7. Cadenas

Fundamentos de Informática

Especialidad de Electrónica – 2013-2014

Ismael Etxeberria Agiriano

Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Índice

7. Cadenas

1. Ejemplos de literales
2. Punteros y cadenas
3. Operaciones con cadenas

1. Ejemplos de cadenas literales

```
char asig[]="Informatica I";
```

Terminador de la cadena

0	1	2	3	4	5	6	7	8	9	10	11	12	13
'I'	'n'	'f'	'o'	'r'	'm'	'a'	't'	'i'	'c'	'a'	' '	'I'	'\0'
73	110	102	111	114	109	97	116	105	99	97	32	73	0

```
char equiv[]="Saludo = \"HOLA\""
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
'S'	'a'	'l'	'u'	'd'	'o'	' '	'='	' '	'\"'	'H'	'O'	'L'	'A'	'\"'	'\0'

- ¿Cuántos caracteres ocupa cada una de estas cadenas?
- ¿Cómo copiar estas cadenas?

Punteros y cadenas: tamaño

```
#include <stdio.h>
void main (void)
{
  int    n1, n2, n3, *p1i, *p2i;
  double f1, f2, f3, *p1f, *p2f;
  char   c1, k0 [3], k1[]= "01", *k2="01234", *k3;
  printf ("n1 (n2, n3): %d bytes\n", sizeof (n1));
  printf ("p1i (p2i):   %d bytes\n", sizeof (p1i));
  printf ("*p1i (*p2i): %d bytes\n", sizeof (*p1i));
  printf ("f1 (f2, f3): %d bytes\n", sizeof (f1));
  printf ("p1f (p2f):   %d bytes\n", sizeof (p1f));
  printf ("*p1f (*p2f): %d bytes\n", sizeof (*p1f));
  printf ("c1:          %d bytes\n", sizeof (c1));
  printf ("k0 (k1):     %d bytes\n", sizeof (k1));
  printf ("k2 (k3):     %d bytes\n", sizeof (k2));
}
```

Tamaño: resultado

```
n1 (n2, n3): 4 bytes
p1i (p2i):   4 bytes
*p1i (*p2i): 4 bytes
f1 (f2, f3): 8 bytes
p1f (p2f):   4 bytes
*p1f (*p2f): 8 bytes
c1:          1 bytes
k0 (k1):     3 bytes
k2 (k3):     4 bytes
```

Dirección y valor

```
#include <stdio.h>
void main (void)
{
  int    n1, n2, n3, *p1i, *p2i;
  double f1, f2, f3, *p1f, *p2f;
  char   c1, k0 [3], k1[]= "01", *k2="0123", *k3;

  printf ("&n1: %x &n2: %x &n3: %x &pi1: %x &pi2: %x\n",
              &n1,    &n2,    &n3,     &p1i,    &p2i);
  printf ("&f1: %x &f2: %x &f3: %x &pf1: %x &pf2: %x\n",
              &f1,    &f2,    &f3,     &p1f,    &p2f);
  printf ("&c1: %x k0: %x &k0: %x k1: %x k2: %x k3: %x\n",
              &c1,    k0,    &k0,    k1,   k2,    k3);
```

Dirección y valor (continuación)

```
  n1 =  10; n2 =  20; n3 =  30;
  f1 = 100; f2 = 200; f3 = 300;
  p1i = &n2; p1f = &f2;

  printf ("*p1i: %d *p1i+1: %d *(p1i+1): %d\n",
                *p1i,      *p1i+1,       *(p1i+1));

  printf ("*p1f: %.0lf *p1f+1: %.0lf *(p1f+1): %.0lf\n",
                 *p1f,         *p1f+1,          *(p1f+1));

}
```

Dirección y valor: resultado

```
&n1: 22ff6c &n2: 22ff68 &n3: 22ff64 &pi1: 22ff60 &pi2: 22ff5c
&f1: 22ff50 &f2: 22ff48 &f3: 22ff40 &pf1: 22ff3c &pf2: 22ff38
&c1: 22ff37 k0: 22ff20 &k0: 22ff20 k1: 22ff10 k2: 403003 k3: 77bfc2de
*p1i: 20 *p1i+1: 21 *(p1i+1): 10
*p1f: 200 *p1f+1: 201 *(p1f+1): 100
```

Operaciones básicas stdio.h

- Accederemos a los elementos (caracteres) de una cadena de igual manera que con cualquier otro vector:

```
v1[0] = 'a';
```

- Podemos leer una cadena mediante la función `gets`:

```
char nombre[80];
printf ("Introduce tu nombre: ");
gets (nombre);
```

- Esta función no sabe cuánto espacio hay reservado para la cadena
- Si no reservamos suficiente espacio para el texto introducido vamos a escribir más allá de la zona reservada para la cadena, produciéndose comportamientos extraños ya que otras variables pueden cambiar de valor

Operaciones básicas string.h

- En ANSI C hay una serie de funciones particulares para el tratamiento de cadenas de caracteres (string.h), entre ellas:
  - `strlen (cad)`: devuelve el número de caracteres de la cadena `cad` hasta el terminador
  - `strcpy (cad1, cad2)`: copia la cadena `cad2` a la cadena `cad1`
  - `strncpy (cad1, cad2, n)`: copia hasta `n` caracteres de la cadena `cad2` a la cadena `cad1`
  - `strcat (cad1, cad2)`: concatena `cad2` al final de `cad1`
  - `strcmp (cad1, cad2)`: compara `cad1` y `cad2`
  - `strncmp (cad1, cad2, n)`: compara hasta `n` caracteres de `cad1` y `cad2`

Operaciones básicas ctype.h

- En ANSI C hay una serie de funciones particulares para consultar el tipo de carácter (ctype.h), entre ellas:
  - `isalnum (c)`: consulta si el carácter `c` es alfanumérico
  - `isalpha (c)`: consulta si el carácter `c` es alfabético
  - `isctrl (c)`: consulta si `c` es carácter de control
  - `isdigit (c)`: consulta si `c` es dígito decimal
  - `isgraph (c)`: consulta si `c` es gráfico (excluido el espacio)
  - `islower (c)`: consulta si `c` es letra minúscula
  - `isprint (c)`: consulta si `c` es imprimible (incluido el espacio)
  - `ispunct (c)`: consulta si `c` es signo de puntuación
  - `isspace (c)`: consulta si `c` es un espacio separador
  - `isupper (c)`: consulta si `c` es mayúscula

Algoritmos de funciones string.h

- `strlen (cad)`: contar los caracteres de la cadena hasta llegar al terminador '\0' o carácter nulo
- `strcpy (cad1, cad2)`: copiar `cad2` en `cad1` carácter a carácter hasta llegar al carácter terminador '\0', que también habrá que copiar
- `strncpy (cad1, cad2, n)`: copiar un carácter de `cad2` a `cad1` hasta que se haya copiado el carácter terminador a no ser que lleguemos a copiar `n` caracteres, en cuyo caso pararemos de copiar
- `strcat (cad1, cad2)`: buscar el terminador de `cad1` y copiar `cad2` a partir de este punto
- `strcmp (cad1, cad2)`: comparar carácter a carácter `cad1` y `cad2` hasta que difieran o hasta que lleguemos al terminador (que habrá de estar en ambos), devolviendo la diferencia entre los últimos caracteres comparados
- `strncmp (cad1, cad2, n)`: como `strcmp` pero no comparará más de `n` caracteres

Codificación string.h

```
int strlen (char str[])
{
  int i;
  i = 1;
  while (str [i] != '\0')
    i++;
  return i;
}

char *strcpy (char str1[], char str2[])
{
  int i;
  for (i = 0; str2 [i] != '\0'; i++)
    str1[i] = str2[i];
  str1[i] = '\0';
  return str1; /* Devuelve un puntero a la cadena copiada */
}
```

Codificación string.h (1)

```
char *strcat (char str1[], char str2[])
{
  int n1, i;
  for (n1 = 0; str1 [n1] != '\0'; n1++)
    ;
  for (i = 0; str2 [i] != '\0'; i++)
    str1[n1+i] = str2[i];
  str1[n1+i] = '\0';
  return str1; /* Devuelve un puntero a la cadena copiada */
}

int strcmp (char str1[], char str2[])
{
  int i;
  for (i = 0; str1 [i] != '\0' && str1 [i] == str2
   [i]; i++)
    ;
  return str1[i] - str2 [i];
}
```

Ejemplo uso ctype.h

```
#include <ctype.h>

void a_minusculas (char str [])
{
  int i;

  for (i = 0; str [i] != '\0'; i++)
    if (isupper (str [i])
      str [i] += 'a' - 'A';
}
```

Escuela Universitaria de Ingeniería Vitoria-Gasteiz

Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea