Архитектуры Вычислительных систем.

Пояснительная записка.

Задание №2.

Вариант 12. Функция 13.

Исполнил: студент БПИ205

Морозова А. В.

Москва 2021

1. Описание задания:

Обобщенный артефакт, используемый в задании: животные. Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив): 1. Рыбы (место проживания – перечислимый тип: река, море, озеро…) 2. Птицы (отношение к перелету: перелетные, остающиеся на зимовку – булевская величина) 3. Звери (хищники, травоядные, насекомоядные… – перечислимый тип). Общие для всех альтернатив переменные: 1. Название – строка символов, 2. Вес в граммах (целое). Функция, общая для всех альтернатив: найти частное от деления суммы кодов символов в названии животного на вес (действительное число).

Функция (13): упорядочить элементы контейнера по убыванию, используя шейкерную сортировку (Shaker Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

1. Описание структуры ВС (архитектуры статически типизированного универсального языка программирования, ориентированной на объектно-ориентированный подход) с наложением на нее обобщенной схемы разработанной программы:

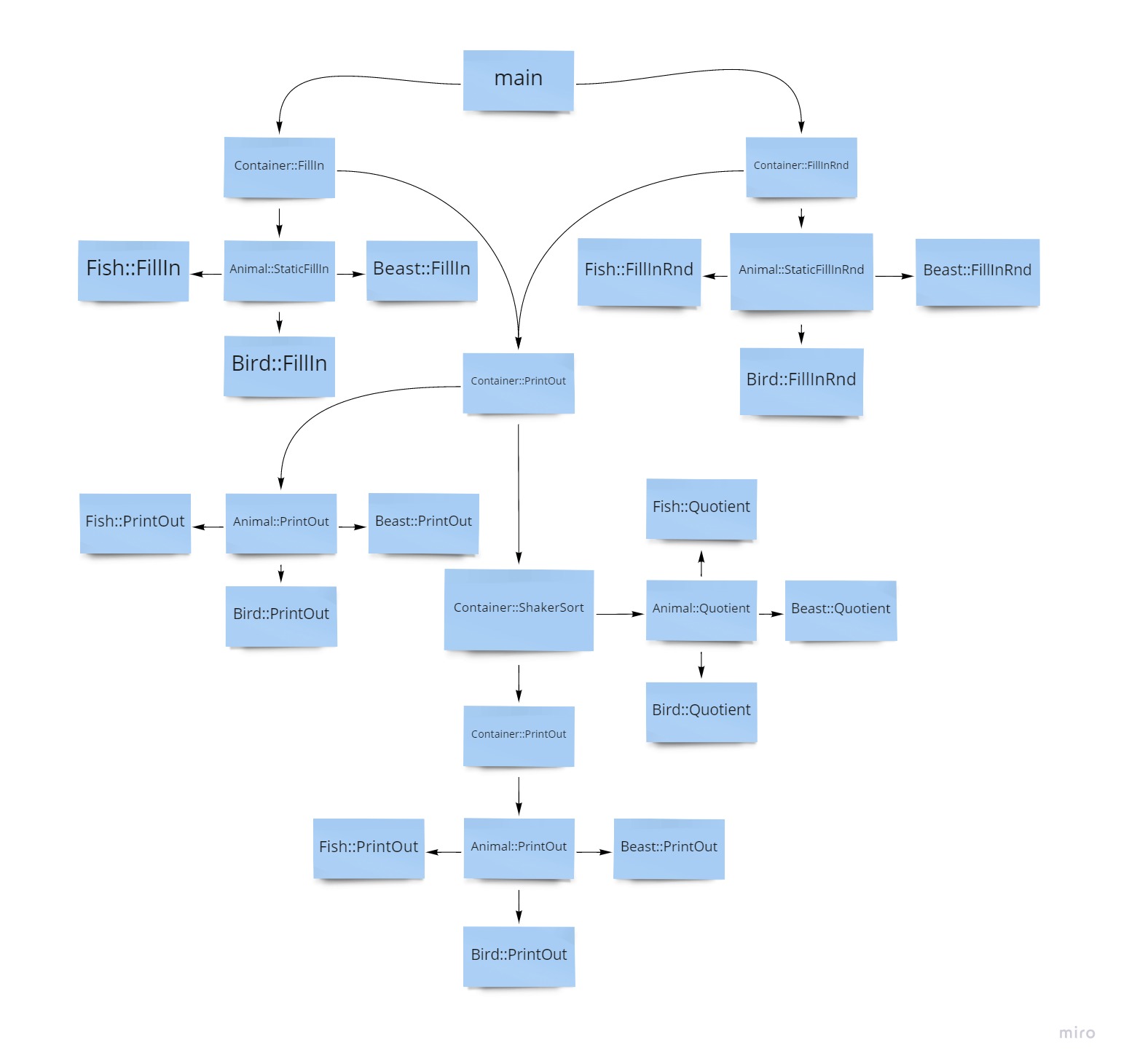
Таблица типов:

|  |  |
| --- | --- |
| Bool | 1 |
| Int | 4 |
| Char | 1 |
| Class Animal  rnd3: Random | 8  8[0] |
| Class container  len: int  array: Animal\*[len] | 80004  4[0]  10000\*8[4] = 80000[4] |
| Class Beast  enum animal\_food\_type  name: char [15]  weight: int | 23  4[0]  15\*1[4] = 15[4]  4[19] |
| Class Bird  is\_migratory: bool  name: char [15]  weight: int | 20  1[0]  15\*1[1] = 15[1]  4[16] |
| Class Fish  enum fish\_home  name: char [15]  weight: int | 23  4[0]  15\*1[4] = 15[4]  4[19] |
| Class Random  first\_number\_in\_range: int  second\_number\_in\_range: int | 8  4[0]  4[4] |

