Задание STR
Обработка строк

В следующих задачах требуется написать указанную в варианте функцию для обработки строк, **отвечающую заданному прототипу**. Требуется использовать эту функцию в тестовой программе, которая должна в цикле читать строки с клавиатуры и выдавать ответ на экран (желательно точно под исходной строкой) до тех пор, **пока пользователь не введет пустую строку** (строку нулевой длины). В этом задании можно считать, что все вводимые строки ограниченной длины (< 100) и использовать буфер (массив) фиксированного размера и функцию чтения строк `gets(s)`.

Буквами договоримся считать прописные и строчные латинские буквы A–Z и a–z, словами — последовательность букв и цифр 0–9. Все остальные символы в строке будем считать **разделителями** слов.

В учебных целях при решении задач данного раздела функций стандартной библиотеки, упрощающих работу со строками и символами (напр. из `<string.h>`), следует **избегать**, реализуя необходимую функциональность самостоятельно.

Примеры диалога программы и пользователя:

```
Задание STR-1: Разворот строк
Введите строку: Hello, world!
Результат     : !dlrow ,olleH
Введите строку: int main(void);
Результат     : ;)diov(niam tni
Введите строку: _
```

Комментарии к задаче

Вспомогательные функции В ходе решения задач необходимо выделить логические блоки, которые решают некоторую независимую подзадачу. Такие блоки необходимо выносить в отдельные функции. Это поможет как пониманию того, что делает ваша программа, так и может сэкономить место, если данная функция будет использоваться не единожды. Примерами таких функций могут служить, как вариант, аналоги функций из библиотеки `<string.h>`, например, проверка того является ли символ буквой (в нашей нотации), в таком случае создается функция `int IsAlNum(char c)`, функции поиска или обработки одного слова, куда могут передаваться строчка и индексы начала и конца слов. Разбиение кода на функции по-

могает его лучшему логическому структурированию, повышает уровень программиста и облегчает процесс поиска ошибок.

Параметризация В тех местах, где данные фиксированные, все равно работайте с ними так, как будто не знаете, что там, например, в вариантах с поиском или удалением слов, искомое слово (скажем, main) должно так же быть параметром некоторой вспомогательной функции.

План решения

1. Сначала необходимо разобраться с алгоритмом решения задачи, понять, какие вспомогательные функции будут для этого нужны.

2. Далее, рекомендуется реализовать код требуемых функций. Реализовывать можно поэтапно, разбивая на меньшие логические блоки. Для ускорения процесса тестирования, сначала можно тестировать функции на строках, заданных статически прямо в коде. Это позволит сэкономить время на ввод строк при каждом запуске.

3. После того, как основная и вспомогательные функции реализованы и протестированы, можно реализовать механизм считывания строки в с клавиатуры в цикле, завершая таким образом решение задачи.

Варианты

Вариант `STR-1` (Разворот строк). В рамках общего условия задачи написать функцию, которая в заданном буфере разворачивает строку задом наперед. Например, из «`The good and the EVIL ones.`» должно получиться «`.seno LIVE eht dna doog ehT`».

Прототип: `void Reverse(char str[]);`

Вариант `STR-2` (Разворот слов). В рамках общего условия задачи написать функцию, которая в заданном буфере разворачивает каждое слово строки задом наперед. Разделители при этом не меняются. Например, из «`The good and the EVIL ones.`» должно получиться «`ehT doog dna eht LIVE seno.`».

Прототип: `void ReverseWords(char str[]);`

Вариант STR-3 (Верхний регистр). В рамках общего условия задачи написать функцию, которая в заданном буфере все буквы заменяет на заглавные. Например, из «`The good and the EVIL ones.`» должно получиться «`THE GOOD AND THE EVIL ONES.`».

Прототип: `void UpperCase(char str[]);`

Вариант STR-4 (Нижний регистр). В рамках общего условия задачи написать функцию, которая в заданном буфере все буквы заменяет на строчные. Например, из «`The good and the EVIL ones.`» должно получиться «`the good and the evil ones.`».

Прототип: `void LowerCase(char str[]);`

Вариант STR-5 (Смена регистра). В рамках общего условия задачи написать функцию, которая в заданном буфере все заглавные буквы заменяет на строчные, а строчные — на заглавные. Например, из «`The good and the EVIL ones.`» должно получиться «`tHE GOOD AND THE evil ONES.`».

Прототип: `void SwapCase(char str[]);`

Вариант STR-6 (Упрощенный заголовок). В рамках общего условия задачи написать функцию, которая в заданном буфере все первые буквы слов заменяет на заглавные, а остальные — на строчные. Например, из «`The good and the EVIL ones.`» должно получиться «`The Good And The Evil Ones.`».

