Лабораторная работа №4

по дисциплине «Программирование на Си»

Обработка строк

Кострицкий А. С., Ломовской И. В.

Москва — 2023 — `TS2304161441`

Содержание

1 Цель работы **1**
- 1.1 Задача №1 . . . 1
- 1.2 Задача №2 . . . 2
- 1.3 Задача №3 . . . 2
- 1.4 Задача №4 . . . 3
- 1.5 Примечания . . . 4

2 Взаимодействие с системой тестирования **4**

1 Цель работы

Целью лабораторной работы является знакомство студентов со строками и их обработкой.

Студент должен научиться:

1. описывать строки;
2. обрабатывать строки с помощью функций из стандартной библиотеки и без оных.

1.1 Задача №1

Решение этой задачи размещается в папке `lab_04_01`. В функции `main` следует сравнить поведение реализованных функций[1] и функций из стандартной библиотеки. От студента ожидается ровно один функциональный тест в этой задаче — на вход ничего не подаётся, файл `pos_01_in.txt` пуст, на экран следует распечатывать количество проваленных сравнений поведения реализованных функций и функций стандартной библиотеки.

Требуется самостоятельно реализовать аналоги следующих строковых функций:

0. `strpbrk`;

[1] В будущем такие тесты мы будем называть *модульными*.

1. `strspn`;
2. `strcspn`;
3. `strchr`;
4. `strrchr`.

Сигнатура аналога должна совпадать с сигнатурой оригинала с точностью до имени, к именам аналогов следует прибавлять префикс «`my_`», например, свой аналог функции `strpbrk` следует объявлять не иначе, как `my_strpbrk`.

1.2 Задача №2

Требуется написать программу, которая запрашивает у пользователя нужное по заданию количество строк, разбивает оные на слова и выполняет обработку слов. Разбиение строки на слова реализуется студентом **самостоятельно** — использовать для выделения слов библиотечные функции, например, `scanf`, `sscanf` или `strtok`, запрещено.

В результате разбора каждой строки должен быть сформирован массив слов, после чего выполняется обработка всех полученных массивов.

Результаты выводятся обязательно после фразы «`Result:␣`». Слова и числа разделяются одним пробелом.

В случае если решение задачи не может быть получено, на экран ничего не выводится, возвращается ненулевой код возврата.

Варианты

0. Ввести одну строку. Для каждого слова подсчитать количество его встреч в исходной строке. Программа должна вывести пары «слово количество-встреч». Каждая пара выводится на отдельной строке. Слова выводятся в том порядке, в котором они встретились в исходной строке.

1. Ввести одну строку. Составить массив из слов исходной строки, каждое слово должно входить в массив только один раз. Упорядочить этот массив в лексикографическом порядке. Слова из упорядоченного массива вывести на экран, разделив одним пробелом.

2. Ввести две строки. Для каждого слова из первой строки (повторяющиеся в этой же строке слова не обрабатываются) определить, входит ли оно во вторую строку. Программа должна вывести пары «слово yes|no». Каждая пара выводится на отдельной строке. Слова выводятся в том порядке, в котором они встретились в первой строке.

3. Ввести две строки. Напечатать слова, которые встречаются в двух строках только один раз, т. е. один раз либо в первой, либо во второй. Сначала выводятся слова из первой строки в том порядке, в котором они встретились в этой строке, затем — слова из второй строки.

1.3 Задача №3

Для решения этой задачи нужно использовать функции стандартной библиотеки.

Требуется написать программу, которая запрашивает у пользователя строку и разбивает оную на слова. В результате разбора строки должен быть сформирован массив слов.

Из слов, отличных от последнего, составляется новая строка. Слова в результирующую строку помещаются в обратном порядке с одним пробелом в качестве разделителя. После последнего слова пробел не добавляется.

Прежде, чем очередное слово помещается в результирующую строку, оно подвергается преобразованию в соответствии с вариантом.

Если результирующая строка не пустая, она выводится на экран следующим образом `printf("Result:␣%s\n", new_str);`

Варианты

0. Удалить из слова все последующие вхождения первой буквы.

1. Оставить в слове только первые вхождения каждой буквы.

