

Для работы со строками используются библиотечные функции, прототипы которых находятся в заголовочных файлах **stdlib.h** и **string.h**.

В программах, в зависимости от типа, вызовы функций для работы со строками задаются в виде:

ИмяФ(СписокАргументов);

или

ИмяПерем=ИмяФ(СписокАргументов);

где **ИмяФ** – имя функции; **СписокАргументов** – список аргументов, передаваемых в тело функции; **ИмяПерем** – идентификатор соответствующего типа.

Например:

y=strlen(st); /*переменной *y* присвоить значение длины строки *st**/

При использовании библиотечных функций следует учитывать некоторые особенности их выполнения и представления символьных данных в памяти.

- Функции, работающие с регистрами, распространяются только на латиницу.
- В C++ некоторые *параметры функций* обработки символов принадлежат типу **int (unsigned)**, поэтому, если число станет больше 128 (255), функция будет работать некорректно.

- Перед первым обращением к строке она должна быть объявлена и проинициализирована. Во многих случаях в качестве начального значения строки бывает необходимо задать *пустую строку*. Такую инициализацию можно выполнить с помощью вызова функции ***strcpy(s, "")***; но более эффективным будет присваивание ****s=0***;. Кроме того пустую строку можно инициализировать ***char s[10]=""***; или ***char s[10]="\0"***;, но при этом размер строки должен быть задан.

- Функции копирования (кроме ***strncpy***) не проверяют *длину строки*. Размер строки-приемника должен быть больше, чем размер источника на 1 символ (для символа '\0').

При вызове функции ***strncpy*** следует помнить, что, если *длина* копируемой строки превосходит *параметр kol*, то строка-получатель не будет завершена символом '\0'. В этом случае такой символ надо дописывать в конец строки вручную.

Функции для работы со строками – файл stdlib.h		
Функция	Прототип	Краткое описание действий
atof	double atof (const char *str);	преобразует строку str в вещественное число типа double
atoi	int atoi (const char *str);	преобразует строку str в целое число типа int
atol	long atol (const char *str);	преобразует строку str в целое число типа long
itoa	char *itoa (int v, char *str, int baz);	преобразует целое v в строку str . При изображении числа используется основание baz (2<=baz<=36).
ltoa	char *ltoa (long v, char *str, int baz);	преобразует длинное целое v в строку str . При изображении числа используется основание baz (2<=baz<=36).
ultoa	char *ultoa (unsignedlong v, char *str, int baz);	преобразует беззнаковое длинное целое v в строку str

Функции для работы со строками – файл string.h		
Функция	Прототип	Краткое описание действий
<i>strcat</i>	<code>char *strcat (char *sp, const char *si);</code>	приписывает строку <i>si</i> к строке <i>sp</i> (конкатенация строк)
<i>strchr</i>	<code>char *strchr (const char *str, int c);</code>	ищет в строке <i>str</i> первое вхождение символа <i>c</i>
<i>strcmp</i>	<code>int strcmp (const char *str1, const char *str2);</code>	сравнивает строки <i>str1</i> и <i>str2</i> . Результат отрицателен, если <i>str1</i> < <i>str2</i> ; равен нулю, если <i>str1</i> == <i>str2</i> , и положителен, если <i>str1</i> > <i>str2</i> (сравнение беззнаковое)
<i>strcpy</i>	<code>char *strcpy (char *sp, const char *si);</code>	копирует байты строки <i>si</i> в строку <i>sp</i>
<i>strcspn</i>	<code>int strcspn (const char *str1, const char *str2);</code>	определяет длину первого сегмента строки <i>str1</i> , содержащего символы, не входящие во множество символов строки <i>str2</i>
<i>strdup</i>	<code>char *strdup (const char *str);</code>	выделяет память и переносит в нее копию строки <i>str</i>
<i>strlen</i>	<code>unsigned strlen(const char *str);</code>	вычисляет длину строки <i>str</i>
<i>strlwr</i>	<code>char *strlwr (char *str);</code>	преобразует буквы верхнего регистра в строке в соответствующие буквы нижнего регистра
<i>strncat</i>	<code>char *strncat (char *sp, const char *si, int kol);</code>	приписывает <i>kol</i> символов строки <i>si</i> к строке <i>sp</i> (конкатенация)
<i>strncmp</i>	<code>int strncmp (const char *str1, const char *str2, int kol);</code>	сравнивает части строк <i>str1</i> и <i>str2</i> , причем рассматриваются первые <i>kol</i> символов. Результат отрицателен, если <i>str1</i> < <i>str2</i> ; равен нулю, если <i>str1</i> == <i>str2</i> , и положителен, если <i>str1</i> > <i>str2</i>
<i>strncpy</i>	<code>char *strncpy (char *sp, const char *si, int kol);</code>	копирует <i>kol</i> символов строки <i>si</i> в строку <i>sp</i>
<i>strnicmp</i>	<code>int strnicmp (char *str1, const char *str2, int kol);</code>	сравнивает не более <i>kol</i> символов строки <i>str1</i> и строки <i>str2</i> , не делая различия регистров (см. функцию <i>strncmp</i>)
<i>strnset</i>	<code>char *strnset (char *str, int c, int kol);</code>	заменяет первые <i>kol</i> символов строки <i>str</i> символом <i>c</i>
<i>strpbrk</i>	<code>char *strpbrk (const char *str1, const char *str2);</code>	ищет в строке <i>str1</i> первое появление любого из множества символов, входящих в строку <i>str2</i>

