Сухова Анна БПИ207

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1. Описание задания

В рамках задания необходимо:

1. Использовать:

• вычислительную систему с архитектурой x86-64;

• операционную систему Linux;

• язык ассемблера NASM

2. Разработать программу в виде консольного приложения с использованием языка ассемблера.

Запуск программы должен осуществляться из командной строки, в которой указываются: имя запускаемой программы; имя файла с исходными данными; имя файла с выходными данными.

Примеры:

command -f infile.txt outfile1.txt outfile2.txt (для ввода из файла)

command -n number_of_elements outfile1.txt outfile2.txt (для случайной генерации)

3. Реализовать в программе следующий функционал (**вариант 13**):

Обобщённый артефакт	Базовые альтернативы (и их отличительные параметры)	Общая для всех альтернатив переменная	Общая для всех альтернатив функция
Растение	1. Деревья (возраст – длинное целое) 2. Кустарники (месяц цветения – перечислимый тип) 3. Цветы (домашние, садовые, дикие –	Название – строка символов (макс. длина = 20 символов)	Частое от деления числа гласных букв в названии на общую длину названия

	перечислимый тип)		

4. Поместить данные объекты в контейнер и в соответствии с вариантом задания (**вариант 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 той же функции.

5. Провести отладку и тестирование разработанной программы на заранее подготовленных тестовых наборах данных (не менее 5). Тестовые данные с большим числом элементов должны порождаться программой с использованием генераторов случайных наборов данных. Управление вводом данных задается из командной строки.

7. Привести результаты сравнительного анализа полученных характеристик с теми, которые были получены для предыдущей программы. Сделать выводы о достоинствах и недостатках этого и предшествующего решения относительно друг друга.

2. Основные характеристики программы:

Число файлов .asm: 5
Число файлов .mac: 1
Число заголовочных файлов (библиотек): 13 (fprintf, printf, fscanf, rand, srand, fclose, fopen, strcmp, clock, atoi, time, stdout, CLOCKS_PER_SEC)
Число макроопределений: 11
Общий размер исходных текстов (программы): 39,4 Кбайт
Полученный размер исполняемого кода: 33,6 Кбайт
Время выполнения программы для тестовых наборов данных:

№ теста	Время в задании №4	Время в задании №3 (на Python)	Время в задании №2 (объектно-ориентированный подход на C++)	Время в задании №1 (процедурный подход на C++)
1	0.00671	0.0026	0.000982	0.001703
2	0.00853	0.0023	0.000843	0.00148
3	0.00671	0.0033	0.000945	0.002179
4	0.00665	0.0025	0.00132	0.01543

5	0.0688	0.5483	0.1245	0.16013
6	0.0408	0.1926	0.038996	0.081633
7	0.0128	0.0329	0.00443	0.014050
8	0.01353	0.0026	0.000752	0.001601
9	0.0636	0.5763	0.11506	0.332784

4. Сравнительный анализ с предыдущими программами

При сравнении программы на NASM с программами на Python и C++, можно выделить следующие различия:

1. По таблице времени выполнения программы для тестовых наборов данных видно, что при небольших наборах данных программа на языке ассемблера работает медленнее остальных, однако с увеличением количества данных скорость работы программы становится быстрее, чем у других программ.
2. По таблице размеров файлов с кодом разработанных программ видно, что файлы с кодом на языке NASM занимают набольшее число Кбайт.

№ программы	1	2	3	4
Общий размер файлов с кодом (Кбайт)	15,84	14,0125	12,7	39,4

3. Таблица размеров исполняемого кода программ:

№ программы	1	2	3	4
Размер исполняемого кода (Кбайт)	23	38,6	7144	33,6

4. По таблице количеств дополнительно подключаемых к разработанным программам библиотек видно, что наибольшее количество внешних модулей подключено к программе, реализованной на языке ассемблера.

№ программы	1	2	3	4
Кол-во подключаемых библиотек	4	4	6	13

5. По таблице общего количества файлов с кодом для каждой программы видно, что их количество неизменно для всех программ.

№ программы	1	2	3	4
Кол-во файлов с кодом	6	6	6	6

Таким образом, программа, разработанная на языке ассемблера NASM (с использованием библиотек языка C для ввода-вывода данных), самая неэффективная по памяти среди разработанных программ (в силу низкоуровневости языка ассемблера программы, разработанные на этом языке, требуют бóльшего количества строк кода), зато наиболее эффективна по времени с увеличением количества входных данных.