C++ Programming/Code/Standard C Library/String & Character

< C++ Programming | Code/Standard C Library

Contents

Standard C String & Character [edit]

The Standard C Library includes also routines that deals with characters and strings. You must keep in mind that in C, a string of characters is stored in successive elements of a character array and terminated by the **NULL** character.

```
/* "Hello" is stored in a character array */
char note[SIZE];
note[0] = 'H'; note[1] = 'e'; note[2] = 'l'; note[3] = 'l'; note[4] = 'o'; note[5]
= '\0';
```

Even if outdated this C string and character functions still appear in old code and more so than the previous I/O functions.

atof [edit]

Syntax

```
#include <cstdlib>
double atof( const char *str );
```

The function atof() converts str into a double, then returns that value. str must start with a valid number, but can be terminated with any non-numerical character, other than "E" or "e". For example,

```
x = atof( "42.0is_the_answer" );
```

results in x being set to 42.0.

Related topics

atoi - atol - strtod

(Standard C I/O) sprintf

atoi [edit]

Syntax

```
#include <cstdlib>
int atoi( const char *str );
```

The atoi() function converts str into an integer, and returns that integer. str should start with a whitespace or some sort of number, and atoi() will stop reading from str as soon as a non-numerical character has been read. For example:

```
int i;
i = atoi( "512" );
i = atoi( "512.035" );
i = atoi( "   512.035" );
i = atoi( "   512+34" );
i = atoi( "   512 bottles of beer on the wall" );
```

All five of the above assignments to the variable i would result in it being set to 512.

If the conversion cannot be performed, then atoi() will return zero:

```
int i = atoi( " does not work: 512" );  // results in i == 0
```

