Escola de Engenharia
Universidade do Minho

# Programação – Aula Teórica 8

## Caracteres e Strings


**Luís Paulo Reis**

lpreis@dsi.uminho.pt

**Professor Associado do Departamento de Sistemas de Informação, Escola de Engenharia, Universidade do Minho, Portugal**


**(Slides Baseados em Deitel e Deitel 2010 e L.P.Reis et al., 2006 )**

Escola de Engenharia
Universidade do Minho

# Chars and Strings

## Outline

Escola de Engenharia
Universidade do Minho

# Objectives

- In this lesson, you will learn:
  - To be able to use the functions of the character handling library (`ctype`)
  - To be able to use the string and character input/output functions of the standard input/output library (`stdio`)
  - To be able to use the string conversion functions of the general utilities library (`stdlib`)
  - To be able to use the string processing functions of the string handling library (`string`)
  - To appreciate the power of function libraries as a means of achieving software reusability

Escola de Engenharia
Universidade do Minho

# 8.1 Introduction

- **Introduce some standard library functions**
  - Easy string and character processing
  - Programs can process characters, strings, lines of text, and blocks of memory
- **These techniques used to make**
  - Word processors
  - Page layout software
  - Typesetting programs

# 8.2 Fundamentals of Strings and Characters

- **Characters**
  - Building blocks of programs
    - Every program is a sequence of meaningfully grouped characters
  - Character constant
    - An `int` value represented as a character in single quotes
    - `'z'` represents the integer value of `z`
- **Strings**
  - Series of characters treated as a single unit
    - Can include letters, digits and special characters (`*`, `/`, `$`)
  - String literal (string constant) - written in double quotes
    - `"Hello"`
  - Strings are arrays of characters
    - String a pointer to first character
    - Value of string is the address of first character

# 8.2 Fundamentals of Strings and Characters

- **String definitions**
  - Define as a character array or a variable of type `char *`

```
char color[] = "blue";
char *colorPtr = "blue";
```

  - Remember that strings represented as character arrays end with `'\0'`
    - `color` has `5 elements`
- **Inputting strings**
  - Use `scanf`

```
scanf("%s", word);
```

    - Copies input into `word[]`
    - Do not need & (because a string is a pointer)
  - Remember to leave room in the array for `'\0'`

Escola de Engenharia
Universidade do Minho

# 8.3 Character Handling Library

- **Character handling library**
  - Includes functions to perform useful tests and manipulations of character data
  - Each function receives a character (an `int`) or `EOF` as an argument
- **The following slide contains a table of all the functions in `<ctype.h>`**

Escola de Engenharia
Universidade do Minho

# 8.3 Character Handling Library

| Prototype | Description |
|---|---|
| `int isdigit( int c );` | Returns `true` if `c` is a digit and `false` otherwise. |
| `int isalpha( int c );` | Returns `true` if `c` is a letter and `false` otherwise. |
| `int isalnum( int c );` | Returns `true` if `c` is a digit or a letter and `false` otherwise. |
| `int isxdigit( int c );` | Returns `true` if `c` is a hexadecimal digit character and `false` otherwise. |
| `int islower( int c );` | Returns `true` if `c` is a lowercase letter and `false` otherwise. |
| `int isupper( int c );` | Returns `true` if `c` is an uppercase letter; `false` otherwise. |
| `int tolower( int c );` | If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged. |
| `int toupper( int c );` | If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged. |
| `int isspace( int c );` | Returns `true` if `c` is a white-space character—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`), or vertical tab (`'\v'`)—and `false` otherwise |
| `int iscntrl( int c );` | Returns `true` if `c` is a control character and `false` otherwise. |
| `int ispunct( int c );` | Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise. |
| `int isprint( int c );` | Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise. |
| `int isgraph( int c );` | Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise. |

```c
/* Fig. 8.2: fig08_02.c
   Using functions isdigit, isalpha, isalnum, and isxdigit */
#include <stdio.h>
#include <ctype.h>

int main()
{
   printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
       isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
       isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
       "According to isalpha:",
       isalpha( 'A' ) ? "A is a " : "A is not a ", "letter",
       isalpha( 'b' ) ? "b is a " : "b is not a ", "letter",
       isalpha( '&' ) ? "& is a " : "& is not a ", "letter",
       isalpha( '4' ) ? "4 is a " : "4 is not a ", "letter" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
       "According to isalnum:",
       isalnum( 'A' ) ? "A is a " : "A is not a ",
       "digit or a letter",
       isalnum( '8' ) ? "8 is a " : "8 is not a ",
       "digit or a letter",
       isalnum( '#' ) ? "# is a " : "# is not a ",
       "digit or a letter" );

