Cap. 8 - Caracteres and Cadenas

Esquema

8.1 Introducción

- Introducir algunas funciones estándar de la biblioteca
  - Fácil procesamiento de cadenas y caracteres
  - Los programas pueden procesar caracteres, cadenas, líneas de texto y bloques de memoria
- Estas técnicas utilizadas para hacer
  - Los procesadores de texto
  - Software de diseño de páginas
  - Programas de composición tipográfica

8.2 Fundamentos de las cadenas y los caracteres

- Caracteres
  - Construir bloques de programas
    - Cada programa es una secuencia de caracteres agrupados con sentido
  - Caracteres constantes - un valor **`int`** representado como un carácter en simple comillas
    - **`'z'`** representa el valor entero de de **`z`**

8.2 Fundamentos de las cadenas y los caracteres (II)

- Cadenas
  - Es una serie de caracteres tratados como una simple unidad
    - Puede incluir letras, dígitos, y ciertos caracteres especiales (`*`, `/`, `$`)
  - Literal de Cadena (cadena constante) - escrito en doble comilla
    - `"Hello"`
  - Las cadenas son arreglos de caracteres
    - La cadena apunta al primer carácter
    - Valor de la cadena es la dirección del prime caracter

8.2 Fundamentos de las cadenas y los caracteres (III)

- Declaraciones de cadena
  - Declarado como un arreglo de cadenas o una variable de tipo `char *`

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

  - Recuerde que las cadenas son representados como arreglos de caracteres que terminan con `'\0'`
    - `color` tiene 5 elementos

8.2 Fundamentos de las cadenas y los caracteres (IV)

- Ingresando cadenas
  - Use **`scanf`**

    `scanf("%s", word);`
    - Copia la entrada en **`word[]`**, el cual no necesita **`&`** (porque es una cadena a puntero)
  - Recuerde dejar espacio para **`'\0'`**

8.3 Librería de Manejo de Caracteres

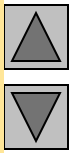
- Librería de Manejo de Caracteres
  - Incluye funciones para realizar pruebas y manipulaciones útiles de los datos de los personajes
  - Cada función recibe un carácter (un int) o EOF como argumento

8.3 Librería de Manejo de Caracteres (II)

- In **<ctype.h>**

Prototipo	Descripción de la función
`int isdigit( int c );`	Devuelve un valor verdadero si `c` es un dígito; de lo contrario devuelve 0 (falso).
`int isalpha( int c );`	Devuelve un valor verdadero si `c` es una letra; de lo contrario devuelve 0 (falso).
`int isalnum( int c );`	Devuelve un valor verdadero si `c` es un dígito o una letra; de lo contrario devuelve 0 (falso).
`int isxdigit( int c );`	Devuelve un valor verdadero si `c` es un dígito hexadecimal; de lo contrario devuelve 0 (falso). (Revise el apéndice E, Sistemas de numeración, para una explicación detallada acerca de los números binarios, números octales, números decimales y números hexadecimales.)
`int islower( int c );`	Devuelve un valor verdadero si `c` es una letra minúscula; de lo contrario devuelve 0 (falso).
`int isupper( int c );`	Devuelve un valor verdadero si `c` es una letra mayúscula; de lo contrario devuelve 0 (falso).
`int tolower( int c );`	Si `c` es una letra mayúscula, `tolower` devuelve `c` como una letra minúscula. De lo contrario, `tolower` devuelve el argumento sin modificación.
`int toupper( int c );`	Si `c` es una letra minúscula, `toupper` devuelve `c` como una letra mayúscula. De lo contrario, toupper devuelve el argumento sin modificación.
`int isspace( int c );`	Devuelve un valor verdadero si `c` es un carácter de espacio en blanco (nueva línea (`'\n'`), espacio (`' '`), avance de página (`'\f'`), retorno de carro (`'\r'`), tabulador horizontal (`'\t'`) o tabulador vertical (`'\v'`); de lo contrario devuelve 0.
`int iscntrl( int c );`	Devuelve un valor verdadero si `c` es un carácter de control; de lo contrario devuelve 0 (falso).
`int ispunct( int c );`	Devuelve un valor verdadero si `c` es un carácter de impresión diferente de un espacio, un dígito o una letra; de lo contrario devuelve 0.
`int isprint( int c );`	Devuelve un valor verdadero si `c` es un carácter de impresión, incluso el espacio (`' '`); de lo contrario devuelve 0.
`int isgraph( int c );`	Devuelve un valor verdadero si `c` es un carácter de impresión diferente del espacio (`' '`); de lo contrario devuelve 0.