Прототип: `void TitleCase(char str[]);`

Вариант STR-7 (Заголовок). В рамках общего условия задачи написать функцию, которая в заданном буфере все первые буквы длинных слов (> 3) заменяет на заглавные, а остальные — на строчные. Короткое слово должно начинаться с заглавной только если это первое слово строки. Например, из «`The good and the EVIL ones.`» должно получиться «`The Good and the Evil Ones.`».

Прототип: `void TitleCase(char str[]);`

Вариант STR-8 (Удаление разделителей). В рамках общего условия задачи написать функцию, которая в заданном буфере удаляет разделители, прижимая слова друг к другу. Например, из «`The good and the EVIL ones.`» должно получиться «`ThegoodandtheEVILones`».

Прототип: `void RemoveSeparators(char str[]);`

Вариант STR-9 (Удаление подстрок). В рамках общего условия задачи написать функцию, которая в заданном буфере находит и удаляет все подстроки «`evil`» (не обязательно слова целиком). Необходимо обойтись без дополнительного буфера. Например, из «`The evil and the superevil ones.`» должно получиться «`The and the super ones.`».

Прототип: `void RemoveEvil(char str[]);`

Вариант STR-10 (Удаление слов). В рамках общего условия задачи написать функцию, которая в заданном буфере находит и удаляет все слова «`evil`» (только слова целиком). Необходимо обойтись без дополнительного буфера. Например, из «`The evil and the superevil ones.`» должно получиться «`The and the superevil ones.`».

Прототип: `void RemoveEvilWord(char str[]);`

Вариант STR-11 (Замена слов). В рамках общего условия задачи написать функцию, которая в заданном буфере находит и заменяет все слова «`wicked`» на «`good`» (только слова целиком). Необходимо обойтись без дополнительного буфера. Например, из «`The wicked and the EVIL ones.`» должно получиться «`The good and the EVIL ones.`».

Прототип: `void ReplaceWickedWord(char str[]);`

Вариант STR-12 (Длинные слова). В рамках общего условия задачи написать функцию, которая в заданном буфере находит первое из слов максимальной длины и оставляет в буфере только его, сдвигая к началу. Например, из «`The good and the EVIL ones.`» должно получиться «`good`».

Прототип: `void FindLongWords(char str[]);`

Вариант STR-13 (Забой длинных слов). В рамках общего условия задачи написать функцию, которая в заданном буфере находит все слова максимальной длины и забивает их звездочками. Например, из «`The good and the EVIL ones.`» должно получиться «`The **** and the **** ****.`».

Прототип: `void HideLongWords(char str[]);`

Вариант STR-14 (Забой слов-палиндромов). В рамках общего условия задачи написать функцию, которая в заданном буфере находит все слова, являющиеся палиндромами (игнорируя регистр букв), и забивает их звездочками. Например, из «`anna AmoreRoma`» должно получиться «`**** *********`».

Прототип: `void HidePalindromes(char str[]);`

Вариант STR-15 (Подмена слов). В рамках общего условия задачи написать функцию, которая в заданном буфере находит все слова максимальной длины и заменяет их все на первое из найденных. Необходимо обойтись без дополнительного буфера. Например, из «`The good and the EVIL ones.`» должно получиться «`The good and the good good.`»..

Прототип: `void SubstituteLongWords(char str[]);`

Вариант STR-16 (Обмен слов). В рамках общего условия задачи написать функцию, которая в заданном буфере меняет местами соседние пары слов, сдвигая разделители при необходимости, то есть из «`a+=bc*d`» должно получиться «`bc+=a*d`». Необходимо обойтись без дополнительного буфера.

Прототип: `void SwapWords(char str[]);`

Вариант STR-17 (Циклическая перестановка слов). В рамках общего условия задачи написать функцию, которая в заданном буфере циклически переставляет слова влево, сдвигая разделители при необходимости, то есть из «`a+=bc*d`» должно получиться «`bc+=d*a`». Необходимо обойтись без дополнительного буфера.

Прототип: `void RotateWords(char str[]);`

Вариант STR-18 (Удаление комментариев). В рамках общего условия задачи написать функцию, которая в заданном буфере удаляет все между соответствующими парами `/*` и `*/`. Вложенные комментарии поддерживать не надо, то есть от «`a/*b/*c*/d*/e`» останется «`ad*/e`». Необходимо обойтись без дополнительного буфера.

Прототип: `void RemoveComments(char str[]);`

Вариант STR-19 (Удаление вложенных комментариев). В рамках общего условия задачи написать функцию, которая в заданном буфере удаляет все между соответствующими парами `/*` и `*/`. Надо поддержать вложенные комментарии, то есть от «`a/*b/*c*/d*/e`» останется «`ae`». Необходимо обойтись без дополнительного буфера.

Прототип: `void RemoveNestedComments(char str[]);`