**Функции работы со строками из библиотеки обработки строк**

*Прототипы этих функций (если не оговорено особо) находятся в заголовочном файле* **<cstring>**

**Копирование строк**

**сhar \*strcpy( char \* s1, const char \*s2);**

Копирует строку s2 в массив символов s1. Возвращает значение s1

**errno\_t strcpy\_s(**

**char \*strDestination,**

**size\_t numberOfElements,**

**const char \*strSource**

**);**

numberOfElements - размер строки **strDestination**

функция возвращает 0 в случае успеха или значение отличное от 0 в случае ошибки.

**сhar \*strncpy( char \* s1, const char \*s2, size\_t n);**

Копирует не более n символов из строки s2 в массив символов s1.Возвращает значение s1

**errno\_t strncpy\_s(**

**char \*strDest,**

**size\_t numberOfElements,**

**const char \*strSource,**

**size\_t count**

**);**

**count -** количество элементов, которые необходимо скопировать.

**Конкатенация строк**

**сhar \*strcat( char \* s1, const char \*s2);**

Добавляет строку s2 к строке s1. Первый символ строки s2 записывается поверх завершающего нулевого символа строки. Возвращает значение s1

сhar \***strncat**( char \* s1, const char \*s2, size\_t n);

Добавляет не более n символов из строки s2 в строку s1. Первый символ из s2 записывается поверх завершающего нулевого символа в s1. Возвращает значение s1

errno\_t strcat\_s(

char \*strDestination,

size\_t numberOfElements,

const char \*strSource

);

errno\_t strncat\_s(

char \*strDest,

size\_t numberOfElements,

const char \*strSource,

size\_t count

);

**Разбиение на лексемы**

char \***strtok**( char \* s1, const char \*s2);

Последовательность вызовов strtok разбивает строку s1 на «лексемы» — логические куски, такие, как слова в строке текста — разделенные символами, содержащимися в строке s2. Первый вызов содержит в качестве первого аргумента s1, а последующие вызовы для продолжения обработки той же строки, содержат в качестве первого аргумента NULL. При каждом вызове возвращается указатель на текущую лексему. Если при вызове функций лексем больше нет, возвращается NULL

char \*strtok\_s(

char \*strToken,

const char \*strDelimit,

char \*\*context

);

strToken - строка, в которой происходит поиск лексем.

strDelimit - строка содержащая разделители.

context - используется для сохранения информации о позиции поиска между вызовами strtok\_s.

**Определение длины строки**

size\_t **strlen**(const char \*s);

Определяет длину строки s. Возвращает количество символов, предшествующих завершающему нулевому символу

**Ввод -вывод**

**int getchar();** - Возвращает значение символа (если он есть), который пользователь набрал на клавиатуре. После ввода символа нужно нажать клавишу Enter. **stdio.h**

**int getch();** - Аналогично предыдущему, только символ на экране не отображается. Используется чаще для организации задержки выполнения программы. **conio.h**

**int putchar(int c);** - Выводит символ c на экран. В случае успеха возвращает сам символ c, в противном случае - EOF. **stdio.h**

**char \*gets(char \*s);** - Читает символы, включая пробелы и табуляции, до тех пор, пока не встретится символ новой строки, который заменяется нулевым символом. Последовательность прочитанных символов запоминается в области памяти, адресуемой аргументом s. В случае успеха возвращает аргумент s, в случае ошибки - нуль. **stdio.h**

**int puts(const char \*s);** - Выводит строку, заданную аргументом **const char \*s**. **stdio.h**

**Сравнение строк**

**int strcmp(const char \*s1, const char \*s2);** - Сравнивает две строки. Возвращает отрицательное значение, если **s1<s2;** нуль, если **s1==s2;** положительное значение, **если s1>s2**. Параметры - указатели на сравниваемые строки

**int stricmp(const char \*s1, const char \*s2);** - Аналогично предыдущему, только сравнение осуществляется без учета регистра символов

**int strncmp(const char \*s1, const char \*s2, int maxlen);** - Аналогично предыдущему, только сравниваются первые **maxlen** символов

**int strnicmp(const char \*s1, const char \*s2, int maxlen);** - Аналогично предыдущему, только сравниваются первые **maxlen** символов без учета регистра

**Изменения регистра всей строки**

**char \*strlwr(char \*s);** - Преобразует все прописные (большие) буквы в строчные (малые) в строке **s**

**char \*strupr(char \*s);** - Преобразует все строчные (малые) буквы в прописные (большие) в строке **s**

**errno\_t \_strlwr\_s(**

**char \*str,**

**size\_t numberOfElements**

**);**

**errno\_t \_strupr\_s(**

**char \*str,**

**size\_t numberOfElements**

**);**

**Изменение порядка символов на обратный**

**char \*strrev(char \*s);** - Изменяет порядок следования символов в строке на обратный (кроме завершающего нулевого символа). Функция возвращает строку **s**

