Нуль-терминированные строки

Вам необходимо написать два консольных приложения, соответствующих задачам A и B. В задаче A головная программа должна демонстрировать возможности работы с разработанной Вами функцией для работы со строками. Использовать функции из файла <string.h> нельзя, за исключением сравнения работы вашей и стандартной функции.

В задаче B предполагается, что длина входной строки не превосходит 300 символов, она вводится с консоли. Вам необходимо вывести на консоль либо преобразованную строку, либо требуемые в условии данные. Необходимо использовать только тип char *, работать с классом string нельзя!

Вариант 1

A. Написать собственную реализацию стандартной функции strcat.

Функция strcat описана в заголовочном файле <string.h> как

```
char *strcat (char *strDestination, const char *strSource);
```

и предназначена для добавления символов из strSource к концу строки strDestination (сцепление строк strDestination и strSource). Никаких проверок на переполнение выделенной памяти не производится. Функция возвращает адрес strDestination.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, в котором число различных символов минимально. Если таких слов несколько, найти первое из них.

Вариант 2

A. Написать собственную реализацию стандартной функции strcmp.

Функция strcmp описана в заголовочном файле <string.h> как

```
int strcmp (char *str1, const char *str2);
```

и предназначена для сравнения строк str1 и str2.

Возвращаемое значение	Отношение между строками
-1	str1 лексикографически меньше, чем str2
0	str1 эквивалентна str2
1	str1 лексикографически больше, чем str2

B. Строка состоит из слов, разделенных ровно одним пробелом, пробелов перед первым и после последнего слова нет. Поместить в начало строки слова, содержащие только цифры, а затем – все остальные слова. Порядок слов внутри заданных групп не должен изменяться.

Вариант 3

A. Написать собственную реализацию стандартной функции strstr.

Функция strstr описана в заголовочном файле <string.h> как

```
char *strstr (const char *string, const char *strCharSet);
```

и предназначена для поиска строки strCharSet в строке string. Возвращается указатель на начальный символ первого вхождения strCharSet в строку string или NULL, если string не содержит strCharSet.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти количество слов, содержащих только символы латинского алфавита, а среди них – количество слов с равным числом гласных и согласных букв.

Вариант 4

A. Написать собственную реализацию стандартной функции strcpy.

Функция strcpy описана в заголовочном файле <string.h> как

```
char *strcpy (char *strDestination, const char *strSource);
```

и предназначена для копирования строки strSource в строку strDestination. Никаких проверок на переполнение выделенной памяти не производится. Функция возвращает адрес Destination.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, символы в котором идут в строгом возрастании их кодов. Если таких слов несколько, найти первое из них.

Вариант 5

A. Написать собственную реализацию стандартной функции itoa.

Функция itoa описана в заголовочном файле <stdlib.h> как

```
char *itoa (int value, char *string, int radix);
```

и предназначена для преобразования целого числа value в его символьное представление string в системе счисления с основанием radix. Значение radix лежит в интервале от 2 до 36, старшая цифра в 36-ричной системе счисления – 'Z'. Функция возвращает адрес string.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, в котором число различных символов максимально. Если таких слов несколько, найти последнее из них.

Вариант 6

A. Написать собственную реализацию стандартной функции atoi.

Функция atoi описана в заголовочном файле <stdlib.h> как

```
int atoi (const char *string);
```

и предназначена для преобразования строки string, содержащей символьное представление целого числа в десятичной системе счисления, к этому числу.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, один и тот же символ в котором встречается максимальное число раз (в рамках всей строки). Если таких слов несколько, найти первое из них.

Вариант 7

A. Написать собственную реализацию стандартной функции strchr.

Функция strchr описана в заголовочном файле <string.h> как

```
char *strchr (const char *string, int c);
```

и предназначена для поиска в нуль-терминированной строке string символа c (признак конца строки также участвует в поиске). Возвращается указатель на начальный символ первого вхождения c в строку string или NULL, если string не содержит этого символа.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти количество слов, представляющих собой правильные записи дат в формате "dd/mm/yyyy".