A complete C++ implementation of atoi.

```
/*
    Description:
    Program to convert a c-style string into decimal, octal and hex integers.
    Copyright (C) <2015>  <AM>

    This program is free software: you can redistribute it and/or modify it under
the terms of the GNU General Public License as published by the Free Software
Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with
this program.  If not, see <http://www.gnu.org/licenses/>.
*/

// Standard IOSTREAM
#include "iostream"

// Convert a string of Numbers to integer
long int atoifunc(const char* str);

// Function to convert a string of Numbers into an integer
long int atoioct(const char* str);

// Function to convert a string of Numbers into a hexadecimal integer
long long int atoiHex(const char* str);

// Find length of a string
static int strlength(const char *str);

// Get digit from character in input string
static int getDecDigit(const char str);
```

```
// Get a digit from corresponding character in input string
static int getOctDigit(const char str);

// Get a hexadecimal digit from corresponding character in input string
static int getHexDigit(const char str);

using namespace std;

// Application program
int main(void)
{
    char decstr[8] = "";
    char octstr[8] = "";
    char Hexstr[8] = "";
    long int IntNumber=0;
    long int OctNumber=0;
    long long int HexNumber=0;

    cout << "Enter a string of 8 characters in decimal digit form: ";
    cin >> decstr;

    cout << "Enter a string of 8 characters is octal form: ";
    cin >> octstr;

    IntNumber = atoifunc(decstr);
    cout <<"\nYou entered decimal number: " << IntNumber;

    cout << oct;
    OctNumber= atoioct(octstr);

    // Displaying octal number should be done in digits 0-7
    cout <<"\nYou entered octal number: " << OctNumber;

    cout << dec;
    // Displaying octal number should be done in digits 0-7
    cout <<"\nYou entered an oct which is decimal number: " << OctNumber;

    cout << "\nEnter a string of 7 characters in Hex form: ";
    cin >> Hexstr;

    HexNumber=atoiHex(Hexstr);
    cout << hex;
    // Displaying octal number should be done in digits 0-9, A-F
    cout <<"\nYou entered a Hexadecimal number: " << HexNumber;

    cout << dec;

    // Displaying octal number should be done in digits 0-9, A-F
    cout <<"\nYou entered a Hexadecimal which is decimal number: " << HexNumber;

    return 0;
}
```

```
/* Function to convert a string of Numbers into an integer */

/* Get the Number of digits entered as a string.

For each digit, place it in appropriate place of an integer such as digit x 1000
or digit x10000 depending on integer size and input range. The multiple of the
first and subsequent digits would be selected depending on the number of digits.

For example, 123 would be calculated as an integer in the following steps:

1* 100
2* 10
3* 1

The calculated value is then returned by the function.

For example, if the digits entered are 12345
Then,the multipliers are:
str[0] * 10000
str[1] * 1000
str[2] * 100
str[3] * 10
str[4] * 1
Check your machine endianness for correct order of bytes.
*/

long int atoifunc(const char* str)
{

  int declength =strlength(str);
  long int Number =0;
   switch(declength)
    {
          case 0:
          Number += getDecDigit(str[0])*0;
          break;

          // Convert characters to digits with another function.
          case 1:
          Number += getDecDigit(str[0])*1;
          break;

          case 2:

          Number+=getDecDigit(str[0])*10;
          Number+=getDecDigit(str[1])*1;
          break;

          case 3:
          Number+=getDecDigit(str[0])*100;
          Number+=getDecDigit(str[1])*10;
          Number+=getDecDigit(str[2])*1;
          break;
```

```
            case 4:
            Number+=getDecDigit(str[0])*1000;
            Number+=getDecDigit(str[1])*100;
            Number+=getDecDigit(str[2])*10;
            Number+=getDecDigit(str[3])*1;
            break;

            case 5:
            Number+=getDecDigit(str[0])*10000;
            Number+=getDecDigit(str[1])*1000;
            Number+=getDecDigit(str[2])*100;
            Number+=getDecDigit(str[3])*10;
            Number+=getDecDigit(str[4])*1;
            break;

            case 6:
            Number+=getDecDigit(str[0])*100000;
            Number+=getDecDigit(str[1])*10000;
            Number+=getDecDigit(str[2])*1000;
            Number+=getDecDigit(str[3])*100;
            Number+=getDecDigit(str[4])*10;
            Number+=getDecDigit(str[5])*1;
            break;

            case 7:
            Number+=getDecDigit(str[0])*1000000;
            Number+=getDecDigit(str[1])*100000;
            Number+=getDecDigit(str[2])*10000;
            Number+=getDecDigit(str[3])*1000;
            Number+=getDecDigit(str[4])*100;
            Number+=getDecDigit(str[5])*10;
            Number+=getDecDigit(str[6])*1;
            break;

            case 8:
            Number+=getDecDigit(str[0])*10000000;
            Number+=getDecDigit(str[1])*1000000;
            Number+=getDecDigit(str[2])*100000;
            Number+=getDecDigit(str[3])*10000;
            Number+=getDecDigit(str[4])*1000;
            Number+=getDecDigit(str[5])*100;
            Number+=getDecDigit(str[6])*10;
            Number+=getDecDigit(str[7])*1;
            break;

            default:
            Number =0;
            break;
    }

    return Number;
}

// Find length of a string
```

```
static int strlength(const char *str)
{
    int count=0;
    while(str[count]!='\0')
    {
        count++;
    }
    return count;
}

// get a digit from corresponding character in input string
static int getDecDigit(const char str)
{
    int digit =0;
    switch (str)
    {
        case '0':
        digit = 0;
        break;

        case '1':
        digit = 1;
        break;

        case '2':
        digit = 2;
        break;

        case '3':
        digit = 3;
        break;

        case '4':
        digit = 4;
        break;

