**Лабораторная работа №15**

**«Строковый тип данных»**

**Теория**

Если требуется ввести в массив весь текст, то следует использовать функцию char \*gets(char \*str); str – это указатель на массив символов, в который записываются вводимые пользователем символы. В качестве результата функция также возвращает str.

Функция puts отображает на экране символы строки своего аргумента - массива, после чего переводит курсор на новую строку в окне программы: int puts(const char \*str); str – указатель на массив с выводимой строкой. В качестве результата функция возвращает целое значение – количество выведенных символов, а в случае ошибки возвращается значение константы EOF.

Введённые по gets строки с русскими буквами выводятся puts правильно, но строковые констаны - не правильно, а для латинских букв такого несоответствия нет.

В языке C операции над строками не определены. Единственным исключением является инициализация массива при его определении значением строковой константы.

Далее даны сведения о некоторых библиотечных функциях работы со строками, объявленных в заголовочном файле string.h. В описаниях их использования приведены прототипы функций, в которых тип size\_t представляет беззнаковое целое, имеющее тот же тип, что и результат оператора sizeof.

Функция void \*memchr(const void \*buf, int ch, size\_t count) находит первое вхождение символа ch в первых count символах массива buf. Функция возвращает указатель на первый из символов ch в массиве buf или нулевой указатель, если символ ch не найден.

Функция int memcmp(const void \*buf1, const void \*buf2, size\_t count) сравнивает в лексикографическом порядке первые count символов двух строк buf1 и buf2. Отрицательный результат означает, что первая строка меньше второй, нулевой – строки равны, положительный – первая строка больше второй.

Функция void \*memcpy(void \*buf1, const void \*buf2, size\_t count) копирует count символов из массива buf2 в массив buf1. Если заданные массивы перекрываются, то поведение функции не определено. Функция возвращает значение указателя buf1.

Функция void \*memmove(void \*masto, const void \*masfrom, size\_t count) копирует count символов из массива masfrom в массив masto. Если заданные массивы перекпываются, то процесс копирования проходит корректно. В этом случае содержимое массива masfrom будет помещено в массив masto, но содержимое массива masfrom изменится. Функция возвращает значение указателя masto.

Функция void \*memset(void \*buf, int ch, size\_t count) копирует параметр ch в первые count символов массива, адресуемого параметром buf. Функция возвращает значение указателя buf. Функция используется в основном для инициализации массива известным значением.

Функция char \*strcat(char \*str1, const char \*str2) присоединяет к строке str1 копию строки str2 и завершает строку str1 нулевым символом. Конечный нуль-символ, первоначально завершающий строку str1, перезаписывается первым символом строки str2. Строка str2 при этом не изменяется. Если заданные массивы перекрываются, то поведение функции не определено. Функция возвращает значение указателя str1. При выполнении операций с массивами символов контроль нарушения их границ не выполняется, поэтому программист должен позаботиться о достаточном размере массива str1, позволяющем вместить как его исходное содержимое, так и содержимое массива str2. В противном случае не хватит места для записи нуль-символа, что приведет к неправильной обработке массива.

Функция char \*strchr(const char \*str, int ch) возвращает указатель на первое вхождение параметра ch в строку str. Если указанный символ не будет найден, то возвращается нулевой указатель.

Функция int strcmp(const char \*str1, const char \*str2) сравнивает в лексикографическом порядке две заданные строки и возвращает целое значение, определяющее результат сравнения. Отрицательное значение означает, что первая строка меньше второй, нулевое – строки равны, положительное – первая строка больше второй.

Функция int strcoll(const char \*str1, const char \*str2) сравнивает строку, адресуемую указателем str1, со строкой, адресуемой указателем str2. Сравнение выполняется с учетом значения параметра locale, заданного с помощью функции setlocale. Функция возвращает целое значение, определяющее результат сравнения. Отрицательное значение означает, что первая строка меньше второй, нулевое – строки равны, положительное – первая строка больше второй.

Функция char \*strcpy(char \*str1, const char \*str2) копирует содержимое строки str2 в строку str1. Строка str2 должна содержать завершающий нуль-символ. Функция возвращает значение указателя str1.

Функция size\_t strcspn(const char \*str1, const char \*str2) возвращает длину начальной подстроки в строке, адресуемой параметром str1, которая не содержит ни одного символа из строки, адресуемой параметром str2. Иначе говоря, функция возвращает индекс первого символа в строке str1, который совпадает с любым символом в строке str2.

Функция char \*strerror(int errnum) возвращает указатель на строку, содержащую ситемное сообщение об оштбке, связанной со значением errnum.

Функция size\_t strlen(const char \*str) возвращает длину строки, адресуемой параметром str, причем строка должна заканчиваться символом конца строки. При этом символ конца стьроки ‘\0’ не учитывается.