1.4 Задача №4

Варианты

0. На вход программе подаётся произвольная строка. Строка может начинаться и заканчиваться произвольным количеством пробельных символов. Программа должна вывести только сообщение «`YES`», если в строке содержится правильный IP-адрес, и только «`NO`» — в противном случае. IP-адрес это четырёхбайтовый код, который принято записывать в виде четырёх десятичных чисел, разделённых точками. Каждое из чисел может принимать значения от 0 до 255. Могут быть напечатаны лидирующие нули. Примеры валидных IP-адресов:

   - 127.0.0.0
   - 192.168.0.1
   - 192.168.43.1
   - 255.0.255.255

1. Вещественное число в экспоненциальной форме описывается следующим регулярным выражением:

   `[+-]?(\d+([.]\d*)?([eE][+-]?\d+)?|[.]\d+([eE][+-]?\d+)?)`

   На вход программе подаётся произвольная строка. Строка может начинаться и заканчиваться произвольным количеством пробельных символов. Программа должна вывести только сообщение «`YES`», если в строке содержится правильное вещественное число, и только «`NO`» — в противном случае.

2. Для представления даты выбрано следующее представление: «`[D]D␣month␣YYYY`»

   Месяц это одна из строк «`january`», «`february`», «`march`», «`april`», «`may`», «`june`», «`july`», «`august`», «`september`», «`october`», «`november`», «`december`». Составные части даты могут разделяться произвольным количеством пробелов. Написание месяца не чувствительно к регистру символов.

   На вход программе подаётся произвольная строка. Строка может начинаться и заканчиваться произвольным количеством пробельных символов. Программа должна вывести только сообщение «`YES`», если в строке содержится правильная дата в указанном формате, и только «`NO`» — в противном случае. Обратите внимание, что проверить нужно не только валидность написания даты, но и валидность самой даты (например, дата «40 april 2010» правильная по написанию, но не по значению). Не забудьте про правила для високосных годов.

3. Для записи телефонного номера используется следующее представление

   [+код_страны](код_оператора)-DDD-DD-DD,

   где код_страны состоит из произвольного количества десятичных цифр; код_оператора состоит из трех десятичных цифр; буква D означает любую из десятичных цифр.

   На вход программе подается произвольная строка. Строка может начинаться и заканчиваться произвольным количеством пробельных символов. Программа должна вывести только сообщение «`YES`», если в строке содержится правильный телефонный номер в указанном формате, и только «`NO`» — в противном случае.

1.5 Примечания

1. *Длина строки* не превышает 256 символов — вызов функции `strlen` ни от одной строки в задании не вернёт число, большее 256. Длина *слова* не превышает 16 символов.

2. Слова разделяются одним или несколькими пробелами и знаками пунктуации: «`,;:-.!?`».

3. *Статические массивы* следует отличать от *массивов переменной длины (англ. Variable Length Array, VLA)*. Во избежание случайного использования последних при компиляции программы необходимо указывать ключ `-Wvla`.

4. Для реализации каждой из задач этой лабораторной работы необходимо выделить несколько осмысленных функций. Необходимо предусмотреть обработку ошибочных ситуаций.

5. При вводе строки следует контролировать переполнение буфера.

6. Помните, что лучше пользоваться функциями и макросами из модуля `ctypes`, чем делать предположения о расположении символов в таблице кодировки самостоятельно.

2 Взаимодействие с системой тестирования

1. Исходный код лабораторной работы размещается студентом в ветви `lab_LL`, а решение каждой из задач — в отдельной папке с названием вида `lab_LL_PP_CC`, где LL — номер лабораторной, PP — номер задачи, CC — вариант студента. Если дана общая задача без вариантов, решение следует сохранять в папке с названием вида `lab_LL_PP`.

   Пример: решения восьми задач седьмого варианта пятой лабораторной размещаются в папках `lab_05_01_07`, `lab_05_02_07`, `lab_05_03_07`, . . . , `lab_05_08_07`.

2. Исходный код должен соответствовать оглашённым в начале семестра правилам оформления.

3. Если для решения задачи студентом создаётся отдельный проект в IDE, разрешается поместить под версионный контроль файлы проекта, добавив перед этим необходимые маски в список игнорирования. Старайтесь добавлять маски общего вида. Для каждого проекта должны быть созданы, как минимум, два варианта сборки: `Debug` — с отладочной информацией, и `Release` — без отладочной информации.