        case '5':
        digit = 5;
        break;

        case '6':
        digit = 6;
        break;

        case '7':
        digit = 7;
        break;

        case '8':
        digit = 8;
        break;

        case '9':
        digit = 9;
```

```
        break;

        default:
        digit =0;
        break;
    }
    return digit;
}

/* Function to convert a string of Numbers into an integer */

long int atoioct(const char* str)
{
  long int Number =0;
  int stroctlength =strlength(str);
   switch(stroctlength)
    {
          case 0:
          Number += getOctDigit(str[0])*0;
          break;

          // Convert characters to digits with another function.
          case 1:
          Number += getOctDigit(str[0])*1;
          break;

          case 2:

          Number+=getOctDigit(str[0])*8;
          Number+=getOctDigit(str[1])*1;
          break;

          case 3:

          Number+=getOctDigit(str[0])*64;
          Number+=getOctDigit(str[1])*8;
          Number+=getOctDigit(str[2])*1;

          break;

          case 4:
          Number+=getOctDigit(str[0])*512;
          Number+=getOctDigit(str[1])*64;
          Number+=getOctDigit(str[2])*8;
          Number+=getOctDigit(str[3])*1;

          break;

          case 5:
          Number+=getOctDigit(str[0])*4096;
          Number+=getOctDigit(str[1])*512;
          Number+=getOctDigit(str[2])*64;
          Number+=getOctDigit(str[3])*8;
          Number+=getOctDigit(str[4])*1;
```

```
            break;

            case 6:
            Number+=getOctDigit(str[0])*32768;
            Number+=getOctDigit(str[1])*4096;
            Number+=getOctDigit(str[2])*512;
            Number+=getOctDigit(str[3])*64;
            Number+=getOctDigit(str[4])*8;
            Number+=getOctDigit(str[5])*1;
            break;

            case 7:
            Number+=getOctDigit(str[0])*262144;
            Number+=getOctDigit(str[1])*32768;
            Number+=getOctDigit(str[2])*4096;
            Number+=getOctDigit(str[3])*512;
            Number+=getOctDigit(str[4])*64;
            Number+=getOctDigit(str[5])*8;
            Number+=getOctDigit(str[6])*1;
            break;

            default:
            Number =0;
            break;
  }

  return Number;

}

// Get a digit from character input in input string
static int getOctDigit(const char str)
{
    int digit =0;
    switch (str)
    {
        case '0':
        digit = 0;
        break;

        case '1':
        digit = 1;
        break;

        case '2':
        digit = 2;
        break;

        case '3':
        digit = 3;
        break;

        case '4':
        digit = 4;
```

```
        break;

        case '5':
        digit = 5;
        break;

        case '6':
        digit = 6;
        break;

        case '7':
        digit = 7;
        break;

        default:
        digit =0;
        break;
    }
    return digit;
}

// Get a hexadecimal digit from corresponding character in input string
static int getHexDigit(const char str)
{
    int digit =0;
    switch (str)
    {
        case '0':
        digit = 0;
        break;

        case '1':
        digit = 1;
        break;

        case '2':
        digit = 2;
        break;

        case '3':
        digit = 3;
        break;

        case '4':
        digit = 4;
        break;

        case '5':
        digit = 5;
        break;

        case '6':
        digit = 6;
        break;
```

```
        case '7':
        digit = 7;
        break;

        case '8':
        digit = 8;
        break;

        case '9':
        digit = 9;
        break;

        case 'A' :
        case 'a':
        digit =10;
        break;

        case 'B' :
        case 'b':
        digit = 11;
        break;

        case 'C' :
        case 'c':
        digit =12;
        break;

        case 'D' :
        case 'd':
        digit =13;
        break;

        case 'E' :
        case 'e':
        digit = 14;
        break;

        case 'F' :
        case 'f':
        digit = 15;
        break;

        default:
        digit =0;
        break;
    }
    return digit;
}

// Function to convert a string of Numbers into a hexadecimal integer
long long int atoiHex(const char* str)
{
  long long int Number =0;
```

```
int strHexlength =strlength(str);
switch(strHexlength)
  {
        case 0:
        Number += getHexDigit(str[0])*0;
        break;

        // Convert characters to digits with another function.
        // Implicit type conversion from int to long int.
        case 1:
        Number += getHexDigit(str[0])*1;
        break;

        case 2:
        Number+=getHexDigit(str[0])*16;
        Number+=getHexDigit(str[1])*1;
        break;

        case 3:
        Number+=getHexDigit(str[0])*256;
        Number+=getHexDigit(str[1])*16;
        Number+=getHexDigit(str[2])*1;
        break;

        case 4:
        Number+=getHexDigit(str[0])*4096;
        Number+=getHexDigit(str[1])*256;
        Number+=getHexDigit(str[2])*16;
        Number+=getHexDigit(str[3])*1;
        break;

        case 5:
        Number+=getHexDigit(str[0])*65536;
        Number+=getHexDigit(str[1])*4096;
        Number+=getHexDigit(str[2])*256;
        Number+=getHexDigit(str[3])*16;
        Number+=getHexDigit(str[4])*1;
        break;

        case 6:
        Number+=getHexDigit(str[0])*1048576;
        Number+=getHexDigit(str[1])*65536;
        Number+=getHexDigit(str[2])*4096;
        Number+=getHexDigit(str[3])*256;
        Number+=getHexDigit(str[4])*16;
        Number+=getHexDigit(str[5])*1;
        break;

        case 7:
        Number+=getHexDigit(str[0])*16777216;
        Number+=getHexDigit(str[1])*1048576;
        Number+=getHexDigit(str[2])*65536;
        Number+=getHexDigit(str[3])*4096;
        Number+=getHexDigit(str[4])*256;
```

```
            Number+=getHexDigit(str[5])*16;
            Number+=getHexDigit(str[6])*1;
            break;

            default:
            Number =0;
            break;
   }

   return Number;

}
```

