Домашнее задание 3
Рахматуллин Карим БПИ207

Описание задания(задание 2, функция 19):

Разработать программный продукт с использованием динамической проверки типов во время выполнения программы. Программа должна содержать следующие структуры:

-Обобщенный артефакт, используемый в задании: Язык программирования

-Базовые альтернативы:

1. Процедурные (наличие, отсутствие абстрактных типов данных – булевская величина)

2. Объектно- ориентированные (наследование: одинарное, множественное, интерфейса – перечислимый тип)

3. Функциональные языки (типизация – перечислимый тип = строгая, динамическая; поддержка «ленивых» вычислений – булевский тип)

Для всех альтернатив общей переменной является популярность в процентах(действительное) и год создания(целое). Общей функцией всех альтернатив выступает вычисление частного от деления года создания на популярность (действительное число). В качестве дополнительной функции контейнера необходимо удалять из него элементы, отношение года создания к популярности которых меньше чем среднее арифметическое отношений всех элементов контейнера (остальные элементы передвигать к началу контейнера с сохранением порядка).

Также нужно: разработать тестовые входные данные и провести тестирование и отладку программы на этих данных; описать структуру используемой ВС с наложением на нее обобщенной схемы разработанной программы; зафиксировать количество заголовочных файлов, программных файлов, общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.

Таблица типов(все размеры в байтах)

Название	Размер	Название	Размер
int	28	Class ObjectOriented	100
bool	28	inheritance_type: enum class Inheritance	48[0]
float	24	popularity: float	24[48]
IO	208	founding_year: int	28[72]
str	40	Class Functional	124
enum class Typization	48	type: enum class Typization	48[0]
enum class Inheritance	48	lazy: bool	28[48]
time	24	popularity: float	24[72]

Название	Размер	Название	Размер
Class Procedural	80	founding_year: int	28[96]
abstract: bool	28[0]	Class Container	254
popularity: float	24[28]	storage: Language[]	254[0]
founding_year: int	52[28]		

Описание работы функции main:

Память программы
if __name__ == '__main__': {…}

Память данных	
argv: str	96[0]
start_time: time	24[96]
container: Container	254[120]
inputFileName: str	40[374]
ifile: IO	208[414]
input_string: str	40[622]
languageNum: int	28[662]
size: int	28[690]
ofile: IO	208[718]