```

```
28     printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
29        "According to isxdigit:",
30        isxdigit( 'F' ) ? "F is a " : "F is not a ",
31        "hexadecimal digit",
32        isxdigit( 'J' ) ? "J is a " : "J is not a ",
33        "hexadecimal digit",
34        isxdigit( '7' ) ? "7 is a " : "7 is not a ",
35        "hexadecimal digit",
36        isxdigit( '$' ) ? "$ is a " : "$ is not a ",
37        "hexadecimal digit",
38        isxdigit( 'f' ) ? "f is a " : "f is not a ",
39        "hexadecimal digit" );
40
41     return 0; /* indicates successful termination */
42
43 } /* end main */
```

```
According to isdigit:
8 is a digit
# is not a digit

According to isalpha:
A is a letter
b is a letter
& is not a letter
4 is not a letter

According to isalnum:
A is a digit or a letter
8 is a digit or a letter
# is not a digit or a letter

According to isxdigit:
F is a hexadecimal digit
J is not a hexadecimal digit
7 is a hexadecimal digit
$ is not a hexadecimal digit
f is a hexadecimal digit
```

```c
/* Fig. 8.3: fig08_03.c
   Using functions islower, isupper, tolower, toupper */
#include <stdio.h>
#include <ctype.h>

int main()
{
   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
           "According to islower:",
           islower( 'p' ) ? "p is a " : "p is not a ",
           "lowercase letter",
           islower( 'P' ) ? "P is a " : "P is not a ",
           "lowercase letter",
           islower( '5' ) ? "5 is a " : "5 is not a ",
           "lowercase letter",
           islower( '!' ) ? "! is a " : "! is not a ",
           "lowercase letter" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
           "According to isupper:",
           isupper( 'D' ) ? "D is an " : "D is not an ",
           "uppercase letter",
           isupper( 'd' ) ? "d is an " : "d is not an ",
           "uppercase letter",
           isupper( '8' ) ? "8 is an " : "8 is not an ",
           "uppercase letter",
           isupper( '$' ) ? "$ is an " : "$ is not an ",
           "uppercase letter" );

```

```
30    printf( "%s%c\n%s%c\n%s%c\n%s%c\n",
31            "u converted to uppercase is ", toupper( 'u' ),
32            "7 converted to uppercase is ", toupper( '7' ),
33            "$ converted to uppercase is ", toupper( '$' ),
34            "L converted to lowercase is ", tolower( 'L' ) );
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
```

```
According to islower:
p is a lowercase letter
P is not a lowercase letter
5 is not a lowercase letter
! is not a lowercase letter

According to isupper:
D is an uppercase letter
d is not an uppercase letter
8 is not an uppercase letter
$ is not an uppercase letter

u converted to uppercase is U
7 converted to uppercase is 7
$ converted to uppercase is $
L converted to lowercase is l
```

```c
/* Fig. 8.4: fig08_04.c
   Using functions isspace, iscntrl, ispunct, isprint, isgraph */
#include <stdio.h>
#include <ctype.h>

int main()
{
   printf( "%s\n%s%s%s\n%s%s%s\n%s%s\n\n",
      "According to isspace:",
      "Newline", isspace( '\n' ) ? " is a " : " is not a ",
      "whitespace character", "Horizontal tab",
      isspace( '\t' ) ? " is a " : " is not a ",
      "whitespace character",
      isspace( '%' ) ? "% is a " : "% is not a ",
      "whitespace character" );

   printf( "%s\n%s%s%s\n%s%s\n\n", "According to iscntrl:",
      "Newline", iscntrl( '\n' ) ? " is a " : " is not a ",
      "control character", iscntrl( '$' ) ? "$ is a " :
      "$ is not a ", "control character" );

```

```c
    printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
        "According to ispunct:",
        ispunct( ';' ) ? "; is a " : "; is not a ",
        "punctuation character",
        ispunct( 'Y' ) ? "Y is a " : "Y is not a ",
        "punctuation character",
        ispunct( '#' ) ? "# is a " : "# is not a ",
        "punctuation character" );

    printf( "%s\n%s%s\n%s%s%s\n\n", "According to isprint:",
        isprint( '$' ) ? "$ is a " : "$ is not a ",
        "printing character",
        "Alert", isprint( '\a' ) ? " is a " : " is not a ",
        "printing character" );

    printf( "%s\n%s%s\n%s%s%s\n",  "According to isgraph:",
        isgraph( 'Q' ) ? "Q is a " : "Q is not a ",
        "printing character other than a space",
        "Space", isgraph( ' ' ) ? " is a " : " is not a ",
        "printing character other than a space" );

    return 0; /* indicates successful termination */

} /* end main */
```

```
According to isspace:
Newline is a whitespace character
Horizontal tab is a whitespace character
% is not a whitespace character

According to iscntrl:
Newline is a control character
$ is not a control character

According to ispunct:
; is a punctuation character
Y is not a punctuation character
# is a punctuation character

According to isprint:
$ is a printing character
Alert is not a printing character

According to isgraph:
Q is a printing character other than a space
Space is not a printing character other than a space
```

Escola de Engenharia
Universidade do Minho

# 8.4 String Conversion Functions

- **Conversion functions**
  - In `<stdlib.h>` (general utilities library)
- **Convert strings of digits to integer and floating-point values**

| Function prototype | Function description |
|---|---|
| `double atof( const char *nPtr );` | Converts the string `nPtr` to `double`. |
| `int atoi( const char *nPtr );` | Converts the string `nPtr` to `int`. |
| `long atol( const char *nPtr );` | Converts the string `nPtr` to `long int`. |
| `double strtod( const char *nPtr, char **endPtr );` | Converts the string `nPtr` to `double`. |
| `long strtol( const char *nPtr, char **endPtr, int base );` | Converts the string `nPtr` to `long`. |
| `unsigned long strtoul( const char *nPtr, char **endPtr, int base );` | Converts the string `nPtr` to `unsigned long`. |

```c
/* Fig. 8.6: fig08_06.c
   Using atof */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   double d; /* variable to hold converted string */

   d = atof( "99.0" );

   printf( "%s%.3f\n%s%.3f\n",
           "The string \"99.0\" converted to double is ", d,
           "The converted value divided by 2 is ",
           d / 2.0 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string "99.0" converted to double is 99.000
The converted value divided by 2 is 49.500
```

```c
/* Fig. 8.7: fig08_07.c
   Using atoi */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int i; /* variable to hold converted string */

   i = atoi( "2593" );

   printf( "%s%d\n%s%d\n",
           "The string \"2593\" converted to int is ", i,
           "The converted value minus 593 is ", i - 593 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string "2593" converted to int is 2593
The converted value minus 593 is 2000
```

```c
/* Fig. 8.8: fig08_08.c
   Using atol */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   long l; /* variable to hold converted string */

   l = atol( "1000000" );

   printf( "%s%ld\n%s%ld\n",
           "The string \"1000000\" converted to long int is ", l,
           "The converted value divided by 2 is ", l / 2 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string "1000000" converted to long int is 1000000
The converted value divided by 2 is 500000
```

```c
/* Fig. 8.9: fig08_09.c
   Using strtod */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   /* initialize string pointer */
   const char *string = "51.2% are admitted";

   double d;        /* variable to hold converted sequence */
   char *stringPtr; /* create char pointer */

   d = strtod( string, &stringPtr );

   printf( "The string \"%s\" is converted to the\n", string );
   printf( "double value %.2f and the string \"%s\"\n", d, stringPtr );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string "51.2% are admitted" is converted to the
double value 51.20 and the string "% are admitted"
```

```c
/* Fig. 8.10: fig08_10.c
   Using strtol */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   const char *string = "-1234567abc"; /* initialize string pointer */

   char *remainderPtr; /* create char pointer */
   long x;             /* variable to hold converted sequence */

   x = strtol( string, &remainderPtr, 0 );

   printf( "%s\"%s\"\n%s%ld\n%s\"%s\"\n%s%ld\n",
           "The original string is ", string,
           "The converted value is ", x,
           "The remainder of the original string is ",
           remainderPtr,
           "The converted value plus 567 is ", x + 567 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The original string is "-1234567abc"
The converted value is -1234567
The remainder of the original string is "abc"
The converted value plus 567 is -1234000
```

```c
/* Fig. 8.11: fig08_11.c
   Using strtoul */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   const char *string = "1234567abc"; /* initialize string pointer */
   unsigned long x;    /* variable to hold converted sequence */
    char *remainderPtr; /* create char pointer */

   x = strtoul( string, &remainderPtr, 0 );

   printf( "%s\"%s\"\n%s%lu\n%s\"%s\"\n%s%lu\n",
           "The original string is ", string,
           "The converted value is ", x,
           "The remainder of the original string is ",
           remainderPtr,
           "The converted value minus 567 is ", x - 567 );

   return 0; //successful termination

} /* end main */
```

```
The original string is "1234567abc"
The converted value is 1234567
The remainder of the original string is "abc"
The converted value minus 567 is 1234000
```

Escola de Engenharia
Universidade do Minho

# 8.5 Standard Input/Output Library Functions

- **Functions in `<stdio.h>`**
- **Used to manipulate character and string data**

| Function prototype | Function description |
|---|---|
| `int getchar( void );` | Inputs the next character from the standard input and returns it as an integer. |
| `char *gets( char *s );` | Inputs characters from the standard input into the array `s` until a newline or end-of-file character is encountered. A terminating null character is appended to the array. |
| `int putchar( int c );` | Prints the character stored in `c`. |
| `int puts( const char *s );` | Prints the string `s` followed by a newline character. |
| `int sprintf( char *s, const char *format, ... );` | Equivalent to `printf`, except the output is stored in the array `s` instead of printing it on the screen. |
| `int sscanf( char *s, const char *format, ... );` | Equivalent to `scanf`, except the input is read from the array `s` instead of reading it from the keyboard. |

```c
/* Fig. 8.13: fig08_13.c
   Using gets and putchar */
#include <stdio.h>

int main()
{
   char sentence[ 80 ]; /* create char array */

   void reverse( const char * const sPtr ); /* prototype */

   printf( "Enter a line of text:\n" );

   /* use gets to read line of text */
   gets( sentence );

   printf( "\nThe line printed backwards is:\n" );
   reverse( sentence );

   return 0; /* indicates successful termination */

} /* end main */

```

```
23 /* recursively outputs characters in string in reverse order */
24 void reverse( const char * const sPtr )
25 {
26    /* if end of the string */
27    if ( sPtr[ 0 ] == '\0' ) {
28       return;
29    } /* end if */
30    else { /* if not end of the string */
31       reverse( &sPtr[ 1 ] );
32
33       putchar( sPtr[ 0 ] ); /* use putchar to display character */
34    } /* end else */
35
36 } /* end function reverse */
```

```
Enter a line of text:
Characters and Strings

The line printed backwards is:
sgnirtS dna sretcarahC
```

```
Enter a line of text:
able was I ere I saw elba

The line printed backwards is:
able was I ere I saw elba
```

```c
/* Fig. 8.14: fig08_14.c
   Using getchar and puts */
#include <stdio.h>

int main()
{
   char c;              /* variable to hold character input by user */
   char sentence[ 80 ]; /* create char array */
   int i = 0;           /* initialize counter i */

   /* prompt user to enter line of text */
   puts( "Enter a line of text:" );

   /* use getchar to read each character */
   while ( ( c = getchar() ) != '\n') {
      sentence[ i++ ] = c;
   } /* end while */

   sentence[ i ] = '\0';

   /* use puts to display sentence */
   puts( "\nThe line entered was:" );
   puts( sentence );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a line of text:
This is a test.

The line entered was:
This is a test.
```

```c
/* Fig. 8.15: fig08_15.c
   Using sprintf */
#include <stdio.h>

int main()
{
   char s[ 80 ]; /* create char array */
   int x;        /* define x */
   double y;     /* define y */

   printf( "Enter an integer and a double:\n" );
   scanf( "%d%lf", &x, &y );

   sprintf( s, "integer:%6d\ndouble:%8.2f", x, y );

   printf( "%s\n%s\n",
           "The formatted output stored in array s is:", s );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter an integer and a double:
298 87.375
The formatted output stored in array s is:
integer:   298
double:   87.38
```

```c
/* Fig. 8.16: fig08_16.c
   Using sscanf */
#include <stdio.h>

int main()
{
   char s[] = "31298 87.375"; /* initialize array s */
   int x;                     /* define x */
   double y;                  /* define y */

   sscanf( s, "%d%lf", &x, &y );

   printf( "%s\n%s%6d\n%s%8.3f\n",
           "The values stored in character array s are:",
           "integer:", x, "double:", y );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The values stored in character array s are:
integer:  31298
double:   87.375
```

# 8.6 String Manipulation Functions of the String Handling Library

- **String handling library has functions to**
  - Manipulate string data
  - Search strings
  - Tokenize strings
  - Determine string length

| Function prototype | Function description |
|---|---|
| `char *strcpy( char *s1, const char *s2 )` | Copies string `s2` into array `s1`. The value of `s1` is returned. |
| `char *strncpy( char *s1, const char *s2, size_t n )` | Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned. |
| `char *strcat( char *s1, const char *s2 )` | Appends string `s2` to array `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, size_t n )` | Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned. |

```c
/* Fig. 8.18: fig08_18.c
   Using strcpy and strncpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char x[] = "Happy Birthday to You"; /* initialize char array x */
   char y[ 25 ];                       /* create char array y */
    char z[ 15 ];                       /* create char array z */

   /* copy contents of x into y */
   printf( "%s%s\n%s%s\n",
           "The string in array x is: ", x,
           "The string in array y is: ", strcpy( y, x ) );

   /* copy first 14 characters of x into z. Does not copy null
      character */
   strncpy( z, x, 14 );

   z[ 14 ] = '\0'; /* append '\0' to z's contents */
   printf( "The string in array z is: %s\n", z );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string in array x is:
Happy Birthday to You
The string in array y is:
Happy Birthday to You
The string in array z is:
Happy Birthday
```

```c
/* Fig. 8.19: fig08_19.c
   Using strcat and strncat */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 20 ] = "Happy "; // initialize char array s1
   char s2[] = "New Year ";  // initialize char array s2
   char s3[ 40 ] = "";       // initialize char array s3

   printf( "s1 = %s\ns2 = %s\n", s1, s2 );

   /* concatenate s2 to s1 */
   printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );

   /* concatenate first 6 characters of s1 to s3. Place '\0'
      after last character */
   printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );

   /* concatenate s1 to s3 */
   printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
s1 = Happy
s2 = New Year
strcat( s1, s2 ) =
Happy New Year
strncat( s3, s1, 6 ) =
Happy
strcat( s3, s1 ) =
Happy Happy New Year
```

# 8.7 Comparison Functions of the String Handling Library

- **Comparing strings**
  - Computer compares numeric ASCII codes of characters in string
  - Appendix D has a list of character codes

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

  - Compares string `s1` to `s2`
  - Returns a negative number if `s1 < s2`, zero if `s1 == s2` or a positive number if `s1 > s2`