Related topics

atof - atol

(Standard C I/O) sprintf

atol [edit]

Syntax

```
#include <cstdlib>
long atol( const char *str );
```

The function atol() converts str into a long, then returns that value. atol() will read from str until it finds any character that should not be in a long. The resulting truncated value is then converted and returned. For example,

```
x = atol( "1024.0001" );
```

results in x being set to 1024L.

Related topics

atof - atoi - strtod

(Standard C I/O) sprintf

isalnum [edit]

Syntax

```
#include <cctype>
int isalnum( int ch );
```

The function isalnum() returns non-zero if its argument is a numeric digit or a letter of the alphabet. Otherwise, zero is returned.

```
char c;
scanf( "%c", &c );
if( isalnum(c) )
  printf( "You entered the alphanumeric character %c\n", c );
```

Related topics

isalpha - iscntrl - isdigit - isgraph - isprint - ispunct - isspace - isxdigit

isalpha [edit]

Syntax

```
#include <cctype>
int isalpha( int ch );
```

The function isalpha() returns non-zero if its argument is a letter of the alphabet. Otherwise, zero is returned.

```
char c;
scanf( "%c", &c );
if( isalpha(c) )
  printf( "You entered a letter of the alphabet\n" );
```

Related topics

isalnum - iscntrl - isdigit - isgraph - isprint - ispunct - isspace - isxdigit

iscntrl [edit]

Syntax

```
#include <cctype>
int iscntrl( int ch );
```

The iscntrl() function returns non-zero if its argument is a control character (between 0 and 0x1F or equal to 0x7F). Otherwise, zero is returned.

Related topics

isalnum - isalpha - isdigit - isgraph - isprint - ispunct- isspace - isxdigit

isdigit [edit]

Syntax

```
#include <cctype>
int isdigit( int ch );
```

The function isdigit() returns non-zero if its argument is a digit between 0 and 9. Otherwise, zero is returned.

```
char c;
scanf( "%c", &c );
if( isdigit(c) )
  printf( "You entered the digit %c\n", c );
```

Related topics

isalnum - isalpha - iscntrl - isgraph - isprint - ispunct- isspace - isxdigit

isgraph [edit]

Syntax

```
#include <cctype>
int isgraph( int ch );
```

The function isgraph() returns non-zero if its argument is any printable character other than a space (if you can see the character, then isgraph() will return a non-zero value). Otherwise, zero is returned.

Related topics

isalnum - isalpha - iscntrl - isdigit - isprint - ispunct - isspace - isxdigit

islower [edit]

Syntax

```
#include <cctype>
int islower( int ch );
```

The islower() function returns non-zero if its argument is a lowercase letter. Otherwise, zero is returned.

Related topics

isupper

isprint [edit]

Syntax

```
#include <cctype>
int isprint( int ch );
```

The function isprint() returns non-zero if its argument is a printable character (including a space). Otherwise, zero is returned.