Функция char \*strncat(char \*str1, const char \*str2, size\_t count) присоединяет к строке, адресуемой параметром str1, не более count символов строки, адресуемой параметром str2, завершая строку str1 нулевым символом. Конечный нуль-символ, первоначально завершающий строку str1, перезаписывается первым символом строки str2. Строка str2 в результате конкатенации не изменяется. Если строки перекрываются, то поведение функции не определено. При использовании эьлй функции программист сам должен позаботиться о том, чтобы строка str1 имела достаточную длину, позволяющую вместить исходную строку, так и присоединяемую. В противном случае нуль-символ не записывается, что приводит к ошибкам обработки строки.

Функция int strnccmp(char \*str1, const char \*str2, size\_t count) сравнивает в лексикографическом порядке не более count символов двух заданных строк и возвращает целое значение, определяющее результат сравнения. Отрицательное значение означает, что первая строка меньше второй, нулевое – строки равны, положительное – первая строка больше второй. Строки должны заканчиваться нуль-символами. Если одна из строк содержит менее count символов, то сравнение заканчивается при обнаружении первого нуль-символа.

Функция char \*strncpy(char \*str1, const char \*str2, size\_t count) копирует не более count символов из строки, адресуемой параметром str2, в строку, адресуемую параметром str1. Строка str2 должна содержать завершающий нуль-символ. Функция возвращает значение указателя str1. Если заданные массивы символов перекрываются, то поведение функции не определено. Если длина строки, адресуемой параметром str2, меньше значения count, то в конец строки-результата str1 добавляются недостаюшие нулевые символы. Если длина строки, адресуемой параметром str2, больше значения count, то строка-результат не будет заканчиваться символом конца строки. Функция возвращает значение указателя str1.

Функция char \*strpbrk(const char \*str1, const char \*str2) возвращает указатель на первый символ в строке, адресуемый параметром str1, который совпадает с любым символомв строке, адресуемой параметром str2. Символы конца строки, которыми должны заканчиваться строки, в расчет не принимаются. Если слвпадений нет, то возвращается нулевой указатель.

Функция char \*strrchr(const char \*str, int ch) возвращает указатель на последнее вхождение символа ch в строку, адресуемую параметром str. Если совпадение не обнаружено, то возвращается нулевой указатель.

Функция size\_t strspn(const char \*str1, const char \*str2) возвращает длину начальной подстроки, адресуемой параметром str1, которая состоит только из символов, содержащихся в строке, адресуемой параметром str2. Иначе говоря, функция возвращает индекс первого символа в строке str1, который не совпадает ни с одним из символов строки str2.

Функция char \*strstr(const char \*str1, const char \*str2) возвращает указатель на первое вхождение подстроки, адресуемой параметром str2, в строку, адресуемую параметром str1. Если совпадение не обнаружено, то возвращается нулевой указатель.

Функция char \*strtok(char \*str1, const char \*str2) возвращает указатель на следующую лексему (слово) в строке, адресуемой параметром str1, и на место разделителя за последним символом лексемы записывает нуль - символ. Символы, образующие строку, адресуемую параметром str2, представляют собой разделители, которые определяют лексему. При отсутствии лексемы, подлежащей возврату, возвращается нулевой указатель. Для выделения лексем из строки при первом обращении к функции параметр str1 должен указывать на начало этой строки. При последующих обращениях к функции в качестве параметра str1 нужно использовать нулевой указатель. При каждом обращении к функции можно использовать разные наборы разделителей.

Функция size\_t strxfrm(char \*str1, const char \*str2, size\_t count) преобразует строку, адресуемую параметром str2, таким образом, чтобы ее можно было использовать функцией strcmp, и помещает результат преобразования в строку, адресуемую параметром str1. После преобразования результат вызова функции strcmp, использующей параметр str1, будет совпадать с результатом вызова функции strcoll, использующей исходную строку, на которую указывает параметр str2. В массив, адресуемый параметром str1, записывается не более count символов. Таким образом, преобразование связано с учетом особенностей национальных алфавитов кодовой таблицы, устанавливаемой функцией setlocale.

**Практика**

1. **Пример.** Инициализация массива при его определении значением строковой константы в языке С. Нельзя одной строке присвоить значение (скопировать данные) другой строки. Для этого надо написать цикл поэлементного копирования элементов одной строки в другую. Разница между cin>>s и gets(s). **(lab15\_1.cpp)**
2. **Пример.** Вывод кодов и символов первой половины таблицы ASCII, начиная с символа пробел. Наборы символов [ASCII](http://ru.wikipedia.org/wiki/ASCII) (таблицы [ASCII](http://ru.wikipedia.org/wiki/ASCII)) используют однобайтовое кодирование. Вторая половина таблицы содержит символы алфавита страны локализации, символы псевдографики и некоторые другие. Для алфавита одной страны могут использоваться разные коды. **(lab15\_2.cpp)**
3. **Пример.** Для вывода русских букв возможен следующий вариант – сделать свою программу обработки строк с русскими буквами пригодной для всех случаев работы с вводимыми текстами и задаваемыми в исходном тексте программы можно по следующей технологии:

а) все вводимые тексты с русскими буквами должны сразу перекодироваться к кодам соответствующих констант,