4. Сборка проекта на сервере происходит с помощью компилятора gcc с ключами `std=c99`, `Wall`, `Werror`, `Wpedantic`, `Wextra`.

   При сборке проектов, в которых используются типы с плавающей точкой, дополнительно указываются флаги `Wfloat-equal` и `Wfloat-conversion`.

   При сборке проектов лабораторных работ, в которых запрещено использовать массивы переменной длины (VLA), дополнительно указывается флаг `Wvla`.

   Если в Вашей программе используются математические функции из стандартной библиотеки, в Linux команда компиляции Вашей программы должна включать ключ `lm`, указывающий компилятору на явную компоновку математической библиотеки, которая в Linux не добавляется по умолчанию, в отличие от оставшейся части стандартной библиотеки.

   Пример:

   ```
   gcc -std=c99 -Wall -Werror -o app.exe main.c -lm
   ```

5. Крайне рекомендуется для проверки с некоторой периодичностью дополнительно собирать проект с помощью компилятора clang с тем же набором флагов.

6. Крайне рекомендуется проводить анализ проекта с помощью одного или нескольких статических анализаторов, которые рассматриваются в рамках практикума. Помните, что рекомендации статанализатора нужно принимать или отвергать обоснованно.

7. Для каждой программы ещё до реализации студентом подготавливаются и помещаются под версионный контроль в подпапку `func_tests/data/` функциональные тесты, демонстрирующие её работоспособность.

Позитивные входные данные следует располагать в файлах вида `pos_TT_in.txt`, выходные — в файлах вида `pos_TT_out.txt`, аргументы командной строки при наличии — в файлах вида `pos_TT_args.txt`, где TT — номер тестового случая.

Негативные входные данные следует располагать в файлах вида `neg_TT_in.txt`, выходные — в файлах вида `neg_TT_out.txt`, аргументы командной строки при наличии — в файлах вида `neg_TT_args.txt`, где TT — номер тестового случая.

Разрешается помещать под версионный контроль в подпапку `func_tests/scripts/` сценарии автоматического прогона функциональных тестов. Если Вы используете при автоматическом прогоне функциональных тестов сравнение строк, не забудьте проверить используемые кодировки. Помните, что UTF-8 и UTF-8(BOM) — две разные кодировки.

Под версионный контроль в подпапку `func_tests/` также помещается файл `readme.md` с описанием в свободной форме содержимого каждого из тестов. Вёрстка файла на языке Markdown обязательной не является, достаточно обычного текста.

Пример: восемь позитивных и шесть негативных функциональных тестов без дополнительных ключей командной строки должны размещаться в файлах `pos_01_in.txt`, `pos_01_out.txt`, ..., `neg_06_out.txt`. В файле `readme.md` при этом может содержаться следующая информация:

```
# Тесты для лабораторной работы №LL

## Входные данные
Целые a, b, c

## Выходные данные
Целые d, e

## Позитивные тесты:
- 01 - обычный тест;
- 02 - в качестве первого числа нуль;
...
- 08 - все три числа равны.

## Негативные тесты:
- 01 - вместо первого числа идёт буква;
- 02 - вместо второго числа идёт буква;
...
- 06 - вводятся слишком большие числа.
```

8. Если не указано обратное, успешность ввода должна контролироваться. При первом неверном вводе программа должна прекращать работу с ненулевым кодом возврата.

9. Вывод программы может содержать текстовые сообщения и числа. Если не указано обратное, тестовая система анализирует числа в потоке вывода, поэтому

они могут быть использованы только для вывода результатов — использовать числа в информационных сообщениях запрещено.

Пример: сообщение «Input point 1:» будет неверно воспринято тестовой системой, а сообщения «Input point A:» или «Input first point:» — правильно.

Тестовая система вычленяет из потока вывода числа, обособленные пробельными символами.

Пример: сообщения «a=1.043» и «a = 1.043.» будут неверно восприняты тестовой системой, а сообщения «a: 1.043» или «a = 1.043» — правильно.

10. Если не указано обратное, числа двойной точности следует выводить, округляя до шестого знака после точки.