Tema 5:
Estructuras de Datos

Índice

- Estructuras de datos en Memoria Principal:
  - Vectores
  - Matrices
  - **Cadenas de caracteres**
  - Estructuras

- Estructuras de datos en Memoria Externa:
  - Ficheros

Cadenas de caracteres

- No existe en C un tipo de datos concreto para definir las **cadenas de caracteres** o string.

- Una cadena de caracteres en C se implementa como un vector cuyos elementos son caracteres (tipo char).

- Por convenio, todas las cadenas de caracteres de C finalizan con el **carácter nulo**: '\0'

- La **longitud** de la cadena "hola" es igual a 4 caracteres (no se incluye el carácter nulo para el cálculo de su longitud), pero si se almacena, la cadena en memoria ocupa el espacio de 5 caracteres.

Cadenas de caracteres. Declaración

- Para **declarar** una cadena lo hacemos igual que con los vectores:

char nomb_array [tamaño]

Ejemplo:

```
char   nombre[20];
char   linea[80];
```

- La forma de **operar** con las cadenas de caracteres es igual que con los arrays, y además se incluyen **algunas ventajas**.

Cadenas de caracteres

- Para **inicializar** una cadena de caracteres podemos hacerlo de varias formas:
  - Mediante **caracteres individuales** (igual que con los arrays):
    - En la declaración
    - Con sentencias de asignación o lectura
  - En la **declaración**, mediante un **literal** cadena
  - La cadena completa, directamente desde el **teclado**

- Es importante observar que las cadenas contienen un carácter especial más de terminación de cadena, el **carácter nulo:** '\0'

Cadenas de caracteres. Inicialización

- Mediante **caracteres individuales**:
  - En la declaración

```
char cadena[] = {'h', 'o', 'l', 'a', '\0'}
```

  - Con sentencias de asignación o lectura:

```
cadena[0] = 'h';
cadena[1] = '0';
…
scanf("%c%c", &cadena[0], &cadena[1]);
…
```

/* En estos casos:
NO se le asigna automáticamente el carácter nulo */

Cadenas de caracteres. Inicialización

- En la **declaración**, mediante un **literal** cadena:

```
char cadena[] = “hola”;
```

inicializa la cadena con los caracteres individuales de la cadena “hola”. Se determina el **tamaño** basándose en la longitud de la cadena y añade un carácter más, el carácter especial de terminación de cadena, **carácter nulo:** ‘\0’

/* En este caso: SI se le asigna el carácter nulo */

Cadenas de caracteres. Inicialización

- La **cadena completa**, directamente desde el **teclado:** (con scanf y el formato de lectura %s)

```
char cadena2[10];       /* tamaño suficiente */
scanf("%s", cadena2); /* no lleva & */
```

lee caracteres hasta encontrar un espacio en **blanco**, un **tabulador** o un **salto de línea**

(recordar que el nombre del array, representa la dirección de memoria del comienzo del mismo y por eso no es necesario el operador de dirección & en scanf)

/* En este caso: SI se le asigna el carácter nulo */

Cadenas de caracteres. Inicialización

- La **cadena completa**, directamente desde el **teclado:** (con gets)

```
char apellidos[30];

printf (“Escribe tus dos apellidos: ”);

gets (apellidos);
```

lee caracteres hasta encontrar un encontrar un **‘\n’** (que no se añade al vector)

(La entrada introducida no debe rebasar el tamaño del vector – 1 (en el ejemplo 29 caracteres). La función gets devuelve la cadena o valor NULL si ha habido un error.)

/* En este caso: SI se le asigna el carácter nulo */

Cadenas de caracteres. Inicialización

- La **cadena completa**, mediante lectura de **ficheros de texto:** (con fscanf y el formato de lectura %s)

```
FILE *fich;

char cadena[10];     /* tamaño suficiente */

fscanf(fich, "%s", cadena);  /* no lleva & */
```

lee caracteres hasta encontrar un espacio en **blanco**, un **tabulador** o un **salto de línea**

(El puntero al fichero se queda posicionado justo después de la cadena, no de su delimitador)

/* En este caso: SI se le asigna el carácter nulo */

Cadenas de caracteres. Inicialización

- La **cadena completa**, mediante lectura de **ficheros de texto :** (con fgets)

```
FILE *fich;
char cadena[30];
int n=29;
fgets (cadena, n, fich);
```

lee caracteres hasta encontrar un encontrar un **'\n'** (que no se añade al vector)