<i>strchr</i>	<code>char *strchr (const char *str, int c);</code>	ищет в строке <i>str</i> последнее вхождение символа <i>c</i>
<i>strset</i>	<code>char *strset (char *str, int c);</code>	заполняет строку <i>str</i> заданным символом <i>c</i>
<i>strspn</i>	<code>int strspn (const char *str1, const char *str2);</code>	определяет длину первого сегмента строки <i>str1</i> , содержащего только символы, из множества символов строки <i>str2</i>
<i>strstr</i>	<code>char *strstr (const char *str1, const char *str2);</code>	ищет в строке <i>str1</i> подстроку <i>str2</i> . Возвращает указатель на тот элемент в строке <i>str1</i> , с которого начинается подстрока <i>str2</i>
<i>strtod</i>	<code>double strtod (const char *str, char **endptr);</code>	преобразует <i>символьную константу str</i> в число двойной точности. Если <i>endptr</i> не равен <i>NULL</i> , то <i>*endptr</i> возвращается как указатель на символ, при достижении которого прекращено чтение строки <i>str</i>
<i>strtok</i>	<code>char *strtok (char *str1, const char *str2);</code>	ищет в строке <i>str1</i> <i>лексемы</i> , выделенные символами из второй строки
<i>strtol</i>	<code>long strtol (const char *str, char **endptr, int baz);</code>	Преобразует <i>символьную константу str</i> к значению "длинное число" с основанием <i>baz</i> ($2 \leq baz \leq 36$). Если <i>endptr</i> не равен <i>NULL</i> , то <i>*endptr</i> возвращается как указатель на символ, при достижении которого прекращено чтение строки <i>str</i>
<i>strupr</i>	<code>char *strupr (char *str);</code>	преобразует буквы нижнего регистра в строке <i>str</i> в буквы верхнего регистра

Сравнение строк с помощью функции *strcmp* производится побайтово в лексикографическом порядке, то есть в порядке прохождения соответствующих байтов строк в таблице кодирования. Именно поэтому значение элементов в строках зависит от регистра.

При использовании библиотечных функций следует иметь в виду, что указатель на строку и имя массива символов указывают адрес размещения строки в памяти. Это означает, что изменения значений элементов строки сохраняются после завершения работы функции. Чтобы не допустить изменений в строке, используется указатель на константу, которая не позволит модифицировать данные, хранящиеся по адресуемой области памяти.

Пример 74. Программа демонстрирует работу функций из файла *stdlib.h*

Visual Studio 2010

```
#include<iostream>
#include <windows.h>
#include <stdlib.h>
int main()
```

```

{
    SetConsoleOutputCP(1251);
    char sv[]="23.547",
        si[]="1234",
        sl[]="-112424",
        st1[15], st2[25], st3[15];
    long l,t=457821;
    l=atol(sl);
    printf("Преобразование строки в длинное целое число = %ld\n", l);
    printf("Преобразование строки в вещественное число = %f\n", atof(sv));
    printf("Преобразование строки в целое число = %d\n", atoi(si));
    ultoa(t,st1,10);
    printf("Преобразование длинного целого числа в строку = %s\n", st1);
    printf("Преобразование длинного целого числа в строку = %s\n", ultoa(t,st2,2));
    printf("Преобразование длинного целого числа в строку = %s\n", ultoa(t,st3,16));
    return 0;
}

```

Результат выполнения программы:

```

Преобразование строки в длинное целое число = -112424
Преобразование строки в вещественное число = 23.547000
Преобразование строки в целое число = 1234
Преобразование длинного целого числа в строку = 457821
Преобразование длинного целого числа в строку = 1101111110001011101
Преобразование длинного целого числа в строку = 6fc5d
Для продолжения нажмите любую клавишу . . .

