Пояснительная записка

Описание полученного задания:

Номер варианта - 232.

1. Номер основного задания - 8.
   - Обобщенный артефакт, используемый в задании:
     Языки программирования.
   - Базовые альтернативы:
     1.1. Процедурные (наличие, отсутствие абстрактных типов данных – булевская величина)
     1.2. Объектно-ориентированные (наследование: одинарное, множественное, интерфейса – перечислимый тип)
     1.3. Функциональные языки (типизация – перечислимый тип = строгая, динамическая; поддержка «ленивых» вычислений – булевский тип)
   - Общие для всех альтернатив переменные:
     Популярность в процентах (TIOBI) — действительное
     Год создания – целое
   - Общие для всех альтернатив функции:
     Частное от деления года создания на количество символов в названии

2. Номер доп. задания (обработка данных в контейнере) - 17.
   Упорядочить элементы контейнера по убыванию используя сортировку с помощью разделения (Quick Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Структура программы:

Для выполнения задания был использован CLion - интегрированная среда разработки для языков программирования C и C++.
В коде присутствуют подробные комментарии.

Это программный продукт с использованием объектно-ориентированного подхода и статической типизации. Интерфейсный модуль language.h содержит описание класса Language, обобщающего все имеющиеся языки программирования.

От него наследуются классы Procedural(содержит описание процедурных языков), ObjectOriented(содержит описание Объектно-ориентированных языков) и Functional(содержит описание функциональных языков). Каждый из перечисленных классов записан в своём одноимённом интерфейсном модуле, и у каждого из перечисленных классов есть приватные переменные, соответствующие характеристикам языка, описанным в задании, и публичные методы, выполняющие следующие функции:

- Ввод параметров языка из файла
- Случайный ввод параметров языка
- Вывод параметров языка в форматируемый поток
- Вычисление частного языка

У каждого из вышеописанных классов есть свой одноименный модуль реализации, где и содержится код вышеописанных методов.

В интерфейсном модуле container.h описан класс Container, в котором создается приватный массив из Language и длина этого массива. В массиве контейнера хранятся ссылки на Language. Ссылка весит 4 байта, поэтому вес массива из 10000 элементов составляет 40000 байт. У Container также есть свой одноименный модуль реализации, в котором содержится код публичных методов, выполняющие следующие функции:

- Очистка контейнера от элементов (освобождение памяти)
- Ввод содержимого контейнера из указанного потока
- Случайный ввод содержимого контейнера
- Вывод содержимого контейнера в указанный поток
- Сортировка языков программирования по их частному

Присутствует интерфейсный модуль rnd.h, в котором описан класс Random, использующийся для получения случайных целых и вещественных чисел в различных диапазонах.

Таблица типов:

Тип	Память(байты)
int	4
double	8

bool	1
Random	8 (int+int)
Procedural	17 (int+bool+double+int)
ObjectOriented	28 (int+3*int+double+int)
Functional	25 (int+2*int+bool+double+int)
Language	4 (просто ссылка на один из языков)
Container	40004 (int+Language*[10000])

Размеры программы:

Число интерфейсных модулей (заголовочных файлов) = 6.
Число модулей реализации = 6 (считая main.cpp).

Размер текста:

* CMakeList.txt: 20 строк. Объем текста (Unicode UTF-8) байт - 390.
* container.h: 46 строк. Объем текста (Unicode UTF-8) байт - 1631.
* container.cpp: 97 строк. Объем текста (Unicode UTF-8) байт - 3330.
* language.h: 44 строк. Объем текста (Unicode UTF-8) байт - 1463.
* language.cpp: 58 строк. Объем текста (Unicode UTF-8) байт - 2071.
* procedural.h: 42 строк. Объем текста (Unicode UTF-8) байт - 1430.
* procedural.cpp: 46 строки. Объем текста (Unicode UTF-8) байт - 1874.
* object_oriented.h: 46 строк. Объем текста (Unicode UTF-8) байт - 1688.
* object_oriented.cpp: 74 строка. Объем текста (Unicode UTF-8) байт - 2726.
* functional.h: 47 строк. Объем текста (Unicode UTF-8) байт - 1620.
* functional.cpp: 74 строк. Объем текста (Unicode UTF-8) байт - 2652.
* main.cpp: 79 строки. Объем текста (Unicode UTF-8) байт – 2729.
* rnd.h: 39 строк. Объем текста (Unicode UTF-8) байт - 1156.

Итого:

712 строк.
Объем текста (Unicode UTF-8) байт - 24 760.
Объем текста (Unicode UTF-8) килобайт – 24,1.

Тесты:

Всего имеется 2 рукописных теста (т.е. два In.txt файла) и 3 теста с использованием генератора случайных значений. Файлов с результатами всего 10 - по два на каждый тест. В файле out<x>_1 находится содержимое контейнера сразу после ввода данных. В файле out<x>_2 находится содержимое уже отсортированного контейнера.

Время выполнения программы при разных тестах:

- Ввод из файла 9 элементов – примерно 13 миллисекунд.
- Ввод из файла 12 элементов – примерно 10 миллисекунд.
- Заполнение случайными элементами (50шт.) – примерно 4 миллисекунды.
- Заполнение случайными элементами (10000 шт.) – примерно 459 миллисекунд
- Заполнение случайными элементами (5000 шт.) – примерно 230 миллисекунд

Правила ввода рукописных тестов:

(Процедурные языки)

1

<name(строка)> <abstract_types (есть или нет (1 или 0))> <popularity(вещественное число)> <year(целое число)>

(Объектно-Ориентированные языки)

2

<name> <key_of_inheritance (наследование: одинарное-1, множественное-2, интерфейсы - 3)> <popularity(вещественное число)> <year(целое число)>

(Функциональные языки)

3

<name(строка)> <typing(типизация: строгая-1, динамическая-2)> <lazy_calculations(поддержка "ленивых" вычислений: есть-1, нет- 0)> <popularity(вещественное число)> <year(целое число)>

Сравнительный анализ с предыдущей версией программы:

Быстродействие программы не изменилось. В этом плане все значения примерно такие же.

Размер программы немножко увеличился – это минус, но некритичный.

Память, занимаемая программой в стеке, значительно уменьшилась, т.к. теперь у нас массив контейнера состоит не из структур языков (статические), а из классов языков(ссылочные), поэтому теперь память под них выделяется в «куче», а стеке хранятся только ссылки. В предыдущей программе массив контейнера весил 400000 байт, а в новой программе 40000, т.е. на порядок меньше. Это безусловно жирный плюс.

Сам вес структур(классов) конкретных языков программирования не сильно изменился, т.к. по сути все переменные остались теми же.

Итог: Новая версия программы лучше предыдущей.