Strings

String Functions (*manipulation* functions)

`char *strcpy (char *s1, const char *s2 )`

Copies the string s2 into the character array s1. The value of s1 is returned.

`char *strncpy( char *s1, const char *s2, int n )`

Copies at most n characters of the string s2 into the character array s1. The value of s1 is returned. But consider the case:

```
char *s1 = “This is a test”;
char *s2 = “The cat”;
strncpy(s1, s2, 3);
cout << s1 << endl;
```

output: **Thes is a test** // will not always null terminate for you

`char *strcat( char *s1, const char *s2 )`

Appends the string s2 to the string s1. The first character of s2 overwrites the terminating null character of s1. The value of s1 is returned.

`char *strncat( char *s1, const char *s2, int n )`

Appends at most n characters of string s2 to string s1. The first character of s2 overwrites the terminating null character of s1. The value of s1 is returned.

`int strcmp( const char *s1, const char *s2 )`

Compares the string s1 to the string s2. The function returns a value of 0, less than 0, ore greater than 0 if s1 is equal to, less than, or greater than s2, respectively.

`int strncmp( const char *s1, const char *s2, int n )`

Compares up to n character of the string s1 to the string s2. The function returns 0, less than 0, or greater than 0 if s1 is equal to, less than, or greater than s2, respectively.

`int stricmp( const char *s1, const char *s2 )`

Compares the string s1 to the string s2 without case sensitivity. The function returns a value of 0, less than 0, ore greater than 0 if s1 is equal to, less than, or greater than s2, respectively.

```
char *strtok( char *s1, const char *s2 )
```

A sequence of calls to strtok breaks string s1 into "tokens" – logical pieces such as words in a line of text – separated by characters contained in string s2. The first call contains s1 as the first argument, and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

```
char string[] = "This is a sentence with 7 tokens";
char *tokenptr;

tokenptr = strtok( string, " " );

while(tokenptr != NULL)
{
   cout << tokenptr << endl;
   tokenptr = strtok( NULL, " ");
};
```

output:
This
is
a
sentence
with
7
tokens

```
int strlen( const char *s )
```

Determines the length of string s. The number of characters preceding the terminating null character is returned.

```
char *_strlwr( char *s);
```

Converts all uppercase letters (A to Z) in string s to lowercase (a to z). Returns a pointer to the string s.

```
char *_strupr( char *s);
```

Converts all lowercase letters (a to z) in string s to uppercase (A to Z). Returns a pointer to the string s.

```
char *_strrev( char *s);
```

Changes all characters in a string to reverse order, except the terminating NULL character.

```
char *strset( char *s, int ch);
```

Sets all characters in string s to the character ch. It quits when the terminating null character is found. Returns pointer to string s.

```
char * strnset( char *s, int ch, int n);
```

Copies the character ch into the first n bytes of string s. It quits when n characters have been set, or when a NULL character is found. Returns pointer to string s.

String Functions (*conversion* functions) <stdlib.h> or <math.h>

```
double atof( const char *nptr);
```

Converts nptr to double or float.

```
char string[20] = “99.05”;
double d;

d = atof(string);
cout << d << endl;

output:  99.05
```

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

Converts nptr to int

```
long atol( const char *nptr);
```

Converts nptr to long int

```
double strtod( const char *nptr, char **remptr);
```

Converts nptr to a double/float and sets remptr to the remainder of the string.

```
char string[20] = “99.05abc”;
char *remptr;
double d;

d = strtod( string, &remptr);

cout << d << endl << remptr << endl;

output:   99.05
          abc
```

```
long strtol( const char *nptr, char **remptr, int base);
```

Converts nptr to a signed long and sets remptr to the remainder of the string. Base allows you to specify the conversion. base(2) binary, base(8) octal, base(16) hexadecimal. Valid ranges are 0, 2-36;

```
unsigned long strtoul( const char *nptr, char **remptr,
                       int base );
```