Related topics

isalnum - isalpha - iscntrl - isdigit - isgraph - ispunct - isspace

ispunct [edit]

Syntax

```
#include <cctype>
int ispunct( int ch );
```

The ispunct() function returns non-zero if its argument is a printing character but neither alphanumeric nor a space. Otherwise, zero is returned.

Related topics

isalnum - isalpha - iscntrl - isdigit - isgraph - isspace- isxdigit

isspace [edit]

Syntax

```
#include <cctype>
int isspace( int ch );
```

The isspace() function returns non-zero if its argument is some sort of space (i.e. single space, tab, vertical tab, form feed, carriage return, or newline). Otherwise, zero is returned.

Related topics

isalnum - isalpha - iscntrl - isdigit - isgraph - isprint - ispunct - isxdigit

isupper [edit]

Syntax

```
#include <cctype>
int isupper( int ch );
```

The isupper() function returns non-zero if its argument is an uppercase letter. Otherwise, zero is returned.

Related topics

islower - tolower

isxdigit [edit]

Syntax

```
#include <cctype>
int isxdigit( int ch );
```

The function isxdigit() returns non-zero if its argument is a hexadecimal digit (i.e. A-F, a-f, or 0-9). Otherwise, zero is returned.

Related topics

isalnum - isalpha - iscntrl - isdigit - isgraph - ispunct - isspace

memchr [edit]

Syntax

```
#include <cstring>
void *memchr( const void *buffer, int ch, size_t count );
```

The memchr() function looks for the first occurrence of ch within count characters in the array pointed to by buffer. The return value points to the location of the first occurrence of ch, or **NULL** if ch isn't found. For example:

```
char names[] = "Alan Bob Chris X Dave";
if( memchr(names,'X',strlen(names)) == NULL )
  printf( "Didn't find an X\n" );
else
  printf( "Found an X\n" );
```

Related topics

memcmp - memcpy - strstr

memcmp [edit]

Syntax

```
#include <cstring>
int memcmp( const void *buffer1, const void *buffer2, size_t count );
```

The function memcmp() compares the first countcharacters of buffer1 and buffer2. The return values are as follows:

Return value	Explanation
less than 0	buffer1 is less than buffer2
equal to 0	buffer1 is equal to buffer2
greater than 0	buffer1 is greater than buffer2

Related topics

memchr - memcpy - memset - strcmp

memcpy [edit]

Syntax

```
#include <cstring>
void *memcpy( void *to, const void *from, size_t count );
```

The function memcpy() copies count characters from the array from to the array to. The return value of memcpy() is to. The behavior of memcpy() is undefined if to and from overlap.

Related topics

memchr - memcmp - memmove - memset - strcpy - strlen - strncpy

memmove [edit]

Syntax

```
#include <cstring>
void *memmove( void *to, const void *from, size_t count );
```

The memmove() function is identical to memcpy(), except that it works even if to and from overlap.

Related topics

memcpy - memset

memset [edit]

Syntax

```
#include <cstring>
void* memset( void* buffer, int ch, size_t count );
```

The function memset() copies ch into the first countcharacters of buffer, and returns buffer. memset() is useful for intializing a section of memory to some value. For example, this command:

```
const int ARRAY_LENGTH;
char the_array[ARRAY_LENGTH];
...
// zero out the contents of the_array
memset( the_array, '\0', ARRAY_LENGTH );
```

...is a very efficient way to set all values of the_array to zero.

The table below compares two different methods for initializing an array of characters: a `for` loop versus memset(). As the size of the data being initialized increases, memset() clearly gets the job done much more quickly:

Input size	Initialized with a `for` loop	Initialized with memset()
1000	0.016	0.017
10000	0.055	0.013
100000	0.443	0.029
1000000	4.337	0.291

Related topics

memcmp - memcpy - memmove

strcat [edit]

Syntax

```
#include <cstring>
char *strcat( char *str1, const char *str2 );
```

The strcat() function concatenates str2 onto the end of str1, and returns str1. For example:

```
printf( "Enter your name: " );
scanf( "%s", name );
title = strcat( name, " the Great" );
printf( "Hello, %s\n", title );  ;
```

Note that strcat() does not perform bounds checking, and thus risks overrunning str1 or str2. For a similar (and safer) function that includes bounds checking, see strncat().

