ЗАДАНИЕ № 6 ПО ПРАКТИКУМУ

для студентов 1 потока 1 курса ф-та ВМК МГУ
в 2010/2011 учебном году, весенний семестр

Тема: «Сборка многомодульных программ. Методы сортировки».
Языки программирования: Си, ассемблер NASM.

ПОСТАНОВКА ЗАДАЧИ

Реализовать два метода сортировки массива чисел и провести их экспериментальное сравнение. Тип элементов массива, конкретные методы и вид сортировки определяются вариантом задания. Для каждого из реализуемых методов необходимо предусмотреть возможность работы с массивами длины от 1 до N ($N \geq 1$). Значение N в зависимости от варианта задания либо фиксировано, либо память под массив следует выделять динамически (рекомендуется последнее).

При реализации каждого метода вычислить ***число сравнений*** элементов, ***число перемещений*** (обменов) элементов и провести ***замер времени*** работы метода в процессорных тактах.

Сравнение методов сортировки необходимо проводить на одних и тех же исходных массивах, при этом следует рассмотреть массивы разной длины. Для вариантов с фиксированным значением N рассмотреть, как минимум, n = 10, 20, 50, 100. Для вариантов с динамическим выделением памяти — n = 10, 100, 1000, 10000. Генерация исходных массивов для сортировки реализуется отдельной функцией, создающей в зависимости от заданного параметра и заданной длины конкретный массив, в котором:

- элементы уже упорядочены (1);
- элементы упорядочены в обратном порядке (2);
- расстановка элементов случайна (3, 4).

Результаты экспериментов оформить на основе нескольких запусков программы в виде следующей сводной таблицы.

Название метода сортировки

n	**Параметр**	**Номер сгенерированного массива**				**Среднее значение**
		1	2	3	4	
10	Сравнения	...	...	...	...	...
	Перемещения	...	...	...	...	...
	Время, такты	...	...	...	...	...
20	Сравнения	...	...	...	...	...
	Перемещения	...	...	...	...	...
	Время, такты	...	...	...	...	...
⋮						

ВАРИАНТЫ ЗАДАНИЯ

Замечание: варианты, помеченные вертикальной чертой слева, являются необязательными. Их выбор определяет преподаватель.

I. Данные (элементы массива).

1. Целые числа (**int**).
2. 64-разрядные целые числа (**long long int**).
3. Вещественные числа двойной точности (**double**).

II. Вид сортировки.

1. Числа упорядочиваются по неубыванию.
2. Числа упорядочиваются по невозрастанию.
3. Числа упорядочиваются по неубыванию модулей, т.е. при сравнении элементов не учитывается знак.
4. Числа упорядочиваются по невозрастанию модулей, т.е. при сравнении элементов не учитывается знак.
5. Реализуются виды сортировки 1 и 2. Конкретный вид выбирается во время препорцессирования с использованием условной компиляции и директив препроцессора (**#ifdef**, **#ifndef**, ...).

III. Методы сортировки.

1. Метод «пузырька» (см. [3] 130-132; [4] 27-28; [5] 101-102).
2. Метод простого выбора (см. [3] 169-171; [4] 15-16; [5] 99-100).
3. Метод Шелла (см. [3] 105-107; [4] 37-40; [5] 105-107).
4. Быстрая сортировка, рекурсивная реализация ([5] 114-117).
5. Пирамидальная сортировка ([2] 178-197).

Таким образом, вариант определяется следующими четырьмя параметрами: (1) переменное или фиксированное число элементов в массиве, (2) тип элементов массива, (3) вид сортировки, (4) два метода сортировки.

ТРЕБОВАНИЯ К ПРОГРАММЕ

1. Сравнение элементов массива и перемещение должны быть реализованы отдельными функциями на языке Ассемблера. Эти же функции должны обновлять глобальные счётчики сравнений и перемещений.
    1) Функция сравнения элементов принимает два числа и возвращает 1, если первое строго меньше второго, и 0, если нет. Функция увеличивает глобальный счётчик сравнений.
    2) Функция перемещения элементов принимает два указателя и меняет местами числа, по ним размещённые. Функция увеличивает глобальный счётчик сравнений.
2. Каждый из предложенных методов сортировки и генерации массива оформляется в виде отдельной функции на языке Си. Функции не имеют возвращаемых значений и принимают по два параметра: число *n* и массив *a*. Выделение динамической памяти под массив, если предусмотрено вариантом задания, делается до вызова соответствующих функций.
3. Функция *main*, а также весь вспомогательный код (включая код вывода данных и статистики), реализуется на языке Си.
4. Программа должна быть снабжена поясняющими комментариями в объёме, достаточном для её понимания.
5. Программа сдаётся в виде .zip-архива, содержащего в себе все необходимые файлы с исходным кодом, а также отчёт со сводной таблицей.
6. Сборка готовой программы должна осуществляться при помощи утилиты *make*. Должны быть определены цели *all* и *clean*, первая из которых полностью собирает программу, а вторая — удаляет все промежуточные файлы (в частности, объектные модули). В этом случае сдаваемый архив должен включать в себя `Makefile`.
7. Выбор конкретной конфигурации программы должен управляться определёнными символами на этапе препроцессирования и передаваться через ключ `-D`.

ЛИТЕРАТУРА

1. Трифонов Н. П., Пильщиков В. Н. Задания практикума на ЭВМ (1 курс). Методическая разработка (составители). — М.: ВМК МГУ, 2001.
2. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. Второе издание. — М.: «Вильямс», 2005.
3. Кнут Д. Искусство программирования для ЭВМ. Том 3. — М.: Мир, 1978.
4. Лорин Г. Сортировка и системы сортировки. — М.: Наука, 1983.
5. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989.
6. Столлман Р., МакГрат Р. GNU Make. — http://www.linuxlib.ru/prog/make_379_manual.html.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Для подсчёта количества тактов можно воспользоваться инструкцией RDTSC, например, при помощи следующей функции. Функцию следует вызвать непосредственно до выполнения сортировки и сразу после. Разность значений — количество тактов, потребовавшихся для сортировки.

```
unsigned long long
timestamp(void);
```

Листинг 1. Си-прототип функции получения значения счётчика тактов.

```
GLOBAL timestamp
timestamp:
        RDTSC
        RET
```

Листинг 2. Реализация функции получения значения счётчика тактов на Ассемблере.

2. Необходимо помнить о том, что внешние имена на платформе Windows снабжаются ведущим подчёркиванием. На этой платформе функцию `timestamp` следует в Ассемблерном коде называть `_timestamp`. Си-код при этом не меняется. К сдаче должен быть подготовлен вариант программы, предназначенный для запуска в UNIX-окружении.
3. Для генерации массива случайных элементов следует использовать функции стандартной библиотеки `rand` и `srand` следующим образом.
    1) Необходимо подключить заголовочные файлы `stdlib.h` и `time.h`.
    2) Для инициализации генератора случайных чисел сделать вызов `srand(time(NULL))`.
    3) Далее, для получения случайных чисел использовать функции `rand()`, возвращающую случайное целое число в диапазоне от 0 до `RAND_MAX`. Как правило, эта константа равна 32767.
    4) Для формирования элемента массива может понадобиться несколько вызовов `rand()`. Так, для генерации случайного 64-битного целого числа можно использовать выражение вида `rand() * rand() * rand() * rand() * rand()`, либо формировать число отдельными байтами.