Сибирцева Анна
БПИ206

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Вариант: **222**
Задание: **12**
Функция: **16**

Описание задачи:

Необходимо реализовать программу, в которой обобщенным артефактом является животное, а базовыми альтернативами: рыбы (поле – место проживания – перечислимый тип: река, озеро, море, пруд, океан), птицы(параметр – отношение к перелёту – перелётные или остающиеся на зимовку), звери(поле – тип питания – хищники, травоядные, насекомоядные, всеядные).

Общим для все альтернатив являются: имя животного/его название – строка символов, вес животного в граммах – целое число.

Общей для всех альтернатив функцией является нахождение частного от деления суммы кодов незашифрованной строки на вес, являющиеся действительным числом.

Обработка данных является их упорядочивание по убыванию, с использованием сортировки методом деления по пополам (*Binary Search*).

Основные характеристики программы:

Число подпрограмм: **27**
Число подключенных программ: **12**
Общий размер исходных текстов: **~54,68 Кб**
Размер исполняемого файла: **390 КБ**
Время работы на тестах:

TEST 1: **0.008 с**
TEST 2: **0.011 с**
TEST 3: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 4: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 5: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 6: **0.279 с**
TEST 7: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 8: **0.044 с**

NB: 4 тест (где имена животных – символы/смайлики) в данной реализации не работает, т.к. неправильно читается кодировка символов. Так же тест 7 на очень длинное имя теперь не работает, ведь мы ввели ограничение на кол-во символов.

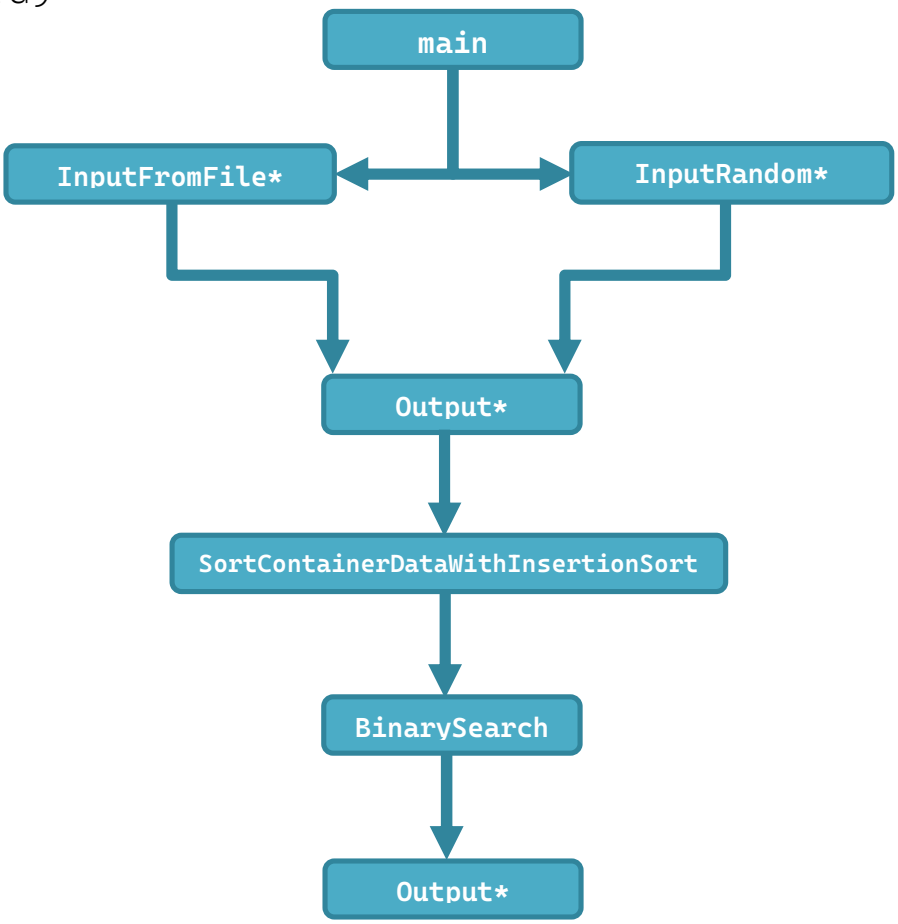
Дополнительная информация:

Программа была выполнена с использованием языка ассемблера NASM (Netwide Assembler).

Подробная информация о корректном использовании находится в файле README.txt

Ниже представлен ПРИБЛИЗИТЕЛЬНЫЙ стек вызовов:

Звездочками помечены методы, которые вызываются более одного раза или различаются в зависимости от параметров. Названия так же аналогично меняются в зависимости от того, с каким объектом работает программа (например, для работы с птицами будут вызваны методы: InputFromFileBird, InputRandomBird и OutputBird)



Сравнение программ:

Это четвертая программа выполненная по данному заданию. В этой версии основным языком программирования был ассемблер NASM (Netwide Assembler).Сравнив четыре отчета, можно выделить следующие отличия:

1. Программа работает гораздо быстрее на больших тестах (генерация 10 000 элементов).
2. Созданный .exe файл занимает больше всего памяти, в сравнении с другими реализациями.
3. Надо было дополнительно учитывать кодировку для чтения из файла, так же, как и на питоне, т.к. иначе тест с необычными символами считается некорректным.

Из отличий, которые появились благодаря использованию ассемблера можно выделить следующее:

Во-первых, программа выполняется быстрее остальных на больших данных, в остальном результаты показали похожие результаты с реализацией на C++ в стиле C (т.е. самой перовой реализацией) т.к. любое преобразование языка высокого уровня в машинный код приводит к издержкам. Ассемблер работает быстрее, потому что программист не пишет ничего лишнего.

Во-вторых, чисто по моему мнению, код стал нечитаемым. Потому что все высоко уровневые языки всё-таки похожи чем-то по синтаксису и если Python программист, не знающий C#, посмотрит на код, то спокойно разберется в основной сути. Здесь же такое невозможно.

В-третьих, тест с длинными именами теперь является некорректным, т.к. для простоты были введены ограничения на длину слова.

Делая выводы из всего перечисленного выше, можно сказать, что данная реализация определено является более эффективной на больших данных, но тем не менее самой сложной для разработки. Читаемость и понятность кода крайне низкая.