Tema 4: Tipos Avanzados de Datos

Oscar Perpiñán Lamigueiro - David Álvarez

Cadenas de caracteres en C (*strings*)

Definición

Tipo particular de vector de caracteres (*char*) que acaba con el caracter $'\backslash 0'$.

Código

```
char identificador[tamaño];
```

tipo	`char`
identificador	Nombre de la cadena.
tamaño	Número de elementos de la cadena **incluído el carácter de cierre** (`\0`).

Definición e Inicialización de cadenas

```
// Declara una cadena de 10 caracteres (+1 para el cierre)
char cadena[5];
cadena[0] = 'H';
cadena[1] = 'o';
cadena[2] = 'l';
cadena[3] = 'a';
cadena[4] = '\0';

// Declara y asigna contenido
char cadena[5] = "Hola"; // 4 + 1
char cadena[] = "Hola"; // 4 + 1
```

Esto es un vector, no una cadena

```
char cadena[5] = {'H', 'o', 'l', 'a'};
```

Imprimir elementos de una cadena

- Como en un vector, el índice es la posición del elemento dentro de la cadena.
- La **primera posición** tiene el **subíndice 0**.
- La **última posición** es el carácter nulo \0.

```
int main()
{
  char cadena[] = "Hola buenos dias.";
  int i=0;
  while(cadena[i] != '\0')
  {
    printf("%c ", cadena[i]);
    i++;
  }
  printf("\n", cadena[i]); // Cambiamos de línea
  printf("%s\n", cadena); // Se imprime la cadena completa
  printf("%.8s\n", cadena); // Se imprimen 8 elementos de la cadena completa
  return 0;
}
```

Lectura y escritura de una cadena

- Usamos el especificador `%s` con `printf` y `scanf`.
- En `scanf` **debemos** especificar el **límite de caracteres** en el especificador de formato.
- **No** ponemos `&` delante del identificador.

```c
#include <stdio.h>

int main()
{
  char texto[8];

  printf("Dime algo: \n");
  // Deja de leer cuando detecta un espacio
  // Imponemos el límite de caracteres
  scanf("%7s", texto);
  printf("Has dicho %s", texto);
  return 0;
}
```

Lectura de una cadena con espacios

- `scanf` con `%s` termina de leer cuando recibe un espacio o salto de línea.
- Para leer cadenas con espacios se emplea el identificador `%[^\n]`
- O usar la función **gets**

```c
#include <stdio.h>

int main()
{
  char texto[11];
  printf("Dime algo: \n");
  // Deja de leer cuando detecta un salto de línea
  // o al alcanzar el límite de caracteres
  scanf("%10[^\n]", texto);

  printf("Has dicho %s\n", texto);
  return 0;
}
```

Lectura de una cadena con espacios (2), gets

- `scanf` con `%s` termina de leer cuando recibe un espacio o salto de línea.
- Para leer cadenas con espacios se emplea el identificador `%[^\n]`
- O usar la función **gets**

```c
#include <stdio.h>

int main()
{
  char texto[11];
  printf("Dime algo: \n");

  // No se controla el límite de caracteres
  gets(texto);

  printf("Has dicho %s\n", texto);
  return 0;
}
```

Recorrido por los elementos

- El bucle `while` es el más indicado, usando el carácter nulo para terminar:

```
#include <stdio.h>

int main()
{
  char cadena[5] = "Hola";
  int i = 0;
  printf("Los caracteres son:\n");
  while (cadena[i] != '\0')
  {
    printf("%c \t", cadena[i]);
    i++;
  }
  return 0;
}
```

Recorrido por los elementos (2)

- También se puede usar un bucle `for` (equivalencia entre `for` y `while`)

```
#include <stdio.h>

int main()
{
  char cadena[5] = "Hola";
  int i;
  printf("Los caracteres son:\n");
  for(i = 0; cadena[i] != '\0'; i++)
  {
    printf("%c \t", cadena[i]);
  }
  return 0;
}
```

Ejemplo: pasar a mayúsculas

```c
#include <stdio.h>

// Leer una cadena del teclado
// Pasar a mayúsculas las letras necesarias

int main() {
  char cadena[20];

}
```

Ejemplo: pasar a mayúsculas

```c
#include <stdio.h>
int main() {
	char cadena[20];
	int inc = 'A' - 'a';  // Distancia entre A y a
	int i = 0;
	printf("Dime algo: \n");
	gets(cadena);
	// Recorremos la cadena hasta encontrar el final
	while(cadena[i] != '\0') {
	// Si el caracter es letra minuscula
		if (cadena[i] >= 'a' && cadena[i] <= 'z') {
			cadena[i] = cadena[i] + inc; //sumamos la distancia para pasar
			    a mayuscula
		}
		i++;
	}
	printf("%s\n", cadena);
	return 0;
}
```

Librería `string.h`

La librería `string.h` incluye numerosas funciones dedicadas a cadenas de caracteres:

```
#include <string.h>
```

Longitud de una cadena `strlen`

Paso a mayúsculas `strupr`

Copiar cadenas `strcpy`

Concatenar cadenas `strcat`

Comparación de cadenas `strcmp`

Longitud de una cadena :: `strlen`

- `strlen` devuelve un entero con el número de caracteres.

```
#include <stdio.h>
#include <string.h>

int main()
{
  char palabra[21];
  int longitud;
  printf("Introduce una palabra: ");
  scanf("%20s", palabra);
  longitud = strlen(palabra);
  printf("Esta palabra tiene %i caracteres\n", longitud);

  return 0;
}
```

Copiar cadenas :: strcpy

Con strcpy tenemos una solución óptima para la **asignación de contenido**.

```
#include <stdio.h>
#include <string.h>
int main()
{
  char s1[50], s2[50];
  strcpy(s1, "Hello World!");
  strcpy(s2, s1);
  printf("%s\n", s2);

  return 0;
}
```

La cadena receptora debe tener espacio suficiente: *los caracteres sobrantes serán eliminados*.

Concatenar cadenas :: strcat

```c
#include <stdio.h>
#include <string.h>

int main() {
  char nombre_completo[50];
  char nombre[ ] = "Juana";
  char apellido[ ] = "de Arco";
  // Copiamos por tramos:
  // Primero el nombre
  strcpy(nombre_completo, nombre);
  // A continuacion un espacio
  strcat(nombre_completo, " ");
  // Finalmente el apellido
  strcat(nombre_completo, apellido);
  printf("El nombre completo es: %s.\n", nombre_completo);

  return 0;
}
```

Comparación de cadenas :: strcmp

- **Si las dos cadenas son iguales entrega un 0.**

```
#include <stdio.h>
#include <string.h>
int main()
{
  char color[] = "negro";
  char respuesta[11];
  do // El bucle se repite mientras
  { // las cadenas *no* coincidan
     printf("Adivina un color: ");
     scanf ("%10s", respuesta);
  } while (strcmp(color, respuesta) != 0);
  printf("¡Correcto!\n");
  return 0;
}
```

Comparación de cadenas :: strcmp

- Si hay diferencias, es positivo si el valor ASCII del primer carácter diferente es mayor en la cadena 1.

```
#include <stdio.h>
#include <string.h>
int main()
{
  char s1[] = "abcdef";
  char s2[] = "abCdef";
  char s3[] = "abcdff";
  int res;
  res = strcmp(s1, s2);
  printf("strcmp(s1, s2) = %i\n", res);
  res = strcmp(s1, s3);
  printf("strcmp(s1, s3) = %i\n", es);

  return 0;
}
```