```

Visual Studio 2019

```

#include<iostream>
#include <windows.h>
#include <stdlib.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    char sv[] = "23.547",
        si[] = "1234",
        sl[] = "-112424",
        st1[15], st2[25], st3[15];
    long l, t = 457821;
    l = atol(sl);
    printf("Преобразование строки в длинное целое число = %ld\n", l);
    printf("Преобразование строки в вещественное число = %f\n", atof(sv));
    printf("Преобразование строки в целое число = %d\n", atoi(si));
    _ultoa_s(t, st1, 10);
    printf("Преобразование длинного целого числа в строку = %s\n", st1);
    _ultoa_s(t, st2, 2);
    printf("Преобразование длинного целого числа в строку = %s\n", st2);
    _ultoa_s(t, st3, 16);
    printf("Преобразование длинного целого числа в строку = %s\n", st3);
    return 0;
}

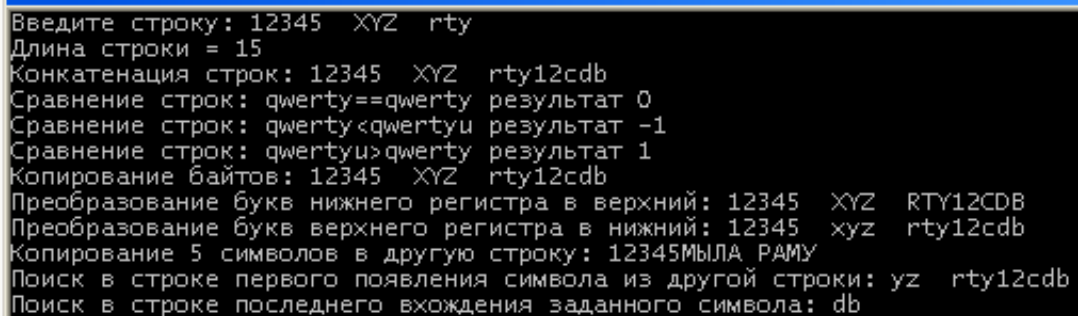
```

Пример 75. Программа демонстрирует работу функций из файла string.h

Visual Studio 2010

```
int main()
{ SetConsoleOutputCP(1251);
  char st[50],sp[100], str[20]="МАМА МЫЛА РАМУ",
    si[]="qwerty",
    sl[]="qwerty",
    sw[]="qwertyu";
  int len=0, sravn1, kol=5;
  printf("Введите строку: ");
  gets(st);
  len=strlen(st);
  printf("Длина строки = %d\n", len);
  printf("Конкатенация строк: %s\n", strcat(st,"12cdb"));
  sravn1=strcmp(si,sl);
  printf("Сравнение строк: %s==%s результат %d\n", si,sl,sravn1);
  printf("Сравнение строк: %s<%s результат %d\n", si,sw,strcmp(si,sw));
  printf("Сравнение строк: %s>%s результат %d\n", sw,si,strcmp(sw,si));
  printf("Копирование байтов: %s\n", strcpy(sp,st));
  printf("Преобразование букв нижнего регистра в верхний: %s\n", strupr(st));
  printf("Преобразование букв верхнего регистра в нижний: %s\n", strlwr(st));
  printf("Копирование %d символов в другую строку: %s\n", kol, strncpy(str,st,kol));
  printf("Поиск в строке первого появления символа из другой строки: %s\n",
  strpbrk(st,si));
  printf("Поиск в строке последнего вхождения заданного символа: %s\n",
  strrchr(st,'d'));
  return 0;
}
```

Результат выполнения программы:



```
Введите строку: 12345 XYZ rty
Длина строки = 15
Конкатенация строк: 12345 XYZ rty12cdb
Сравнение строк: qwerty==qwerty результат 0
Сравнение строк: qwerty<qwertyu результат -1
Сравнение строк: qwertyu>qwerty результат 1
Копирование байтов: 12345 XYZ rty12cdb
Преобразование букв нижнего регистра в верхний: 12345 XYZ RTY12CDB
Преобразование букв верхнего регистра в нижний: 12345 xyz rty12cdb
Копирование 5 символов в другую строку: 12345МАМА МЫЛА РАМУ
Поиск в строке первого появления символа из другой строки: yz rty12cdb
Поиск в строке последнего вхождения заданного символа: db
```

Visual Studio 2019

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <string.h>
using namespace std;
int main()
{
  SetConsoleOutputCP(1251);
  char st[50]="", sp[100]="", str[20]="МАМА МЫЛА РАМУ",
```