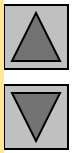
```c
/* Fig. 8.2: fig08_02.c
   Using functions isdigit, isalpha, isalnum, and isxdigit */
#include <stdio.h>
#include <ctype.h>

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" );
   printf( "%s\n%s%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",
```



Outline

1. Cargar cabecera
2. Realiza Testeo
3. Imprime

```
33          isxdigit( '7' ) ? "7 is a " : "7 is not a ",
34          "hexadecimal digit",
35          isxdigit( '$' ) ? "$ is a " : "$ is not a ",
36          "hexadecimal digit",
37          isxdigit( 'f' ) ? "f is a " : "f is not a ",
38          "hexadecimal digit" );
39       return 0;
40    }
```



Outline

Salida de Programa

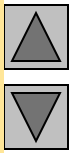
```
According to isdigit:
8 is a digit
# is not a digit

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
```

8.4 Función de Conversión de Cadena

- Funciones de Conversiones
  - En **<stdlib.h>** (librería de utilidades general)
  - Convierte cadenas de dígitos a números enteros y punto flotante

Prototipo de la función	Descripción de la función
`double atof( const char *ptrN );`	Convierte la cadena `ptrN` a `double`.
`int atoi( const char *ptrN );`	Convierte la cadena `ptrN` a `int`.
`long atol( const char *ptrN );`	Convierte la cadena `ptrN` a `long int`.
`double strtod( const char *ptrN, char **ptrFinal );`	Convierte la cadena `ptrN` a `double`.
`long strtol( const char *ptrN, char **ptrFinal, int base );`	Convierte la cadena `ptrN` a `long`.
`unsigned long strtoul( const char *ptrN, char **ptrFinal, int base );`	Convierte la cadena `ptrN` a `unsigned long`.

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

int main()
{
   double d;

   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;
}
```



Outline

1. Inicializa variable

2. Convierte cadenas
2.1 Asigna a variable

3. Imprime

Salida

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

8.5 Funciones de Librería Estándar Input/Output

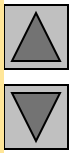
- Funciones en **<stdio.h>**

Prototipo de la función	Descripción de la función
`int getchar( void );`	Lee el siguiente carácter de la entrada estándar y lo devuelve como un entero.
`char *gets( char *s ) ;`	Lee el siguiente carácter de la entrada estándar y lo coloca en el arreglo `s` hasta que encuentra un carácter de nueva línea o de fin de archivo. Agrega un carácter de terminación nulo al arreglo.
`int putchar( int c );`	Imprime el carácter almacenado en `c`.
`int puts( const char *s ) ;`	Imprime la cadena `s` seguida por el carácter de nueva línea.
`int sprintf( char *s, const char *formato, ... );`	Equivalente a `printf`, excepto que la salida se almacena en el arreglo `s`, en lugar de imprimirse en la pantalla.
`int sscanf( char *s, const char *formato, ... );`	Equivalente a `scanf`, excepto que la entrada se lee desde el arreglo `s`, en lugar leerlo desde el teclado.

```c
/* Fig. 8.13: fig08 13.c
   Using gets and putchar */
#include <stdio.h>

int main()
{
   char sentence[ 80 ];
   void reverse( const char * const );

