**Описание полученного задания**

Необходимо написать программу, оформленную в виде консольного приложения. Запуск программы осуществляется из командной строки, в которой указываются имя запускаемой программы, имя файла с исходными данными, имя файла с выходными данными.

Для каждого программного объекта, загружаемого в контейнер, исходный файл с тестовым набором должен содержать признак альтернативы, а также список параметров, необходимых альтернативе. В выходной файл необходимо вывести введенные в контейнер данные. Помимо этого, необходимо вывести информацию об общем количестве объектов, содержащихся в контейнере. После этого в тот же файл необходимо вывести новые данные в соответствии с результатами, полученными в ходе работы программы.

Основное задание для составления программы – создание обобщенного артефакта – языки программирования – со следующими базовыми альтернативами, имеющими уникальные параметры: процедурные языки программирования (уникальные параметры: наличие/отсутствие абстрактных типов данных [булевская величина]), объектно-ориентированные языки программирования (уникальные параметры: виды наследования: одинарное, множественное, интерфейс [перечислимый тип]), функциональные языки программирования (уникальные параметры: типизация: строгая, динамическая [перечислимый тип]; поддержка «ленивых» вычислений [булевская величина]). Все базовые альтернативы имеют общие переменные: название [строка символов], популярность в процентах [действительное число], год создания [целое число]. Также все базовые альтернативы имеют общую для всех функцию: частное от деления года создания на количество символов в названии.

После размещения данных в контейнер необходимо упорядочить элементы контейнера по убыванию, используя сортировку с помощью прямого слияния (Straight Merge). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив. В программе должны поддерживаться следующие операции: заполнение контейнера данными, поступающими из входного потока (файла); вывод значений всех элементов в выходной поток (файл), выводятся параметры элементов, размещенных в контейнере.

**Таблица типов**

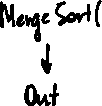
|  |  |
| --- | --- |
| **Тип данных** | **Размер** |
| int | 4 |
| float | 4 |
| char | 1 |
| struct language | 117 |
| key:k | 4 [0] |
| union:анонимный | 113 [4] |
| struct procedural | 109 |
| bool:abstract\_data\_types | 1 [0] |
| int:creat\_year | 4 [1] |
| float popularity\_perc | 4 [5] |
| char[]:name[100] | 100\*1=100 [9] |
| struct functional | 113 |
| tipization:k | 4 [0] |
| bool:lazy\_calculations | 1 [4] |
| int:creat\_year | 4 [5] |
| float popularity\_perc | 4 [9] |
| char[]:name [100] | 100\*1=100 [13] |
| struct object-oriented | 112 |
| inheritance:k | 4 [0] |
| int:creat\_year | 4 [4] |
| float popularity\_perc | 4 [8] |
| char[]:name[100] | 100\*1=100 [12] |
| struct:container | 1170008 |
| enum: анонимный | 4 [0] |
| int:len | 4 [4] |
| language\*:cont[max\_len] | 10000\*117 [8] |

Глобальная память не используется.

**Таблица данных функции main**

|  |  |
| --- | --- |
| **Данные** | **Размер** |
| int:argc | 4 [0] |
| char\*[]:argv | 1 [4] |
| container:c | 40004 [5] |
| FILE\*:infile | 4 [40009] |
| int:size | 4 [40013] |
| FILE\*:outfile1 | 4 [40017] |
| FILE\*:outfile2 | 4 [40021] |

**Стек**



**Таблица данных функции InRnd(functional &r)**

|  |  |
| --- | --- |
| **Данные** | **Размер** |
| functional:r | 116 [0] |
| int:length | 4 [116] |
| int:key | 4 [120] |

**Таблица данных функции In(container &c, FILE\* infile)**

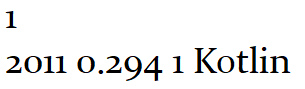
|  |  |
| --- | --- |
| **Данные** | **Размер** |
| container:c | 116 [0] |
| FILE\*:infile | 4 [116] |
| int:l | 4 [120] |