б) выполнять все работы с перекодированными введёнными текстами,

в) перед выводом строк (с вводимыми и перекодированными русскими буквами, а также русских строк - констант) выполнить обратное перекодирование. **(lab15\_3.cpp)**

1. **Пример.** Библиотечная функция перекодировки AnsiToOEM(AnsiStr, OemStr: PChar) – тpанслиpует AnsiStr в символьный набоp, опpеделенный OEM. Паpаметpы: AnsiStr – Cтpока (заканчивающаяся пустым символом) символов ANSI; OEMStr – Место, куда копиpуется отpанслиpованная стpока, может совпадать с AnsiStr; возвpащаемое значение -1. **(lab15\_4.cpp)**
2. **Пример.** Написать функцию конкатенации двух строк **(lab15\_5.cpp)**
3. **Пример.** Написать функцию сравнения двух строк по количеству символов **(lab15\_6.cpp)**
4. Стандартная сортировка строк **(lab15\_7.cpp)**
5. **Задание.** Написать программу генерации пароля. Пароль имеет длину 12 символов и состоит из цифр и больших букв английского алфавита. (ABCDEFGHIJKLMNOPQRSTUVWXYZ). Нужно задать массив допустимых символов и при помощи генератора случайных чисел выбрать из него 12 символов пароля. Вывести пароль на экран.
6. **Задание.** Написать и отладить функцию, вычисляющую длину текста. В цикле с консоли вводятся слова, в ответ выдаётся длина введённого слова. Прототип: **int mlen(char t[]).**
7. **Задание.** Создать массив с двойной индексацией. Ввести несколько слов в этот массив. Отсортировать методом поиска минимального элемента и используя стандартные функции работы со строками, вывести на консоль.
8. **Пример.** Составить программу нахождения в тексте на русском языке слова, содержащего максимальное количество букв 'а', 'ы', 'и'.

Решение этой задачи можно разбить на три этапа.

**Алгоритм.** На первом этапе выполнить ввод слов (в конце слов допускаются знаки препинания) текста в элементы массива, состоящего из массивов символов (массива слов), и изменение кодов символов введённых слов на принятые для символьных констант.

На втором этапе найти в массиве слов индекс слова, удовлетворяющего условию задачи, а также количество символов в слове.

На третьем этапе, если нужных слов в тексте нет, то следует вывести соответствующее сообщение, иначе - вывести найденное слово, предварительно удалив в его конце знак препинания, если таковой есть, и перекодировав символы слова для правильного вывода. **(lab15\_11.cpp)**

1. **Пример.** Найти в строке символов все слова, в которых количество цифр составляет не менее заданного значения.

**Алгоритм.** При решении этой задачи сначала строится массив всех слов sl, содержащихся в исходной строке. Затем с помощью функции zifra определяется, содержит ли слово требуемое количество цифровых символов. Слова, удовлетворяющие поставленному условию, переписываются во второй массив zif (массив требуемых слов). Выполнять перекодирование введённых русских букв не нужно, так как сравнений с константами не потребуется, но для вывода поясняющих текстов - констант на русском языке потребуется использовать функцию OutPuts. **(lab15\_12.cpp)**

1. **Пример.** Найти в строке, содержащей буквы только латинского алфавита, максимальное слово и подсчитать, сколько раз в нем встретился каждый символ этого слова.

**Алгоритм.** Для решения этой задачи необходимо выделить все слова, содержащиеся в строке. При выделении очередного слова следует сравнить его с максимальным словом и, если оно превышает максимальное, скопировать текущее слово в максимальное.

Для определения количества раз, которое встретился каждый символ в слове, необходимо каждый символ слова сравнить с каждым из последующих символов этого слова. Для этого надо записать вложенный цикл Во внешнем цикле перебираются все символы слова, а во внутреннем перебираются все последующие символы слова и сравниваются с текущим символом. При их совпадении счетчик увеличивается на единицу. Начальное значение счетчика устанавливается равным единице, т.к. наличие очередного символа означает, что один раз этот символ уже встретился.

При нахождении совпадающего символа его надо удалить из строки, чтобы в дальнейшем этот символ не мог рассматриваться в качестве нового символа. Для этого следует в еще одном вложенном (третьем) цикле произвести сдвиг всех последующих символов на одну позицию в сторону еачала слова. В этом случае количество анализируемых символов (длина строки) уменьшается на единицу. При этом также надо на единицу уменьшить текущее значение параметра цикла, т.к. после сдвига символов строки влево в текущей позиции оказывается еще не анализировавшийся символ. **(lab15\_13.cpp)**

1. **Задание.** Во введенной строке символов, включающей и буквы русского алфавита, найти все слова, начинающиеся с заглавных букв. Найденные слова вывести в алфавитном порядке.
2. **Задание.** Подсчитать в строке количество симметричных слов из нескольких букв и вывести результат.
3. **Задание.** Найти в строке самое короткое и самое длинное слова, в которых нет повторяющихся букв.