Пояснительная записка к третьему домашнему заданию по АВС

Подюков Иван Владимирович, БПИ207, вариант 259

Номер задачи – 7, номер дополнительной функции – 19

Задача:

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив переменные	Общие для всех альтернатив функции
7. Фильмы	1. Игровой (режиссер – строка символов) 2. Мультфильм (способ создания – перечислимый тип = рисованный, кукольный, пластилиновый…) 3. Документальный фильм (длительност в минутах – целое)	1. Название фильма – строка символов. 2. Год выхода - целое	Частное от деления года выхода фильма на количество символов в названии (действительное число)

Дополнительная функция:

19. Удалить из контейнера те элементы, для которых значение, полученное с использованием функции, общей для всех альтернатив, меньше чем среднее арифметическое для всех элементов контейнера, полученное с использованием этой же функции.

Отображение содержимого всех классов:

Таблица классов
Container
Movie
Fiction
Cartoon
Science

Таблица имён	Описание	
store	list	[…]
__init__	func	def…
RandomIn	func	def…
Write	func	def…
SumOfQuotients	func	def…
DeleteElementsWithQuotientLessThanAverage	func	def…

Таблица имён	Описание	
year	int	<number>
name	string	«…»
__init__	func	def…
RandomIn	func	def…
Write	func	def…
Quotient	func	def…
ReadStrArray	func	def..

Таблица имён	Описание	
producer	string	«…»
__init__	func	def…
RandomIn	func	def…
Write	func	def…
ReadStrArray	func	def..

Таблица имён	Описание	
type_of_cartoon	string	«…»
__init__	func	def…
RandomIn	func	def…
Write	func	def…
ReadStrArray	func	def…

Таблица имён	Описание	
length_of_movie_in_minutes	int	<number>
__init__	func	def…
RandomIn	func	def…
Write	func	def…
ReadStrArray	func	def..

Отображение на память методов классов:

Память программы	Таблица имён	Память данных	
main.py	container	Container	container.py
	outputFileName	string	«…»
	number	int	<number>
	inputFileName	string	«…»
	ifile	file	fileName
	str	string	«…»
	strArray	list	[…]
	movieNum	int	<number>
	ofile	file	fileName
	container.py	module	extender.py
Container.__init__	self	Container	container.py
Container.RandomIn	self	Container	Container.py
	number	int	<number>
ReadStrArray	container	Container	container.py
	strArray	list	[…]
Container.Write	self	Container	container.py
	ostream	file	fileName
Container. DeleteElementsWithQuotientLessThanAverage	self	Container	container.py
Container.Write	self	Container	container.py
	ostream	file	filename

Общая информация о программе:

Число интерфейсных модулей (заголовочных файлов) – 1

Число модулей реализации – 7

Общий размер исходных текстов – 9,8 КБ (в прошлом задании было 15,4 КБ), 276 (в прошлом задании было 496) строчек кода.

Исполняемого кода нет (в прошлом задании был, и он весил 70,4 КБ)

Тесты	Время выполнения программы для различных тестовых наборов данных (в скобках время в предыдущем задании)
test1.txt	0.0005955 (0.0010345)
test2.txt	0.0006849 (0.0011554)
test3.txt	0.0006254 (0.0010337)
test4.txt	0.0007151 (0.0020002)
test5.txt	0.0009313 (0.0034175)

Сравнение с предыдущей версией программы:

Размер исполняемого кода (при использовании динамически типизированного языка исполняемого кода нет) и общий размер исходных текстов при использовании статически типизированного языка больше. На таких же тестах программа при использовании динамически типизированного языка работает быстрее, чем при статически типизированном примерно в 2–3 раза. Но при генерации 2000 объектов данная программа тратит 0.3960066 секунд, а предыдущая 0.0409959 секунд. В целом можно сказать, что программа при использовании динамически типизированного языка на малых тестах работает быстрее, а на больших – намного медленнее, чем при использовании статически типизированного языка. Писать код при использовании динамически типизированного языка намного быстрее и проще.