**Таблица данных функции Merge(container \*c, int first, int last)**

|  |  |
| --- | --- |
| **Данные** | **Размер** |
| container:c | 4 [0] |
| int:first | 4 [8] |
| int:last | 4 [12] |
| int:middle | 4 [16] |
| int:start | 4 [20] |
| int:final | 4 [24] |
| int:j | 4 [28] |
| container\*:mas | 4 [32] |

**Требуемые метрики, определяющие характеристики программы, для различных тестовых прогонов:**

*Пример* тестовых входных данных:



*Замечание*: в первой строке указывается номер базовой альтернативы в диапазоне от 1 до 3 (1 – процедурный язык программирования; 2 – объектно-ориентированный язык программирования; 3 – функциональный язык программирования). В следующей строке через пробел вводятся параметры альтернативы (уникальные для каждой) в следующем порядке:

1. *процедурный язык программирования*: год создания (целочисленный тип в диапазоне от 1940 до 2020), процент популярности (вещественный тип в диапазоне от 0 до 100), наличие абстрактных типов данных (булевский тип, который может принимать значение 1 – true, 0 – false), имя (строка, начинающаяся с заглавной буквы, не превышающая длину 100).
2. *объектно-ориентированный язык программирования*: вид наследования (целочисленный ключ в диапазоне от 1 до 3, где 1 – одиночное наследование, 2 – множественное наследование, 3 – наследование интерфейса), год создания (целочисленный тип в диапазоне от 1940 до 2020), процент популярности (вещественный тип в диапазоне от 0 до 100), имя (строка, начинающаяся с заглавной буквы, не превышающая длину 100).
3. *функциональный язык программирования*: вид типизации (целочисленный ключ в диапазоне от 1 до 2, где 1 – строгая типизация, 2 – динамическая типизация), год создания (целочисленный тип в диапазоне от 1940 до 2020), процент популярности (вещественный тип в диапазоне от 0 до 100), наличие «ленивых» вычислений (булевский тип, который может принимать значение 1 – true, 0 – false), имя (строка, начинающаяся с заглавной буквы, не превышающая длину 100).

В примере приведен процедурный язык программирования, год создания 2011, процент популярности = 0.294, наличие абстрактных типов данных = true, название – Kotlin.

Все составленные тесты лежат вместе с .exe файлом программы (test1.txt, test2.txt, test3.txt, test4.txt, test5.txt). Файлы для вывода результатов работы программы – result.txt.out и result2.txt.out

**Основные характеристики программы**

Время работы тестовых данных:

1. **Тест 1:**

*2*

*2 1991 9.7 Python*

*3*

*2 1958 0.3 1 Lisp*

*1*

*2011 0.294 1 Kotlin*

Время работы = 3ms

1. **Тест 2:**

*3*

*1 2005 0.232 1 F#*

*2*

*2 1991 9.7 Python*

*2*

*1 2000 5.3 C#*

*3*

*2 2004 0.378 1 Scala*

Время работы = 5ms

1. **Тест 3:**

*1*

*2009 0.9 1 Go*

*2*

*1 2000 5.3 C#*

*1*

*2011 0.294 1 Kotlin*

*3*

*2 2004 0.378 1 Scala*

Время работы = 4ms

1. **Тест 4**

*1*

*2011 0.294 1 Kotlin*

*2*

*1 2000 5.3*

*C#*

Время работы = 6ms

1. **Тест 5**

*3*

*2 2004 0.378 1 Scala*

*3*

*1 2005 0.232 1 F#*

*2*

*2 1991 9.7 Python*

*3*

*2 1958 0.3 1 Lisp*

*1*

*2011 0.294 1 Kotlin*

Время работы = 18ms

Время работы для рандомного формирования входынх данных в зависимости от их количества:

1: 1ms

2: 3ms

3: 3ms

4: 3ms

5: 5ms

Число интерфейсных модулей = 6

Число модулей реализации = 6

Общий размер исходных текстов = 31Кб

Полученный размер исполняемого кода = 50Кб