Converts nptr to an unsigned long and sets remptr to the remainder of the string. Base allows you to specify the conversion. base(2) binary, base(8) octal, base(16) hexadecimal. Valid ranges are 0, 2-36;
Ranges: 0 the first number of the sting decides

```
char string[] = "179ahz!#";
char *remptr;

long default0;
long base8;
long base10;
long base16;
long base24;

default0 = strtol(string, &remptr, 0);
cout << default0 << "    " << *remptr << endl;

char string[] = "179ahz!#";
base8 = strtol(string, &remptr, 8);
cout << base8 << "    " << *remptr << endl;

base10 = strtol(string, &remptr, 10);
cout << base10 << "    " << *remptr << endl;

base16 = strtol(string, &remptr, 16);
cout << base16 << "    " << *remptr << endl;

base24 = strtol(string, &remptr, 24);
cout << base24 << "    " << *remptr << endl;
```

outputs:

```
179       ahz!#
15        9ahz!#
179       ahz!#
96683     hz!#
10415513  z!#
```

String Functions (*Searching* functions)

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

Locates the first occurrence of character c in string s. If c is found, a pointer to the character c in s is returned, otherwise a NULL pointer is returned.

```
int strcspn( const char *s1, const char *s2);
```

Finds the initial segment of string s1 that consists entirely of characters NOT from string s2.

```
char *string1 = "1234567890";
char *string2 = "747DC8";

int length;

length = strcspn(string1, string2);
cout << "Character where strings intersect is “
     << “at position " << length << endl;
```

output: **Character where strings intersect is at position 3**

```
int strspn( const char *s1, const char *s2);
```

Finds the initial segment of string s1 that consist entirely of characters from sting s2.

```
char *string1 = "1234567890";

char *string2 = "1234DC8";

int length;

length = strspn(string1, string2);

cout << "Character where strings differ "

     << "is at position " << length << endl;
```

Output: **Character where strings differ is at position 4**

```
char *strpbrk( const char *s1, const char *s2);
```

Scans string s1 for the first occurrence of any character appearing in s2. Upon success, strpbrk returns a pointer to the first occurrence of any of the characters in s2.

```
char *string1 = "abcdefghijklmnopqrstuvwxyz";
char *string2 = "onm";
char *ptr;

ptr = strpbrk(string1, string2);

if (ptr)
   cout << "strpbrk found first character: "
        << ptr << endl;
else
   cout << "strpbrk didn't find character in set"
        << endl;
```

output: **strpbrk found first character: mnopqrstuvwxyz**

```
char *strrchr( const char *s1, int c);
```

Scans string s1 in the reverse direction looking for a specific character specified by the integer c;

```
char string[15];
char *ptr,
     c = 'r';

strcpy(string, "This is a string");
ptr = strrchr(string, c);
if (ptr)
  cout << "The character " << c << " is at position: "
       << ptr << endl;
else
  cout << "The character was not found" << endl;
```

output: **The character r is at position: ring**

```
char *strstr( const char *s1, const char *s2);
```

Scans string s1 for the first occurrence of the substring s2. Upon success a pointer to the element in s1 where s2 begins is returned. (points to s2 in s1). Otherwise a NULL pointer is returned.

```
char *str1 = "Dr. Toni Logar",
     *str2 = "Toni",
     *ptr;

ptr = strstr(str1, str2);
cout << "The substring is: " << ptr << endl;
```

Output: **The substring is: Toni Logar**

String Functions (*Miscellaneous* Functions)

```
char *strerror( int errornum);
```

Takes an error number and returns a pointer to an error message string associated with that error number. // dos only

```
#include <stdio.h>
#include <errno.h>

int main(void)
{
   char *buffer;
   buffer = strerror(errno);
   cout << “Error: “ << buffer << endl;
   return 0;
}
```

```
  errno = 2;
  output:  No such file or directory
```

```
char *_strerror(const char *s);
```

Generates a custom error message.

```
#include <stdio.h>

int main(void)
{
   FILE *fp;

   // open a file for writing
   fp = fopen("TEST.$$$", "w");

   // force an error condition by attempting to read
   if (!fp) fgetc(fp);

   if (ferror(fp))
      // display a custom error message
      cout << _strerror("Custom Message") << endl;

   fclose(fp);
   return 0;
}
```