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

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

   return 0;
}

void reverse( const char * const sPtr )
{
   if ( sPtr[ 0 ] == '\0' )
      return;
   else {
      reverse( &sPtr[ 1 ] );
      putchar( sPtr[ 0 ] );
   }
}
```

reverse calls itself using substrings of the original string. When it reaches the **'\0'** character it prints using **putchar**

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

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



Outline

1. Inicializa variables

2. Entrada

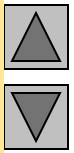
3. Imprime

3.1 Definición de Función (note recursion)

8.6 Funciones de Manipulación de Cadenas de la Librería de Manejo de Cadenas

- Librería de Manejo de Cadenas tiene funciones para
  - Manipular cadena de datos
  - Buscar cadenas
  - Señalar las cadenas
  - Determinar la longitud de la cadena

Prototipo de la función	Descripción de la función
`char *strcpy( char *s1, const char *s2 )`	Copia la cadena `s2` dentro del arreglo `s1`. Devuelve el valor de `s1`.
`char *strncpy( char *s1, const char *s2, size_t n )`	Copia al menos `n` caracteres de la cadena `s2` dentro del arreglo `s1`. Devuelve el valor de `s1`.
`char *strcat( char *s1, const char *s2 )`	Agrega la cadena `s2` al arreglo `s1`. El primer carácter de `s2` sobrescribe al carácter de terminación nulo de `s1`. Devuelve el valor de `s1`.
`char *strncat( char *s1, const char *s2, size_t n )`	Agrega al menos `n` caracteres de la cadena `s2` al arreglo `s1`. El primer carácter de `s2` sobrescribe al carácter de terminación nulo de `s1`. Devuelve el valor de `s1`.



Outline

```
1   /* Fig. 8.19: fig08_19.c
2      Using strcat and strncat */
3   #include <stdio.h>
4   #include <string.h>
5
6   int main()
7   {
8      char s1[ 20 ] = "Happy ";
9      char s2[] = "New Year ";
10     char s3[ 40 ] = "";
11
12     printf( "s1 = %s\ns2 = %s\n", s1, s2 );
13     printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
14     printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 )
`.
15     printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
16     return 0;
17  }
```

1. Inicializar variables

2. Funciones llamando

3. Imprimir

Salida

```
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 Función comparación de Librería de Manejo de Cadenas

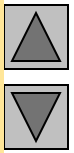
- Comparación de cadenas
  - La computadora compara los códigos numéricos ASCII de los caracteres en cadena

- `int strcmp( const char *s1, const char *s2 );`
  - Compara cadena **`s1`** a **`s2`**
  - Retorna un número negativo (**`s1 < s2`**), cero (**`s1 == s2`**), o número positivo (**`s1 > s2`**)

- `int strncmp( const char *s1, const char *s2, size_t n );`
  - Compara hasta el caracter n de la cadena **`s1`** a **`s2`**
  - Retorna valores como los anteriores

8.8 Función de Búsqueda de la librería de manejo de Cadenas

Prototipo de función	Descripción de la función
`char *strchr( const char *s, int c );`	Localiza la primera ocurrencia del carácter `c` en la cadena `s`. Si se localiza a `c`, se devuelve un apuntador a `c` en `s`. De lo contrario, se devuelve un apuntador NULL.
`size_t strcspn( const char *s1, const char *s2 );`	Determina y devuelve la longitud del segmento inicial de la cadena `s1`, que consiste en los caracteres no contenidos en la cadena `s2`.
`size_t strspn( const char *s1, const char *s2 );`	Determina y devuelve la longitud del segmento inicial de la cadena `s1`, que consiste sólo en los caracteres contenidos en la cadena `s2`.
`char *strpbrk( const char *s1, const char *s2 );`	Localiza la primera ocurrencia en la cadena `s1` de cualquier carácter de la cadena `s2`. Si se localiza un carácter de la cadena `s2`, se devuelve un apuntador al carácter de la cadena `s1`. De lo contrario, se devuelve un apuntador NULL.
`char *strrchr( const char *s, int c );`	Localiza la última ocurrencia de `c` en la cadena `s`. Si se localiza a `c`, se devuelve un apuntador a `c` en la cadena `s`. De lo contrario, se devuelve un apuntador NULL.
`char *strstr( const char *s1, const char *s2 );`	Localiza la primera ocurrencia en la cadena `s1` de la cadena `s2`. Si se localiza la cadena, se devuelve un apuntador a la cadena en `s1`. De lo contrario, se devuelve un apuntador NULL.
`char *strtok( char *s1, const char *s2 );`	Una secuencia de llamadas a `strtok` separa la cadena `s1` en "tokens" (piezas lógicas como palabras de una línea de texto) separados por caracteres contenidos en la cadena `s2`. La primera llamada contiene `s1` como el primer argumento, y las llamadas subsiguientes contienen a NULL como el primer argumento, para continuar separando la misma cadena. Un apuntador al token actual es devuelto por cada llamada. Si no hay más tokens cuando se llama a la función, se devuelve NULL.