Related topics

strchr - strcmp - strcpy - strncat

strchr [edit]

Syntax

```
#include <cstring>
char *strchr( const char *str, int ch );
```

The function strchr() returns a pointer to the first occurrence of ch in str, or **NULL** if ch is not found.

Related topics

strcat - strcmp - strcpy - strlen - strncat - strncmp - strncpy - strpbrk - strrchr -strspn - strstr - strtok

strcmp [edit]

Syntax

```
#include <cstring>
int strcmp( const char *str1, const char *str2 );
```

The function strcmp() compares str1 and str2, then returns:

Return value	Explanation
less than 0	str1 is less than str2
equal to 0	str1 is equal to str2
greater than 0	str1 is greater than str2

For example:

```
printf( "Enter your name: " );
scanf( "%s", name );
```

```
if( strcmp( name, "Mary" ) == 0 ) {
  printf( "Hello, Dr. Mary!\n" );
}
```

Note that if str1 or str2 are missing a null-termination character, then strcmp() may not produce valid results. For a similar (and safer) function that includes explicit bounds checking, see strncmp().

Related topics

memcmp - strcat - strchr - strcoll - strcpy - strlen - strncmp - strxfrm

strcoll [edit]

Syntax

```
#include <cstring>
int strcoll( const char *str1, const char *str2 );
```

The strcoll() function compares str1 and str2, much like strcmp(). However, strcoll() performs the comparison using the locale specified by the (Standard C Date & Time) setlocale() function.

Related topics

strcmp - strxfrm

(Standard C Date & Time) setlocale

strcpy [edit]

Syntax

```
#include <cstring>
char *strcpy( char *to, const char *from );
```

The strcpy() function copies characters in the string 'from to the string to, including the null termination. The return value is to.

Note that strcpy() does not perform bounds checking, and thus risks overrunning from or to. For a similar (and safer) function that includes bounds checking, see strncpy().

Related topics

memcpy - strcat - strchr - strcmp - strncmp - strncpy

strcspn [edit]

Syntax

```
#include <cstring>
size_t strcspn( const char *str1, const char *str2 );
```

The function strcspn() returns the index of the first character in str1 that matches any of the characters in str2.

Related topics

strpbrk - strrchr - strstr - strtok

strerror [edit]

Syntax

```
#include <cstring>
char *strerror( int num );
```

The function strerror() returns an implementation defined string corresponding to num. If an error occurred, the error is located within the global variable **errno**.

Related topics

perror

strlen [edit]

Syntax

```
#include <cstring>
size_t strlen( char *str );
```

The strlen() function returns the length of str(determined by the number of characters before null termination).

Related topics

memcpy - strchr - strcmp - strncmp

strncat [edit]

Syntax

```
#include <cstring>
char *strncat( char *str1, const char *str2, size_t count );
```

The function strncat() concatenates at most countcharacters of str2 onto str1, adding a null termination. The resulting string is returned.

Related topics

strcat - strchr - strncmp - strncpy

strncmp [edit]

Syntax

```
#include <cstring>
int strncmp( const char *str1, const char *str2, size_t count );
```

The strncmp() function compares at most count characters of str1 and str2. The return value is as follows:

Return value	Explanation
less than 0	str1 is less than str2
equal to 0	str1 is equal to str2
greater than 0	str1 is greater than str2

If there are less than count characters in either string, then the comparison will stop after the first null termination is encountered.

Related topics

strchr - strcmp - strcpy - strlen - strncat - strncpy

strncpy [edit]

Syntax

```
#include <cstring>
char *strncpy( char *to, const char *from, size_t count );
```

The strncpy() function copies at most count characters of from to the string to. Only if from has less than countcharacters, is the remainder padded with '\0' characters. The return value is the resulting string.

> **Note:**
> Using strings not padded with the '\0' character can create security vulnerabilities.

Related topics

memcpy - strchr - strcpy - strncat - strncmp

strpbrk [edit]

Syntax

```
#include <cstring>
char * strpbrk( const char *str, const char *ch );
```

The function strchr() returns a pointer to the first occurrence of any character within ch in str, or **NULL** if no characters were not found.

