PEN203

Characters and Strings

C++ How to Program
Deitel & Deitel

Outline

- Fundamentals of Strings and Characters
- Character-Handling Library
- String-Conversion Functions
- Standard Input/Output Library Functions
- String Manipulation Functions
- String Comparison Functions

Fundamentals of Strings and Characters

- Characters
  - Character constant is an int value represented as a character
- Strings
  - A series of characters considered as a single unit
  - String literal is written in double quotes

    "Hello"
  - Basically strings are arrays of characters
    - The actual value of string is the address of first character

Fundamentals of Strings and Characters

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

    ```
    char color[] = "blue";
    char *colorPtr = "blue";
    ```
  - Strings represented as character arrays end with '\0'
  - color variable has 4+1=5 elements
  - To input strings using scanf:
    - cin>>word;

Character-Handling Library (ctype.h)

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

int main( void )
{
   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" );

```

Character-Handling Library

```
   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" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
       "According to isxdigit:",
       isxdigit( 'F' ) ? "F is a " : "F is not a ",
       "hexadecimal digit",
       isxdigit( 'J' ) ? "J is a " : "J is not a ",
       "hexadecimal digit",
       isxdigit( '7' ) ? "7 is a " : "7 is not a ",
       "hexadecimal digit",
       isxdigit( '$' ) ? "$ is a " : "$ is not a ",
```

Character-Handling Library

```c
        "hexadecimal digit",
        isxdigit( 'f' ) ? "f is a " : "f is not a ",
        "hexadecimal digit" );

    return 0; /* indicates successful termination */

} /* 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
```

String-Conversion Functions (stdlib.h)

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`.

Standard Input/Output Library Functions (stdio.h)

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. Returns the string inputted into `s`. Note that an error will occur if `s` is not large enough to hold the string.
`int putchar( int c );`	Prints the character stored in `c` and returns it as an integer.
`int puts( const char *s );`	Prints the string `s` followed by a newline character. Returns a non-zero integer if successful, or `EOF` if an error occurs.
`int sprintf( char *s, const char *format, ... );`	Equivalent to `printf`, except the output is stored in the array `s` instead of printed on the screen. Returns the number of characters written to `s`, or `EOF` if an error occurs.
`int sscanf( char *s, const char *format, ... );`	Equivalent to `scanf`, except the input is read from the array `s` rather than from the keyboard. Returns the number of items successfully read by the function, or `EOF` if an error occurs.

String Manipulation Functions (string.h)

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.

String Comparison Functions (string.h)

Function prototype	Function description
`int strcmp( const char *s1, const char *s2 );`	Compares the string `s1` with 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 strncmp( const char *s1, const char *s2, size_t n );`	Compares up to `n` characters of the string `s1` with 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.

String Comparison Functions (string.h)

```
// Fig. 5.30: fig05_30.cpp
// Using strcmp and strncmp.
#include <iostream>

using std::cout;
using std::endl;

#include <iomanip>

using std::setw;

#include <cstring>  // prototypes for strcmp and strncmp

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

  cout << "s1 = " << s1 << "\ns2 = " << s2
      << "\ns3 = " << s3 << "\n\nstrcmp(s1, s2) = "
      << setw( 2 ) << strcmp( s1, s2 )
      << "\nstrcmp(s1, s3) = " << setw( 2 )
      << strcmp( s1, s3 ) << "\nstrcmp(s3, s1) = "
      << setw( 2 ) << strcmp( s3, s1 );
```

String Comparison Functions (string.h)

```
   cout << "\n\nstrncmp(s1, s3, 6) = " << setw( 2 )
      << strncmp( s1, s3, 6 ) << "\nstrncmp(s1, s3, 7) = "
      << setw( 2 ) << strncmp( s1, s3, 7 )
      << "\nstrncmp(s3, s1, 7) = "
      << setw( 2 ) << strncmp( s3, s1, 7 ) << endl;

   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
```