(Lee caracteres del fichero, la lectura termina cuando lee el carácter '\n' , que también se guarda en la cadena, o bien cuando se han leído n (en el ejemplo 29) caracteres. El puntero al fichero se queda posicionado justo después del n-ésimo carácter leído o después del '\n' leído)

/* En este caso: SI se le asigna el carácter nulo */

Cadenas de caracteres

- Para **escribir** una cadena de caracteres:
  - Con la función **printf y %s**

```
printf (“%s”, cadena2);
```

  - Recorriendo los elementos de la cadena:

```
int  i=0;
while (cadena[i] != ‘\0’)
{
     printf(“%c”, cadena[i]);
      i++;
}
```

Cadenas de caracteres

- **Otra forma de escribir** cadenas de caracteres es con la función **puts()** que escribe en la salida una cadena de caracteres, **incluyendo el carácter fin de línea.**

Ejemplo:

```
char apellidos[30];
printf ("Escribe tus dos apellidos: ");
gets (apellidos);     // lee los 2 apellidos
puts (apellidos);     // escribe los 2 apellidos y salta
                      // de línea
```

Cadenas de caracteres. Resumen

- Escritura en ficheros de texto:
  - `fprintf(fichero,“%s”, cadena)`
  - `fputs(cadena, fichero)`

    //escribe la cadena en el fichero sin añadir el salto de línea
    //ni el carácter de fin de cadena en la escritura

Cadenas de caracteres. Resumen

- Se puede trabajar igual que con los arrays.
- Se puede asignar la cadena completa en la declaración: (asigna automáticamente el '\0')

```
char  cad[10] =“hola”
```

- Lectura (asigna automáticamente el '\0') :
  - `scanf(“%s”, cad)` // lee hasta el primer blanco, tabulador o salto de línea
  - `gets(cad)` //lee hasta el primer salto de línea
- Escritura:
  - `printf(“%s”, cad)` //escribe sin saltar de línea
  - `puts(cad)` //escribe y salta de línea

Cadenas de caracteres. Resumen

Con Ficheros de texto:

- Lectura (asigna automáticamente el '\0') :
  - **`fscanf(F, "%s", cad)`** /* lee hasta el primer blanco, tabulador o salto de línea *7
  - **`fgets(cad, n ,fich)`** /* lee hasta el primer salto de línea (que se añade a la cadena) */
- Escritura:
  - **`fprintf(F,"%s", cad)`** //escribe sin saltar de línea
  - **`fputs(cad)`** //escribe sin saltar de línea

Ejercicio

- Escribir una función que a partir de una cadena de caracteres **calcule su longitud**.

```
int longitud (char cad[])
{
        int i =0;
        while (           )









}
```

Ejercicio

- Realizar una función C que **copie** una cadena en otra.

```
void copiaCad (char c1[], char c2[])  // equivaldría a c1=c2
{
        int i=0;
        while (          )
        {




        }
        c1[i] =   ;
}
```

Ejercicio

- Realizar una función que **busque un carácter** en una cadena y devuelva, si existe, el índice de la primera aparición y -1 si no existe.

```
int buscarCar (char c[], char car)
{
   //busca el caracter car en la cadena c,
   int i=0;

   while (                          )




   if (              )
      i= -1;
   return i;
}
```

Ejercicio

- Realizar un subprograma C que compare dos cadenas y devuelva 1 si las **cadenas son iguales** y 0 en caso contrario.

```
int iguales (char c1[], char c2[])
//devuelve 1 si las cadenas son iguales y 0 en otro caso
{
  int i=0;
  while (                          )



  return          ;
}
```

Ejercicio

- Realizar un subprograma C que **concatene** dos cadenas y la almacene en otra cadena.

```
void concatena (char c1[], char c2[], char c3[])
{
// concatena las dos cadenas y trunca la c2 si fuese necesario
        int l1, l2, i, final;
        l1 = longitud(c1);
        l2 = longitud(c2);
        if (          )  final = l1+l2;
        else final =    ;

        for (i=  ;             ; i++)
                c3[   ] = c2[     ];
        c3[     ]='\0';
}
```