**Поиск в строке**

**char \*strchr(const char \*s, int c);** - Ищет в строке **s** первое вхождение символа **c**, начиная с начала строки. В случае успеха возвращает указатель на найденный символ, иначе возвращает нуль

**char \*strrchr(const char \*s, int c);** - Аналогично предыдущему, только поиск осуществляется с конца строки

**char \*strstr(const char \*s1, const char \*s2);** - Ищет в строке **s1** строку **s2**. Возвращает адрес первого символа вхождения строки **s2**. Если строка отсутствует - возвращает нуль

**Заполнение и замена символов в строке**

**char \*strnset(char \*s, int c, int n);** - Заполняет строку **s** символами **c**. Параметр **n** задает количество размещаемых символов в строке

errno\_t \_strnset\_s(  
 char \*str,  
 size\_t numberOfElements,  
 int c,  
 size\_t count   
);

**char \*strset(char \*s, int c);** - Заменяет все символы строки **s** заданным символом **c**

**errno\_t \_strset\_s(  
 char \*str,  
 size\_t numberOfElements,  
 int c   
);**

**Преобразование строк**

**double atof(const char \*s);** - Преобразует строку **s** в число с плавающей точкой типа **double**. Заголовочный файл - **math.h**

**int atoi(const char \*s);** - Преобразует строку **s** в число типа **int**. Возвращает значение или нуль, если строку преобразовать нельзя. Заголовочный файл - **stdlib.h**

**long atol(const char \*s);** - Преобразует строку **s** в число типа **long**. Возвращает значение или нуль, если строку преобразовать нельзя. Заголовочный файл - **stdlib.h**

**char \*itoa(int value, char \*s, int radix);** - Преобразует значение целого типа **value** в строку **s**. Возвращает указатель на результирующую строку. Значение **radix** - основание системы счисления, используемое при преобразовании (от 2 до 36). Заголовочный файл - **stdlib.h**

**Обработка символов <ctype.h>**

Функции, представленные в этом разделе, возвращают ненулевое значение (true) в том и только в том случае, если значение аргумента **с** удовлетворяет условиям, указанным в описании функции.

**Функции проверки символа**

**int isalnum (int c) ;Является ли символом или цифрой**

Возвращает true для любого символа, для которого функции **isalpha** или **isdigit** возвращают true

**int isalpha (int с) ;Является ли символом**

Возвращает true для любого символа, для которого функции **isupper** или **islower** возвращают true

**int iscntrl (int с) ;**

Проверяет, является ли аргумент управляющим символом

**int isdigit (int с)** ;

Проверяет, является ли аргумент десятичной цифрой

**int isxdigit (int с) ;**

Проверяет, является ли аргумент шестнадцатеричной цифрой

**int isgraph (int с)** ;

Проверяет, является ли аргумент печатаемым символом, исключая символ пробела (' ')

**int isprint (int с) ;**

Проверяет, является ли аргумент печатаемым символом, включая символ пробела

**int ispunct (int c) ;**

Проверяет, является ли аргумент печатаемым символом, исключая символ пробела (' ') и символы, для которых функция **isalnum** возвращает true

**int isspace (int с) ;**

Проверяет, является ли аргумент стандартным символом разделителем. Стандартными символами разделителями являются: пробел **(' ')**, прогон страницы **(‘\f’)**, новая строка **('\n')**, возврат каретки **('\r'),** символы горизонтальной табуляции **(‘\t')** и вертикальной табуляции **('\v')**

**int islower (int с) ;**

Проверяет, является ли аргумент символом в нижнем регистре

**int isupper (int с) ;**

Проверяет, является ли аргумент символом в верхнем регистре

**Изменения регистра символа**

**int tolower (int с) ;**

**Преобразует символ верхнего регистра в соответствующий символ нижнего регистра.** Если аргументом является символ, для которого isupper возвращает true, и имеется соответствующий символ, для которого islower возвращает true, функция tolower возвратит этот соответствующий символ; в противном случае, аргумент будет возвращен неизмененным (Замечание: к сожалению, для символов кириллицы функция не работает)

**int toupper (int c) ;**

**Преобразует символ нижнего регистра в соответствующий символ верхнего регистра.** Если аргументом является символ, для которого islower возвращает true, и имеется соответствующий символ, для которого isupper возвращает true, функция toupper возвратит этот соответствующий символ; в противном случае, аргумент будет возвращен неизмененным (Замечание: к сожалению, для символов кириллицы функция не работает).