Related topics

strchr - strrchr - strstr

strrchr [edit]

Syntax

```
#include <cstring>
char *strrchr( const char *str, int ch );
```

The function strrchr() returns a pointer to the last occurrence of ch in str, or **NULL** if no match is found.

Related topics

strchr - strcspn - strpbrk - strspn - strstr - strtok

strspn [edit]

Syntax

```
#include <cstring>
size_t strspn( const char *str1, const char *str2 );
```

The strspn() function returns the index of the first character in str1 that doesn't match any character in str2.

Related topics

strchr - strpbrk - strrchr - strstr - strtok

strstr [edit]

Syntax

```
#include <cstring>
char *strstr( const char *str1, const char *str2 );
```

The function strstr() returns a pointer to the first occurrence of str2 in str1, or **NULL** if no match is found. If the length of str2 is zero, then strstr() will simply return str1.

For example, the following code checks for the existence of one string within another string:

```
char* str1 = "this is a string of characters";
char* str2 = "a string";
char* result = strstr( str1, str2 );
if( result == NULL ) printf( "Could not find '%s' in '%s'\n", str2, str1 );
  else printf( "Found a substring: '%s'\n", result );
```

When run, the above code displays this output:

```
Found a substring: 'a string of characters'
```

Related topics

memchr - strchr - strcspn - strpbrk - strrchr - strspn - strtok

strtod [edit]

Syntax

```
#include <cstdlib>
double strtod( const char *start, char **end );
```

The function strtod() returns whatever it encounters first in start as a double. end is set to point at whatever is left in start after that double. If overflow occurs, strtod() returns either **HUGE_VAL** or **-HUGE_VAL**.

```
x = atof( "42.0is_the_answer" );
```

results in x being set to 42.0.

Related topics

atof

strtok [edit]

Syntax

```
#include <cstring>
char *strtok( char *str1, const char *str2 );
```

The strtok() function returns a pointer to the next "token" in str1, where str2 contains the delimiters that determine the token. strtok() returns **NULL** if no token is found. In order to convert a string to tokens, the first call to strtok() should have str1 point to the string to be tokenized. All calls after this should have str1 be **NULL**.

For example:

```
char str[] = "now # is the time for all # good men to come to the # aid of their
country";
char delims[] = "#";
```

```
char *result = NULL;
result = strtok( str, delims );
while( result != NULL ) {
  printf( "result is \"%s\"\n", result );
  result = strtok( NULL, delims );
}
```

The above code will display the following output:

```
result is "now "
result is " is the time for all "
result is " good men to come to the "
result is " aid of their country"
```

Related topics

strchr - strcspn - strpbrk - strrchr - strspn - strstr

strtol [edit]

Syntax

```
#include <cstdlib>
long strtol( const char *start, char **end, int base );
```

The strtol() function returns whatever it encounters first in start as a long, doing the conversion to base if necessary. end is set to point to whatever is left in startafter the long. If the result can not be represented by a long, then strtol() returns either **LONG_MAX** or **LONG_MIN**. Zero is returned upon error.

Related topics

atol - strtoul

strtoul [edit]

Syntax

```
#include <cstdlib>
unsigned long strtoul( const char *start, char **end, int base );
```

The function strtoul() behaves exactly like strtol(), except that it returns an **unsigned** long rather than a mere long.

Related topics

strtol

strxfrm [edit]

Syntax

```
#include <cstring>
size_t strxfrm( char *str1, const char *str2, size_t num );
```

The strxfrm() function manipulates the first numcharacters of str2 and stores them in str1. The result is such that if a strcoll() is performed on str1 and the old str2, you will get the same result as with a strcmp().

Related topics

strcmp - strcoll

tolower [edit]

Syntax

```
#include <cctype>
int tolower( int ch );
```

The function tolower() returns the lowercase version of the character ch.

Related topics

isupper - toupper

toupper [edit]

Syntax

```
#include <cctype>
int toupper( int ch );
```

The toupper() function returns the uppercase version of the character ch.

Related topics

tolower