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

  - Compares up to `n` characters of string `s1` to `s2`
  - Returns values as above

```c
/* Fig. 8.21: fig08_21.c
   Using strcmp and strncmp */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *s1 = "Happy New Year"; /* initialize char pointer */
   const char *s2 = "Happy New Year"; /* initialize char pointer */
   const char *s3 = "Happy Holidays"; /* initialize char pointer */

   printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
          "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
          "strcmp(s1, s2) = ", strcmp( s1, s2 ),
          "strcmp(s1, s3) = ", strcmp( s1, s3 ),
          "strcmp(s3, s1) = ", strcmp( s3, s1 ) );

   printf("%s%2d\n%s%2d\n%s%2d\n",
          "strncmp(s1, s3, 6) = ", strncmp( s1, s3, 6 ),
          "strncmp(s1, s3, 7) = ", strncmp( s1, s3, 7 ),
          "strncmp(s3, s1, 7) = ", strncmp( s3, s1, 7 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  1
strncmp(s3, s1, 7) = -1
```

# 8.8 Search Functions of the String Handling Library

| Function prototype | Function description |
|---|---|
| `char *strchr( const char *s, int c );` | Locates the first occurrence of character `c` in string `s`. If `c` is found, a pointer to `c` in `s` is returned. Otherwise, a `NULL` pointer is returned. |
| `size_t strcspn( const char *s1, const char *s2 );` | Determines and returns the length of the initial segment of string `s1` consisting of characters not contained in string `s2`. |
| `size_t strspn( const char *s1, const char *s2 );` | Determines and returns the length of the initial segment of string `s1` consisting only of characters contained in string `s2`. |
| `char *strpbrk( const char *s1, const char *s2 );` | Locates the first occurrence in string `s1` of any character in string `s2`. If a character from string `s2` is found, a pointer to the character in string `s1` is returned. Otherwise, a `NULL` pointer is returned. |
| `char *strrchr( const char *s, int c );` | Locates the last occurrence of `c` in string `s`. If `c` is found, a pointer to `c` in string `s` is returned. Otherwise, a `NULL` pointer is returned. |
| `char *strstr( const char *s1, const char *s2 );` | Locates the first occurrence in string `s1` of string `s2`. If the string is found, a pointer to the string in `s1` is returned. Otherwise, a `NULL` pointer is returned. |
| `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. |

```c
/* Fig. 8.23: fig08_23.c
   Using strchr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string = "This is a test"; /* initialize char pointer */
   char character1 = 'a';                 /* initialize character1 */
    char character2 = 'z';                 /* initialize character2 */

   /* if character1 was found in string */
   if ( strchr( string, character1 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
              character1, string );
   } /* end if */
   else { /* if character1 was not found */
      printf( "\'%c\' was not found in \"%s\".\n",
              character1, string );
   } /* end else */