Память программы:

|  |  |
| --- | --- |
| main (int argc, char\* argv[])  argc: int  argv: char\*  isFile: bool  container: Container  input\_file: FILE\*  output\_file: FILE\* | 4[0]  1[4]  1[5]  5[6]  8[11]  8[19] |
| ShakerSort()  length: int  tmp: Animal\* | 4[0]  1[4] |
| Bird::FillInRnd()  alphabet: char [26]  rnd15: Random  name\_len: int  rnd26: Random  rnd100: Random  rnd2: Random  random\_variable: int | 26[0]  8[26]  4[34]  8[38]  8[46]  8[54]  4[62] |
| Bird::FillIn(FILE\* file)  file: FILE\*  name: char [15] | 1[0]  15[1] |
| Bird::PrintOut(FILE\* file)  file: FILE\* | 1[0] |

Стек(stack):



1. Основные характеристики программы:
2. число интерфейсных модулей (заголовочных файлов) - 6
3. модулей реализации (фалов с определением программных объектов) - 6
4. общий размер исходных текстов – 656 строк
5. Время выполнения программы для различных тестовых наборов данных:
6. Ввод из файла input, вывод в output

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Ввод из файла input\_1, вывод в output\_1

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, газета, снимок экрана

Автоматически созданное описание

1. Рандомизация 20 элементов контейнера, вывод в файл output\_2

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, газета

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Сравнительный анализ:

Характеристики данной программы:

1. число интерфейсных модулей (заголовочных файлов) - 6
2. модулей реализации (фалов с определением программных объектов) – 6
3. общий размер исходных текстов – 656 строки

Характеристики программы из задания 1:

1. число интерфейсных модулей (заголовочных файлов) - 5
2. модулей реализации (фалов с определением программных объектов) – 6
3. общий размер исходных текстов – 632 строки

Время работы данной программы, показанное на 3 вышеуказанных тестах:

Изображение выглядит как текст

Автоматически созданное описание

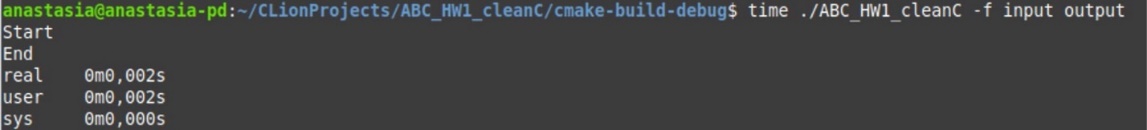
Изображение выглядит как текст

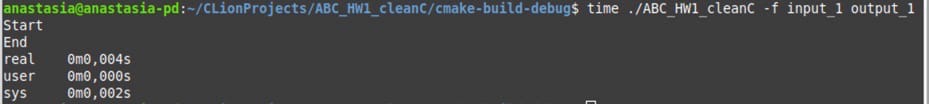
Автоматически созданное описание

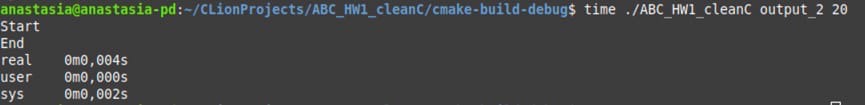
Изображение выглядит как текст

Автоматически созданное описание

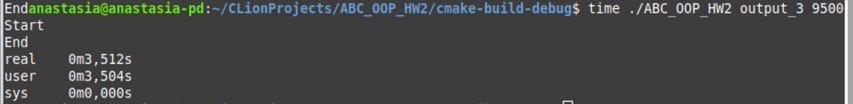
Время работы программы из задания 1 на тех же тестах:



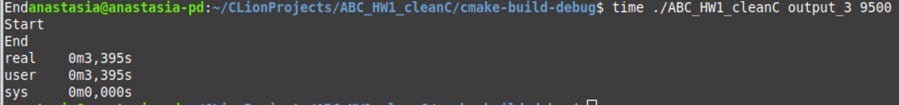




Время работы данной программы (ОО подход) на тесте со случайной генерацией 9500 элементов:



Время работы программы из задания 1 (процедурный подход) на тесте со случайной генерацией 9500 элементов:



Можно сделать вывод, что общие размеры исходного текста, а также количество модулей программ почти одинаковы. На небольших объемах данных время работы программы, написанной на статически типизированном языке и ориентированной на процедурный подход, чуть больше, чем время работы программы, также написанной на статически типизированном языке, но ориентированной на объектно-ориентированный подход, но также практически одинаково. Тест с генерацией 9500 элементов показал обратный результат по времени – ОО подход занял примерно на 0,1 секунду больше, чем процедурный.