ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К домашнему заданию №2 по АВС

*Сергеев Илья БПИ207*

Описание задания

Задание выполнено с целью изучения архитектуры статически типизированного объектно-ориентированного языка программирования C++.

Условие задачи

-Обобщённый артефакт, используемый в задании – различные числа.

-Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив) - 1. Комплексные (действительная и мнимая части – пара действительных чисел) 2. Простые дроби (числитель, знаменатель – пара целых чисел) 3. Полярные координаты (угол [радиан] – действительное; координаты конечной точки на плоскости)

-Общие для всех альтернатив переменные – отсутствуют

-Общие для всех альтернатив функции - Приведение каждого значения к действительному числу, эквивалентному записанному. Например, для комплексного числа осуществляется по формуле: sqrt(d^2+i^2)), а для полярных координат - расстояние.

Обработка данных контейнера

Удалить из контейнера те элементы, для которых значение, полученное с использованием функции, общей для всех альтернатив, меньше чем среднее арифметическое для всех элементов контейнера, полученное с использованием этой же функции.

Тестирование

Программа обрабатывает лишь часть множества некорректного ввода. Завершение работы с ошибкой возможно при несоответствии локали (такой тест содержится в папке tests и описывает ситуацию, при которой для ввода нецелых чисел используется запятая заместо точки).

Поступающие на вход из файла входные данные корректны тогда и только тогда, когда:

1. Файл начинается с неотрицательного целого числа типа int, описывающего кол-во поступающих на вход артефактов. Если число больше нуля, то за ним следует пробел.
2. После кол-ва артефактов в файле идут по трое через пробел сначала название последующего артефакта (complex – комплексное число, polar – число в полярной с-ме координат, fraction – дробь), а затем два числа типа double или int (первая и вторая составляющая артефакта, например, в случае с дробью, знаменатель и числитель).

После ввода кол-ва элементов создаётся, заполняется и выводится в первый выходной файл контейнер для хранения артефактов со значениями общей для всех альтернатив функции. Далее во второй выходной файл выводится список значения общей для всех альтернатив функции и контейнер **без** элементов, чьё значение функции меньше среднего.

Не обрабатываются следующие некорректные входные данные:

1. На вход помимо названий артефактов подаются не числа.
2. Значения, поступающие не приводимы к типам int (в случае с дробью) или к double (в случае с комплексным числом и числом в полярной с-ме координат)
3. Кол-во артефактов меньше первого числа в файле.
4. Для обозначения нецелых чисел используется несоответствующий настройкам локали знак (например, запятая)

Обрабатываются следующие исключения

1. Указано название несуществующего артефакта
2. Количество артефактов превышает значение первого числа в файле.

Метрики, определяющие характеристики программы для различных тестов

Число интерфейсных модулей:

Число заголовочных файлов – 6

Число модулей реализации:

Число программных файлов – 6

Общий размер исходных текстов

complex\_number.cpp – 387 байт, container.cpp – 3.33 кб, fraction\_number.cpp – 290 байт, main.cpp – 1.98 кб, number.cpp – 994 байта, polar\_number.c – 333 байта

complex\_number.h – 522 байта, container.h – 968 байта, fraction\_number.h – 536 байт, number.h – 1.12 кб, polar\_number.h – 526 байт, randGenerator.h – 844 байта

исполняемый файл (.exe) – 109 кб.

Время работы программы на различных тестовых данных

input1.txt – 2 мс, input2.txt– 1 мс, input3.txt – 2 мс, input4.txt – 2 мс, input5.txt –2 мс, тест с 12 случайно сгенрированными объектами – 3 мс. Input7.txt – 30 мс (666 артефактов)

Сравнение решением на процедурном ЯП

Решение на объектно-ориентированном языке оказалось медленнее. Тест с 666 артефактами был отработан на C за 15мс, что вдвое быстрее, чем на C++.

Однако, стоит заметить, что размер исходных кодов большинства файлов, реализующих одинаковый функционал, был меньше на C++. Это обусловлено, в частности, тем, что объектно-ориентированное программирование помогает бороться с дублированием кода и большинство функционала реализуется проще. В качестве примера рассмотрим исполняемые файлы complex\_number.c и complex\_number.cpp, реализующие функционал комплексных чисел. В первом содержится 5 реализация 5 функций, тогда как в complex\_number.cpp – лишь двух, специфичных для комплексных чисел. В решении на C++ те функции, которые являются общими для всех артефактов реализованы в number.cpp (например, ввод числа).