Capitolo 8 - Caratteri e Stringhe

8.2 Concetti fondamentali di stringhe e caratteri

- Caratteri
  - Costanti di Carattere
    - Un valore `int` rappresentato come un carattere fra apici
    - `'z'` rappresenta il valore intero di `z`

Esercizio: Scrivere un programma che chieda all'utente di inserire un carattere e visualizzi il codice ASCII del carattere inserito.

Esercizio: Realizzare un programma che legge in ingresso un intero e, se il valore inserito appartiene all'insieme dei codici ASCII dei caratteri stampabili, stampa il carattere corrispondente. (Si supponga di non conoscere il codice ASCII dei caratteri, ma di sapere solo che i codici dei caratteri stampabili appartengono all'intervallo che va dal codice del carattere ' ' (spazio) al codice del carattere '~' esclusi.

8.2 Concetti fondamentali di stringhe e caratteri

- Stringhe
  - Serie di caratteri trattati come singola unità
    - Possono includere lettere, cifre e caratteri speciali (`*, /, $`)
  - Le stringhe letterali (o constanti stringa) – scritte fra virgolette
    - `"Hello"`
  - Le stringhe sono array di caratteri
    - La stringa è un puntatore al primo carattere
    - Il valore di una stringa è l'indirizzo del primo carattere

8.2 Concetti fondamentali di stringhe e caratteri

- Definizioni di stringhe
  - Definita come un array di caratteri o come una variabile di tipo `char *`

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

  - Ricordare che le stringhe rappresentate come array di caratteri terminano con `'\0'`
    - `color` ha `5 elementi`
- Acquisizione di stringhe
  - Utilizzo di `scanf`

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

    - Copia l'input in `word[]`
    - Non è necessario il `&` (poiché una stringa è già un puntatore)
  - Ricordare di lasciare spazio nell'array per `'\0'`

8.3 Libreria per la manipolazione di caratteri

- Libreria per la manipolazine di caratteri
  - Contiene le funzioni per effettuare utili test e manipolazioni su caratteri
  - Ogni funzione riceve un carattere (un `int`) o `EOF` come argomento
- #include `<ctype.h>`

8.3 Libreria per la manipolazione di caratteri

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.

8.5 Funzioni della libreria Input/Output

- Usate per manipolare caratteri e stringhe
- #include `<stdio.h>`

Function prototype	Function description
`int getchar( void );`	Inputs the next character from the standard input and re turns 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.

```
1  /* Fig. 8.13: fig08_13.c
2     Using gets and putchar */
3  #include <stdio.h>
4
5  int main()
6  {
7     char sentence[ 80 ]; /* create char array */
8
9     void reverse( const char * const sPtr ); /* prototype */
10
11    printf( "Enter a line of text:\n" );
12
13    /* use gets to read line of text */
14    gets( sentence );
15
16    printf( "\nThe line printed backwards is:\n");
17    reverse( sentence );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
22
```

fig08_13.c (Part 1 of 2)

```
/* recursively outputs characters in string in reverse order */
void reverse( const char * const sPtr )
{
   /* if end of the string */
   if ( sPtr[ 0 ] == '\0' ) {
      return;
   } /* end if */
   else { /* if not end of the string */
      reverse( &sPtr[ 1 ] );

      putchar( sPtr[ 0 ] ); /* use putchar to display character */
   } /* end else */

} /* end function reverse */
```

fig08_13.c (Part 1 of 2)

```
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
```

Program Output

```
/* 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 */
```

fig8_14.c

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

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

Program Output

8.6 Funzioni per la manipolazione di stringhe

- La libreria delle funzioni per la manipolazione di stringhe contiene funzioni per
  - Manipolare stringhe
  - Ricercare stringhe
  - Tokenizzare stringhe
  - Determinare la lunghezza di una stringa
- #include `<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.

fig08_18.c

```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 */
```

Program Output

```
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
```

fig08_19.c

```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
```

Program Output

8.7 Funzioni di confronto fra stringhe

- Confrontare le stringhe
  - Il computer confronta i codici numerici ASCII dei caratteri delle stringhe

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

  - Confronta la stringa `s1` con `s2`
  - Restituisce un numero negativo se `s1 < s2`, zero se `s1 == s2` o un numero positivo se `s1 > s2`

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

  - Confronta `n` caratteri della stringa `s1` con `s2`
  - Restituisce gli stessi valori come sopra

```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 */
```

fig08_21.c

```
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
```

Program Output

8.8 Funzioni di ricerca di stringhe

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.