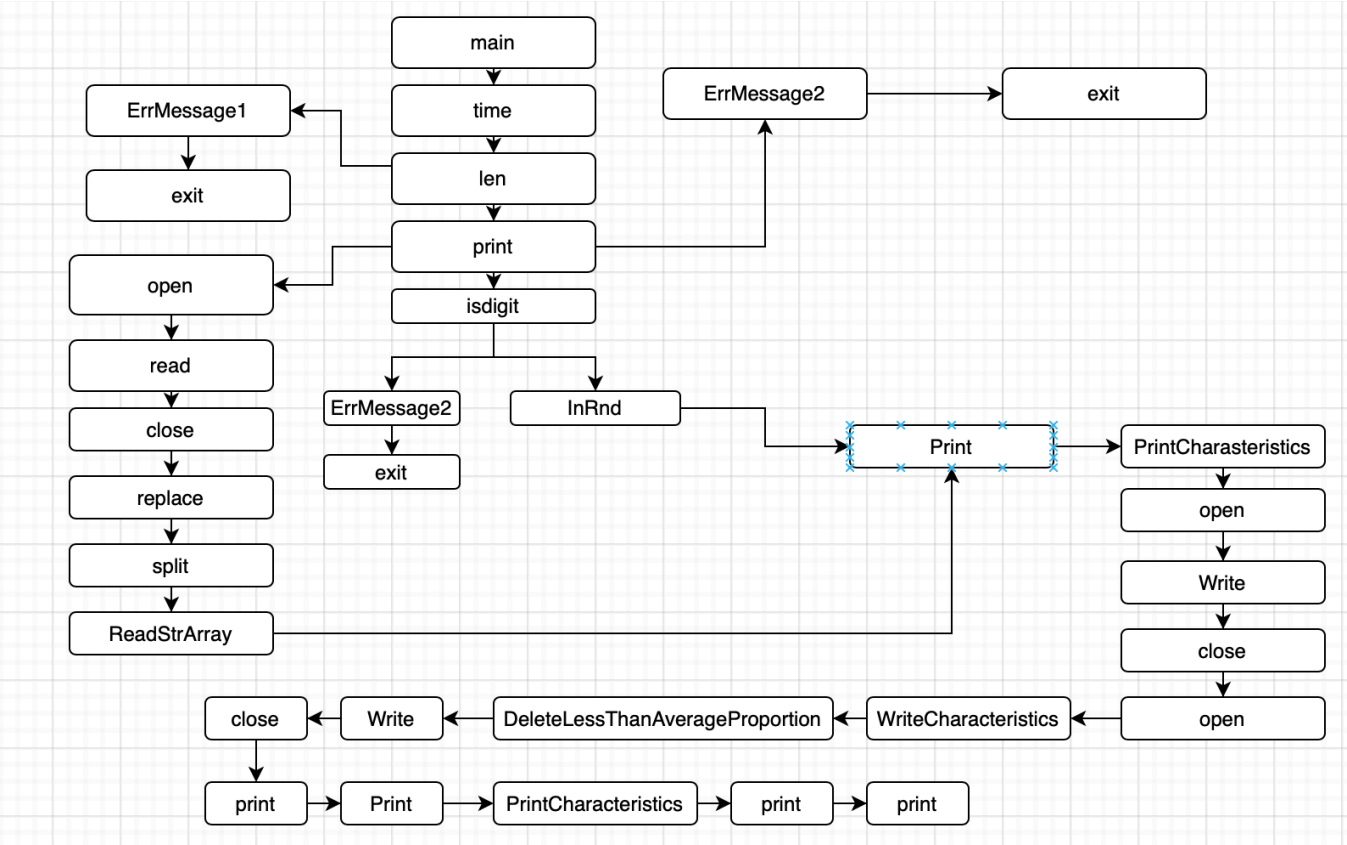
Глобальная память
0

Heap
«main» - 0
«-f» - 1
«test01.txt» - 2
«out1.txt» - 3
«out2.txt» - 4

Stack
ReadStrArray
Print
PrintCharacteristics
Write
WriteCharacteristics
DeleteLessThenAverage Proportion
Write
Print
PrintCharacteristics

В таблице стек представлен для частного случая (загрузка данных из файла) и содержит только созданные в процессе написания программы функции).

Блок схема стека, в результате работы функции main (глубина - 1 шаг):



Описание работы функции InRnd

Stack
InRnd
randint
uniform
randint
randint
Append

Память данных	
key: int	28[0]
language: Procedural	80[28]
language: Functional	124[108]
language: ObjectOriented	100[232]

Глобальная память
0

Heap
Language[0]

Память программы

@staticmethod
def InRnd(input_container, size):
  for i in range(size):
  key = random.randint(1, 3)
  if key == 1:
  language = procedural.Procedural()
  language.popularity = random.uniform(0, 1) * 100
  language.founding_year = 1990 + random.randint(0, 120)
  language.is_abstract = random.randint(0, 1) == 1
  elif key == 2:
…

Описание работы функции DeleteLessThanAverageProportion

Stack
DeleteLessThanAverageProportion
len
Proportion
Proportion
pop

Память данных	
i: int	28[0]
elements_cnt: int	28[28]
average_value: float	24[56]

Heap

Глобальная память
0

Память программы

def DeleteLessThanAverageProportion(self):
  i = 0
  elements_cnt = len(self.store)
  average_value = self.Proportion() / elements_cnt
  while elements_cnt > 0 and i < elements_cnt:
  if self.store[i].Proportion() < average_value:
  self.store.pop(i)
  i -= 1
  elements_cnt -= 1
  i += 1

Описание работы функции Proportion

Stack
Proportion
Proportion

Память данных	
total_proportion: double	24[0]

Heap

Глобальная память
0

Память программы

```
def Proportion(self):
    total_proportion = 0.0
    for language in self.store:
        total_proportion += language.Proportion()
    return total_proportion
```

Основные характеристики программы

- Число заголовочных файлов: 0
- Число модулей реализации: 8
- Общий размер исходных текстов - 401 строк кода
- Размер исполняемого кода -32 кб
- Время выполнения программы:
  - Тест 1 - 0.002894 с
  - Тест 2 - 0.002102 с
  - Тест 3 - 0.000535 с
  - Тест 4 - 0.002343 с
  - Тест 5 - 0.003642 с

	Процедурный подход и статическая типизация	Объектно-ориентированный подход и статическая типизация	Динамическая типизация
Число заголовочных файлов	8	7	1
Число модулей реализации	7	7	8
Общий размер исходных текстов(строк кода)	679	654	401
Размер исполняемого кода(кб)	53	57	32
Тест 1(с)	0.000460	0.000452	0.002894
Тест 2(с)	0.001523	0.000565	0.002102
Тест 3(с)	0.000436	0.000701	0.000535
Тест 4(с)	0.000844	0.000858	0.002343
Тест 5(с)	0.000627	0.000643	0.003642

Выводы

Динамическая типизация сказалась на времени работы программы. Она работает медленнее, чем предыдущие варианты. Но количество строк кода так же уменьшилось и код стал более читаем. Также я использовал процедурный подход, поэтому скорость работы снизилась. ООП подход остается предпочтительным вариантом.