Лабораторная работа № 4 «Работа со строками»

Из входного потока вводится произвольное количество строк произвольной длины. Каждая строка в общем случае содержит одно или более слов, разделенных пробелами и/или знаками табуляции. Завершение ввода определяется концом файла. Для каждой входной строки формируется новая выходная строка, в которую помещается результат. В полученной строке слова разделяются только одним пробелом, пробелов в её начале и в конце быть не должно. Введённая и сформированная строки выводятся на экран в двойных кавычках.

В ходе выполнения лабораторной работы должны быть разработаны:

1. Программа, использующая функцию `readline()` из состава библиотеки GNU readline для ввода строк и функции стандартной библиотеки для их обработки (`<string.h>`).
2. Программа, идентичная п. 1, за исключением того, что все библиотечные функции заменены на собственную реализацию данных функций, представленную в отдельных файлах (например: `mystring.h`, `mystring.c`).

Отчётность по выполнению лабораторной работы должна включать:

1. Блок-схему алгоритма работы основной программы.
2. Блок-схемы алгоритмов работы функций по обработке строк.
3. Исходные коды всех программ.
4. Тестовые наборы для программ п. 1 и п. 2.
5. Сравнительный анализ времени, потраченного на решение задачи программами п. 1 и п. 2 (на конкретных примерах).

Примечания:

1. Каждая строка представлена на физическом уровне вектором.
2. Использование массивов переменной длины (VLA — variable length arrays) не допускается.
3. Ввод строк должен быть организован с помощью функции `scanf()` со спецификациями для ввода строк. Использование функций семейства `gets()`, `getchar()`, а также спецификаций `%c` и `%m` в `scanf()` не допускается.
4. Целочисленные и строковые константы, используемые в формулировках индивидуальных заданий, должны быть заданы в исходном коде с помощью директив препроцессора `#define`.
5. Программа должна корректным образом завершаться при обнаружении `EOF` — конца файла (в UNIX-подобных ОС инициируется нажатием клавиш `Ctrl + D`, в Windows — `Ctrl + Z`).
6. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами.

7. Исходные коды программ должны быть логичным образом разбиты на несколько файлов (необходимо использовать как `*.c`-файлы, так и `*.h`-файлы).

8. Использование глобальных переменных не допускается.

9. Программы должна корректным образом работать с памятью. Для проверки необходимо использовать соответствующие программные средства, например, `valgrind` (при тестировании и отладке программ п. 1 и п. 2 необходимо запускать их командой вида `valgrind ./lab4`, а при анализе производительности — `./lab4`).