Пояснительная записка к четвертому домашнему заданию по АВС

Подюков Иван Владимирович, БПИ207, вариант 259

Номер задачи – 7, номер дополнительной функции – 19

Задача:

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив переменные	Общие для всех альтернатив функции
7. Фильмы	1. Игровой (режиссер – строка символов) 2. Мультфильм (способ создания – перечислимый тип = рисованный, кукольный, пластилиновый…) 3. Документальный фильм (длительност в минутах – целое)	1. Название фильма – строка символов. 2. Год выхода - целое	Частное от деления года выхода фильма на количество символов в названии (действительное число)

Дополнительная функция:

19. 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.

Общая информация о программе:

Число файлов с макроопределениями – 1

Число модулей реализации – 5

Общий размер исходных текстов – 32,9 КБ (в задании с языком С было 15,9 КБ), 1092 строчки кода.

Размер исполняемого кода – 31,1 КБ (в задании с языком С было 57,1 КБ)

Тесты	Время выполнения программы для различных тестовых наборов данных в секундах
test1.txt	0,000179624
test2.txt	0.000221375
test3.txt	0.000240484
test4.txt	0.000286064
test5.txt	0.000387164

При генерации 2000 фильмов время выполнения программы составляет 0.009523084 секунд

Сравнение характеристик при исполнении задания на языке C и на языке Assembler:

Размер исполняемого кода меньше при использовании Assembler подходе, а общий размер исходных текстов наоборот, больше. На таких же тестах программа при использовании Assembler намного быстрее (в 8 – 9 раз) работает, чем при использовании C. При генерации 2000 объектов данная программа тратит 0.009523084 секунд, а предыдущая около 0.018 секунд. Таким образом, работая на ассемблере, можно добиться наибольшей эффективности программы. Но в то же время код на ассемблере получается достаточно большим, тяжелым для писания и восприятия. В этом плане низкоуровневые языки программирования сильно уступают высокоуровневым.