```

```
22    /* if character2 was found in string */
23    if ( strchr( string, character2 ) != NULL ) {
24       printf( "\'%c\' was found in \"%s\".\n",
25               character2, string );
26    } /* end if */
27    else { /* if character2 was not found */
28       printf( "\'%c\' was not found in \"%s\".\n",
29              character2, string );
30    } /* end else */
31
32    return 0; /* indicates successful termination */
33
34 } /* end main */
```

```
'a' was found in "This is a test".
'z' was not found in "This is a test".
```

```c
/* Fig. 8.24: fig08_24.c
   Using strcspn */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize two char pointers */
   const char *string1 = "The value is 3.14159";
    const char *string2 = "1234567890";

    printf( "%s%s\n%s%s\n\n%s\n%s%u",
            "string1 = ", string1, "string2 = ", string2,
            "The length of the initial segment of string1",
            "containing no characters from string2 = ",
            strcspn( string1, string2 ) );

    return 0; /* indicates successful termination */

} /* end main */
```

```
string1 = The value is 3.14159
string2 = 1234567890

The length of the initial segment of string1
containing no characters from string2 = 13
```

```c
/* Fig. 8.25: fig08_25.c
   Using strpbrk */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string1 = "This is a test"; /* initialize char pointer */
   const char *string2 = "beware";         /* initialize char pointer */

   printf( "%s\"%s\"\n'%c'%s\n\"%s\"\n",
           "Of the characters in ", string2,
           *strpbrk( string1, string2 ),
           " is the first character to appear in ", string1 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Of the characters in "beware"
'a' is the first character to appear in
"This is a test"
```

```c
/* Fig. 8.26: fig08_26.c
   Using strrchr */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize char pointer */
   const char *string1 = "A zoo has many animals "
                         "including zebras";
    int c = 'z'; /* initialize c */

    printf( "%s\n%s'%c'%s\"%s\"\n",
            "The remainder of string1 beginning with the",
            "last occurrence of character ", c,
            " is: ", strrchr( string1, c ) );

    return 0; /* indicates successful termination */

} /* end main */
```

```
The remainder of string1 beginning with the
last occurrence of character 'z' is: "zebras"
```

```c
/* Fig. 8.27: fig08_27.c
   Using strspn */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize two char pointers */
   const char *string1 = "The value is 3.14159";
    const char *string2 = "aehi lsTuv";

    printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
            "string1 = ", string1, "string2 = ", string2,
            "The length of the initial segment of string1",
            "containing only characters from string2 = ",
            strspn( string1, string2 ) );

    return 0; /* indicates successful termination */

} /* end main */
```

```
string1 = The value is 3.14159
string2 = aehi lsTuv

The length of the initial segment of string1
containing only characters from string2 = 13
```

```c
/* Fig. 8.28: fig08_28.c
   Using strstr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string1 = "abcdefabcdef"; /* initialize char pointer */
   const char *string2 = "def";          /* initialize char pointer */

   printf( "%s%s\n%s%s\n\n%s\n%s%s\n",
           "string1 = ", string1, "string2 = ", string2,
           "The remainder of string1 beginning with the",
           "first occurrence of string2 is: ",
           strstr( string1, string2 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
string1 = abcdefabcdef
string2 = def

The remainder of string1 beginning with the
first occurrence of string2 is: defabcdef
```

```c
/* Fig. 8.29: fig08_29.c
   Using strtok */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize array string */
   char string[] = "This is a sentence with 7 tokens";
    char *tokenPtr; /* create char pointer */

   printf( "%s\n%s\n\n%s\n",
           "The string to be tokenized is:", string,
           "The tokens are:" );

   tokenPtr = strtok( string, " " ); /* begin tokenizing sentence */

   /* continue tokenizing sentence until tokenPtr becomes NULL */
   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, " " ); /* get next token */
   } /* end while */

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string to be tokenized is:
This is a sentence with 7 tokens

The tokens are:
This
is
a
sentence
with
7
tokens
```

Escola de Engenharia
Universidade do Minho

# 8.9 Memory Functions of the String-handling Library

- **Memory Functions**
  - In `<stdlib.h>`
  - Manipulate, compare, and search blocks of memory
  - Can manipulate any block of data
- **Pointer parameters are `void *`**
  - Any pointer can be assigned to `void *`, and vice versa
  - `void *` cannot be dereferenced
    - Each function receives a size argument specifying the number of bytes (characters) to process