Outline

1. Inicializar variables

2. Función llama

3. Imprime

```
1   /* Fig. 8.27: fig08_27.c
2      Using strspn */
3   #include <stdio.h>
4   #include <string.h>
5
6   int main()
7   {
8      const char *string1 = "The value is 3.14159";
9      const char *string2 = "aehi lsTuv";
10
11     printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
12             "string1 = ", string1, "string2 = ", string2,
13             "The length of the initial segment of string1",
14             "containing only characters from string2 = ",
15             strspn( string1, string2 ) );
16     return 0;
17  }
```

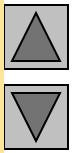
Salida

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

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

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

   tokenPtr = strtok( string, " " );

   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, " " );
   }

   return 0;
}
```

Salida

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

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



Outline

1. Inicializa variables

2. Función llama

3. Imprime

8.9 Función de Memoria de la Librería de Manejo de Cadenas

- Función Memoria
  - En **<stdlib.h>**
  - Manipula, compara, y busca bloque de memoria
  - Puede manipular cualquier bloque de datos
- Puntero parametrizado son **void ***
  - Cualquier puntero puede ser asignado a **void ***, y vice versa
  - **void *** no puede ser desreferenciado
    - Cada función recibe un tamaño de argumento especificando el número de bytes (caracteres) a pocesar

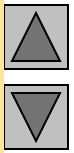
8.9 Función de Memoria de la Librería de Manejo de Cadenas(II)

"Object" se refiere a un bloque de datos

Prototipo de función	Descripción de la función
`void *memcpy( void *s1, const void *s2, size_t n );`	Copia `n` caracteres desde el objeto al que apunta `s2`, dentro del objeto al que apunta `s1`. Devuelve un apuntador al objeto resultante.
`void *memmove( void *s1, const void *s2, size_t n );`	Copia `n` caracteres desde el objeto al que apunta `s2` dentro del objeto al que apunta `s1`. La copia se lleva a cabo como si los caracteres primero se copiaran desde el objeto al que apunta `s2` en un arreglo temporal y después desde el arreglo temporal hacia el objeto al que apunta `s1`. Devuelve un apuntador al objeto resultante.
`void *memcmp( const void *s1, const void *s2, size_t n );`	Compara los primeros `n` caracteres de los objetos a los que apuntan `s1` y `s2`. La función devuelve un numero igual, menor o mayor que `0` si `s1` es igual, menor o mayor que `s2`.
`void *memchr( const void *s, int c, size_t n );`	Localiza la primera ocurrencia de `c` (convertida a `unsigned char`) en los primeros `n` caracteres del objeto al que apunta `s`. Si se encuentra `c`, devuelve un apuntador a `c`. De lo contrario, devuelve `NULL`.
`void *memset( void *s, int c, size_t n );`	Copia `c` (convertido a `unsigned char`) dentro de los primeros `n` caracteres del objeto al que apunta `s`. Devuelve un apuntador al resultado.

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

int main()
{
   char x[] = "Home Sweet Home";

   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;
}
```

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



Outline

1. Inicializar variables

2. Función llama

3. Imprime

Salids

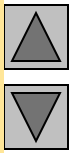
8.10 Otras funciones de la librería de manejo de cadenas

- **`char *strerror( int errornum );`**
  - Crea un mensaje de error dependiente del sistema basado en **`errornum`**
  - Devuelve un puntero a la cadena
  -

- **`size_t strlen( const char *s );`**
  - Devuelve el número de caracteres (antes de NULL) en la cadena **`s`**

```
1    /* Fig. 8.37: fig08_37.c
2       Using strerror */
3    #include <stdio.h>
4    #include <string.h>
5
6    int main()
7    {
8       printf( "%s\n", strerror( 2 ) );
9       return 0;
10   }
```

```
No such file or directory
```



Outline

1. Función llamada

2. Imprime

Salida