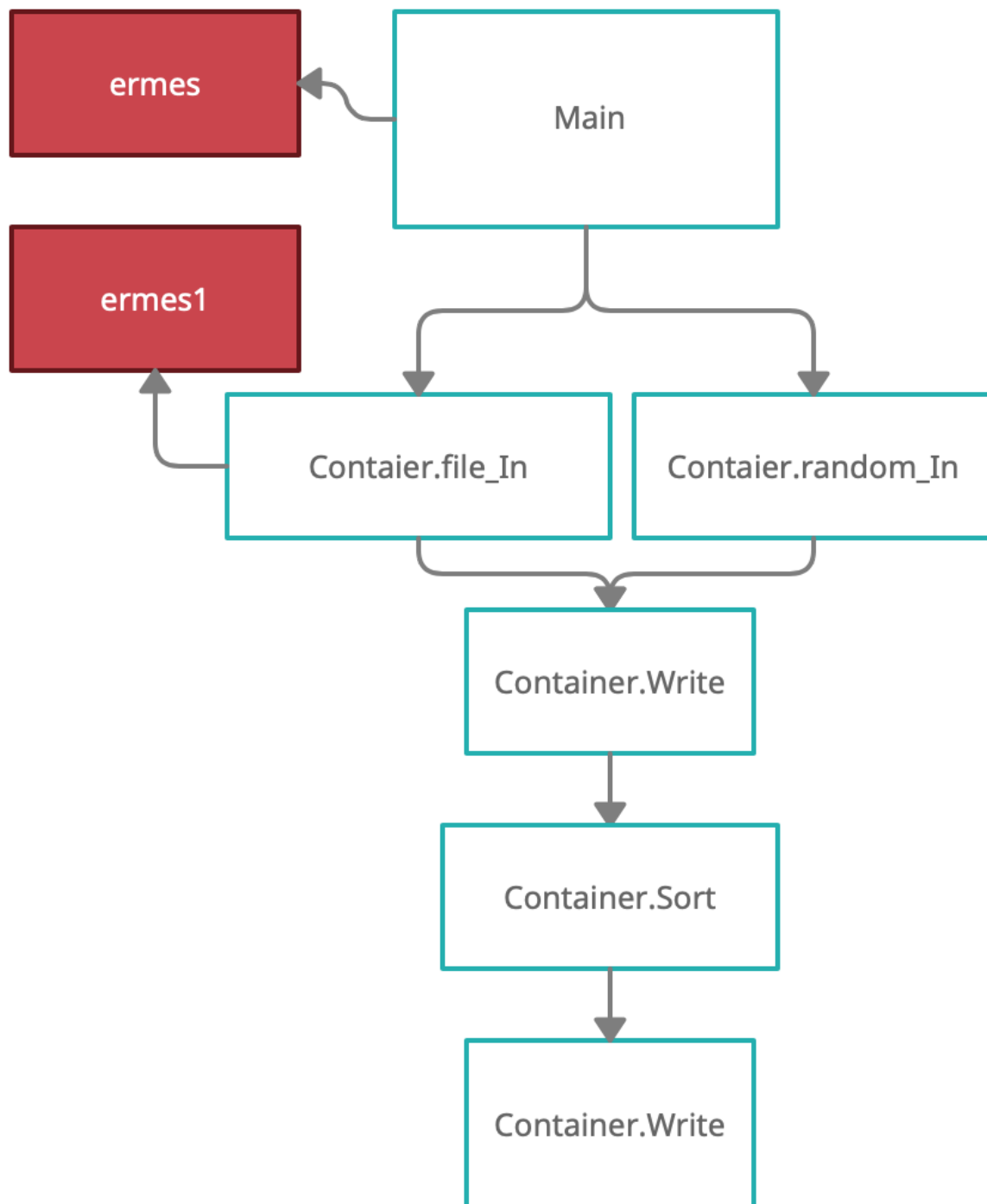
 Класс	 Имя	 Тип	 Размер
---	---	---	--

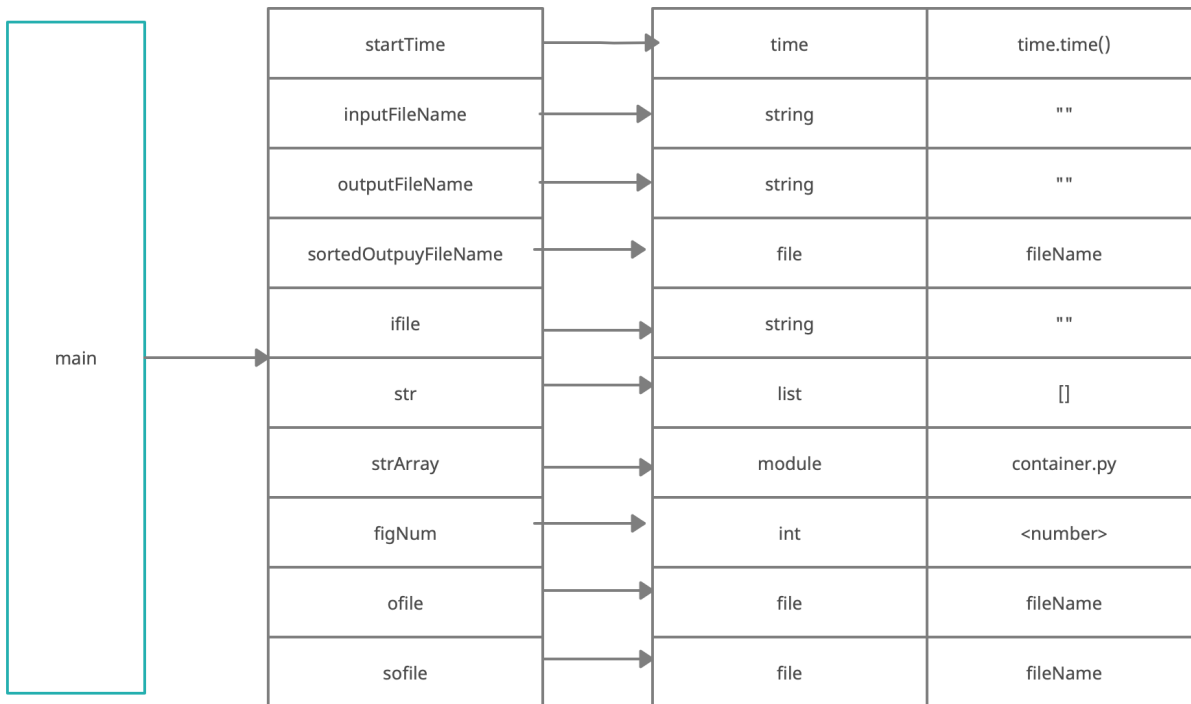
Аа Класс	☰ Имя	☰ Тип	▼ Размер
<u>Plants</u>	name	string	dynamic size
<u>Bush</u>	monthes	Monthes(enum)	dynamic size
<u>Tree</u>	age	int	dynamic size
<u>Flowets</u>	typeof	Typeof(enum)	dynamic size
<u>Container</u>	store	list[]	dynamic size

Таблица классов

Аа Класс	☰ Имя
<u>Plants</u>	def __init__(self name) def quotient(self)
<u>Bush</u>	def Print(self) def Write(self ostream) def __init__(self name monthes)
<u>Tree</u>	def Print(self) def Write(self ostream) def __init__(self name age)
<u>Flowers</u>	def Print(self) def Write(self ostream) def __init__(self name typeof)
<u>Container</u>	def Print(self) def Sort(self) def Write(self ostream) def __init__(self) def file_in(self strArray) def random_in(self figureNimbers)




Схема





Сору of Время работы программы

<u>Aa</u> Тест	<u>≡</u> Размер	<u>⋮</u> Время в секундах
----------------	-----------------	---------------------------

 Тест	 Размер	 Время в секундах
<u>test01</u>	10	0.0014658
<u>test02</u>	100	0.016716
<u>test03</u>	1000	1.015269
<u>test04</u>	5000	22.6628621
<u>test05</u>	9999	89.0690303

Вывод

Код на Python работает гораздо медленней, чем более низкоуровневый C++. Но на python гораздо удобней писать, не надо думать об утечках памяти и типизации. Так же на написание кода тратится намного меньше времени и на много меньше строчек кода.