Escola de Engenharia
Universidade do Minho

# 8.9 Memory Functions of the String-handling Library

| Function prototype | Function description |
|---|---|
| `void *memcpy( void *s1, const void *s2, size_t n );` | Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. A pointer to the resulting object is returned. |
| `void *memmove( void *s1, const void *s2, size_t n );` | Copies `n` characters from the object pointed to by `s2` into the object pointed to by `s1`. The copy is performed as if the characters were first copied from the object pointed to by `s2` into a temporary array and then from the temporary array into the object pointed to by `s1`. A pointer to the resulting object is returned. |
| `int memcmp( const void *s1, const void *s2, size_t n );` | Compares the first `n` characters of the objects pointed to by `s1` and `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`. |
| `void *memchr( const void *s, int c, size_t n );` | Locates the first occurrence of `c` (converted to `unsigned char`) in the first `n` characters of the object pointed to by `s`. If `c` is found, a pointer to `c` in the object is returned. Otherwise, `NULL` is returned. |
| `void *memset( void *s, int c, size_t n );` | Copies `c` (converted to `unsigned char`) into the first `n` characters of the object pointed to by `s`. A pointer to the result is returned. |

```c
/* Fig. 8.31: fig08_31.c
   Using memcpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 17 ];                    /* create char array s1 */
   char s2[]  = "Copy this string"; /* initialize char array s2 */

   memcpy( s1, s2, 17 );
   printf( "%s\n%s\"%s\"\n",
           "After s2 is copied into s1 with memcpy,",
           "s1 contains ", s1 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
After s2 is copied into s1 with memcpy,
s1 contains "Copy this string"
```

```c
/* Fig. 8.32: fig08_32.c
   Using memmove */
#include <stdio.h>
#include <string.h>

int main()
{
   char x[] = "Home Sweet Home"; /* initialize char array x */

   printf( "%s%s\n", "The string in array x before memmove is: ", x );
   printf( "%s%s\n", "The string in array x after memmove is: ",
           memmove( x, &x[ 5 ], 10 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The string in array x before memmove is: Home Sweet Home
The string in array x after memmove is: Sweet Home Home
```

```c
/* Fig. 8.33: fig08_33.c
   Using memcmp */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[] = "ABCDEFG"; /* initialize char array s1 */
   char s2[] = "ABCDXYZ"; /* initialize char array s2 */

   printf( "%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n",
           "s1 = ", s1, "s2 = ", s2,
           "memcmp( s1, s2, 4 ) = ", memcmp( s1, s2, 4 ),
           "memcmp( s1, s2, 7 ) = ", memcmp( s1, s2, 7 ),
           "memcmp( s2, s1, 7 ) = ", memcmp( s2, s1, 7 ) );

   return 0; /* indicate successful termination */

} /* end main */
```

```
s1 = ABCDEFG
s2 = ABCDXYZ

memcmp( s1, s2, 4 ) =  0
memcmp( s1, s2, 7 ) = -1
memcmp( s2, s1, 7 ) =  1
```

```c
/* Fig. 8.34: fig08_34.c
   Using memchr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *s = "This is a string"; /* initialize char pointer */

   printf( "%s\'%c\'%s\"%s\"\n",
           "The remainder of s after character ", 'r',
           " is found is ", memchr( s, 'r', 16 ) );

   return 0; /* indicates successful termination */

}
```

```
The remainder of s after character 'r' is found is "ring"
```

Escola de Engenharia
Universidade do Minho

# 8.10 Other Functions of the String Handling Library

- `char *strerror( int errornum );`
  - Creates a system-dependent error message based on `errornum`
  - Returns a pointer to the string
- `size_t strlen( const char *s );`
  - Returns the number of characters (before `NULL`) in string `s`

```c
/* Fig. 8.35: fig08_35.c
   Using memset */
#include <stdio.h>
#include <string.h>

int main()
{
   char string1[ 15 ] = "BBBBBBBBBBBBBB"; /* initialize string1 */

   printf( "string1 = %s\n", string1 );
   printf( "string1 after memset = %s\n", memset( string1, 'b', 7 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
string1 = BBBBBBBBBBBBBB
string1 after memset = bbbbbbbBBBBBBB
```

```c
/* Fig. 8.37: fig08_37.c
   Using strerror */
#include <stdio.h>
#include <string.h>

int main()
{
   printf( "%s\n", strerror( 2 ) );

    return 0; /* indicates successful termination */

} /* end main */
```

```
No such file or directory
```

```c
/* Fig. 8.38: fig08_38.c
   Using strlen */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize 3 char pointers */
   const char *string1 = "abcdefghijklmnopqrstuvwxyz";
    const char *string2 = "four";
    const char *string3 = "Boston";

    printf("%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n",
       "The length of ", string1, " is ",
       ( unsigned long ) strlen( string1 ),
       "The length of ", string2, " is ",
       ( unsigned long ) strlen( string2 ),
       "The length of ", string3, " is ",
       ( unsigned long ) strlen( string3 ) );

    return 0; /* indicates successful termination */

} /* end main */
```

```
The length of "abcdefghijklmnopqrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
```

Escola de Engenharia
Universidade do Minho

# Questões?

# Programação – Aula Teórica 8

## Caracteres e Strings


**Luís Paulo Reis**

lpreis@dsi.uminho.pt

**Professor Associado do Departamento de Sistemas de Informação, Escola de Engenharia, Universidade do Minho, Portugal**


**(Slides Baseados em Deitel e Deitel 2010 e L.P.Reis et al., 2006)**