Вариант 8

A. Написать собственную реализацию стандартной функции strrchr.

Функция strrchr описана в заголовочном файле <string.h> как

```
char *strrchr (const char *string, int c);
```

и предназначена для поиска в нуль-терминированной строке string символа c (признак конца строки также участвует в поиске). Возвращается указатель на начальный символ последнего вхождения c в строку string или NULL, если string не содержит этого символа.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, состоящее только из различных символов. Если таких слов несколько, найти первое из них.

Вариант 9

A. Написать собственную реализацию стандартной функции strncat.

Функция strncat описана в заголовочном файле <string.h> как

```
char *strncat (char *strDest, const char *strSource, size_t count);
```

и предназначена для добавления не более чем count символов из strSource к концу строки strDest. Никаких проверок на переполнение выделенной памяти не производится. Функция возвращает адрес strDest.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Среди слов, состоящих только из цифр, найти слово-палиндром. Если таких слов больше одного, найти второе из них.

Вариант 10

A. Написать собственную реализацию стандартной функции strspn.

Функция strspn описана в заголовочном файле <string.h> как

```
int strspn (const char *string, const char *strCharSet);
```

и возвращает длину начальной подстроки из string, содержащей только символы из множества strCharSet.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Среди слов, состоящих только из цифр, найти слово, содержащее максимальное число нулей. Если таких слов больше одного, найти предпоследнее из них.

Вариант 11

A. Написать собственную реализацию стандартной функции strcspn.

Функция strcspn описана в заголовочном файле <string.h> как

```
int strcspn (const char *string, const char *strCharSet);
```

и возвращает длину начальной подстроки из string, не содержащей символов из множества strCharSet.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Если первый символ слова является строчной латинской буквой, заменить этот символ на соответствующую прописную букву.

Вариант 12

A. Написать реализацию функции rtrim.

Ее прототип выглядит следующим образом:

```
char *rtrim (const char *string);
```

Функция возвращает указатель на новую строку, полученную из string удалением начальных пробелов. Память под эту строку должна выделяться внутри функции rtrim путем вызова new.

B. Строка состоит из слов, разделенных ровно одним пробелом, пробелов перед первым и после последнего слова нет. Поместить в начало строки слова, содержащие только символы латинского алфавита, а затем – все остальные слова, Порядок слов внутри заданных групп не должен изменяться.

Вариант 13

A. Написать реализацию функции ltrim.

Ее прототип выглядит следующим образом:

```
char *ltrim (const char *string);
```

Функция возвращает указатель на новую строку, полученную из string удалением конечных пробелов. Память под эту строку должна выделяться внутри функции ltrim путем вызова new.

B. В исходной строке выделить максимальную по длине подстроку-палиндром. Пробелы влияют на свойство палиндрома. Если таких подстрок несколько, найти первую из них.

Вариант 14

A. Написать реализацию функции padr.

Ее прототип выглядит следующим образом:

```
char *padr (const char *string, int len, int c=' ');
```

Функция возвращает указатель на новую строку длины len, полученную из string либо удалением лишних символов справа, либо добавлением справа нужного числа символов c. Память под эту строку должна выделяться внутри функции padr путем вызова new.

B. Строка состоит из слов, разделенных ровно одним пробелом, пробелов перед первым и после последнего слова нет. Найти первое из самых длинных слов, последнее из самых коротких слов, поменять найденные слова местами.

Вариант 15

A. Написать реализацию функции padl.

Ее прототип выглядит следующим образом:

```
char *padl (const char *string, int len, int c=' ');
```

Функция возвращает указатель на новую строку длины len, полученную из string либо удалением лишних символов слева, либо добавлением слева нужного числа символов c. Память под эту строку должна выделяться внутри функции padl путем вызова new.

B. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, символы в котором идут в строгом убывании их кодов. Если таких слов несколько, найти последнее из них.

Вариант 16

A. Написать реализацию функции repeat.

Ее прототип выглядит следующим образом:

```
char *repeat (const char *string, unsigned int count);
```

Функция возвращает указатель на новую строку, полученную несколькими последовательными повторениями строки string. Количество повторений задается вторым параметром. Память под эту строку должна выделяться внутри функции repeat путем вызова new.

В. Строка состоит из слов, разделенных одним или несколькими пробелами. Найти слово, символы в котором идут в строгом возрастании их кодов. Если таких слов несколько, найти первое из них.