```

    si[] = "qwerty",
    sl[] = "qwerty",
    sw[] = "qWerty";
    int len = 0, sravn1, kol = 5;
    printf("Введите строку: ");
    cin.getline(st, 50, '\n');
    len = strlen(st);
    printf ("Длина строки = %d\n", len);
    strcat_s (st, "12cdb");
    printf ("Конкатенация строк: %s\n", st);
    sravn1 = strcmp(si, sl);
    printf ("Сравнение строк: %s==%s результат %d\n", si, sl, sravn1);
    printf ("Сравнение строк: %s<%s результат %d\n", sw, si, strcmp(sw,si));
    printf ("Сравнение строк: %s>%s результат %d\n", si, sw, strcmp(si,sw));
    strcpy_s (sp, st);
    printf("Копирование байтов: %s\n", sp);
    _strupr_s (st);
    printf ("Преобразование букв нижнего регистра в верхний: %s\n", st);
    _strlwr_s (st);
    printf ("Преобразование букв верхнего регистра в нижний: %s\n", st);
    strncpy_ s(str, st, kol);
    printf ("Копирование %d символов в другую строку: %s\n", kol, str);
    printf ("Поиск в строке первого появления символа из другой строки: %s\n", strpbrk(st,
si));
    printf ("Поиск в строке последнего вхождения заданного символа: %s\n", strrchr(st, 'd'));
    return 0;
}

```

```

Введите строку: 12345 XYZ rty
Длина строки = 14
Конкатенация строк: 12345 XYZ rty12cdb
Сравнение строк: qwerty==qwerty результат 0
Сравнение строк: qWerty<qwerty результат -1
Сравнение строк: qwerty>qWerty результат 1
Копирование байтов: 12345 XYZ rty12cdb
Преобразование букв нижнего регистра в верхний: 12345 XYZ RTY12CDB
Преобразование букв верхнего регистра в нижний: 12345 xyz rty12cdb
Копирование 5 символов в другую строку: 12345
Поиск в строке первого появления символа из другой строки: yz rty12cdb
Поиск в строке последнего вхождения заданного символа: db

```

Из файла <ctype.h>:

int isdigit(int); // определяет, цифра или нет

int isalpha(int); //буква

int isupper(int); //буква в верхнем регистре

int islower(int); //буква в нижнем регистре

int isspace(int); //символ — разделитель

int ispunct(int); //символ пунктуации (ни один из вышеупомянутых)

int isalnum(int); //буква или цифра

int toupper(int); //перевод в верхний регистр

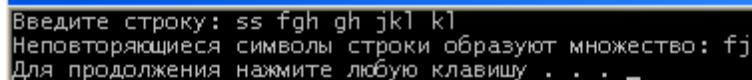
int tolower(int); //перевод в нижний регистр

Пример 75. Поиск множества неповторяющихся символов строки

Visual Studio 2010

```
int main()
{
    SetConsoleOutputCP(1251);
    char st[80];
    int i,j,flag,len;
    printf("Введите строку: ");
    gets(st);
    len=strlen(st);          //длина строки
    printf("Неповторяющиеся символы строки образуют множество: ");
    for (i=1; i<len; i++)
    {
        flag=0;              //флаг проверки на совпадение
        for (j=0; j<i; j++)    //сравнение символа с предыдущими
            if (st[i]==st[j]) flag=1;
        for (j=i+1; j<len; j++) //сравнение символа с последующими
            if (st[i]==st[j]) flag=1;
        if (flag==0) printf("%c", st[i]);
    }
    cout<<"\n";
    system("pause");
    return 0;
}
```

Результат выполнения программы:



```
Введите строку: ss fgh gh jkl kl
Неповторяющиеся символы строки образуют множество: fj
Для продолжения нажмите любую клавишу . . . _
```

Ключевые термины

Конкатенация строк – это результат последовательного *соединения строк*.

Лексикографический порядок – правило сравнения символов, основанное на величине кода внутреннего представления каждого символа.

Пустая строка – это строка единичной длины, содержащая только *символ конца строки*.

Сравнение строк – это результат проверки выполнения отношения "больше", "меньше" или "равно" над строками.

Стандартные функции по работе со строками – это функции обработки строк, прототипы которых входят в *стандартные библиотеки C++*.

Краткие итоги

1. Для работы со строками в языке C++ предусмотрены стандартные функции, прототипы которых включены в *стандартные библиотеки stdlib.h* и *string.h*.

2. При обращении к функциям для работы со строками следует учитывать, что изменение значений элементов строк сохраняются после завершения работы функции.

3. Перед использованием строки в программном коде ее необходимо проинициализировать. Неинициализированные строки могут привести к некорректной работе программы.

4. В некоторых стандартных функциях по работе со строками следует проводить контроль длин параметров.

5. Результат работы некоторых функций требует принудительного добавления к строке *символа конца строки*.

6. Значения элементов строк зависят от регистра.

7. Изменение регистра символов кириллицы в программе может выполняться некорректно.