Programación I

Práctica #4 **Curso 2024-2025** **1º curso**

Arrays

Introducción

En esta práctica, profundizaremos en el estudio de los tipos de datos.

En las prácticas anteriores, hemos trabajado con tipos de datos simples, en los que cada variable tiene un único valor.

En esta práctica, introduciremos el tipo estructurado de datos "`array`". Una variable de este tipo consiste en una colección numerada de variables del mismo tipo, a las que accedemos mediante un índice.

Estudiaremos la declaración y la sintaxis de este tipo de datos.

Además, introduciremos los arrays con 2 índices, es decir, los llamados arrays bidimensionales.

A lo largo de la práctica, seguiremos trabajando con las instrucciones que nos proporciona el lenguaje C para codificar la ejecución repetitiva.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Empezar a utilizar variables estructuradas del tipo "array".
- Seguir trabajando con las instrucciones de control de flujo del lenguaje C.
- Aprender a trabajar con números aleatorios
- Empezar a manejar cadenas de caracteres.

Planificación

Sesión 4 de laboratorio (3 horas)

1. Realización de los ejercicios 1 y 2.
2. Realización de los ejercicios 3 y 4.
3. Realización de los ejercicios 5 y 6.

Trabajo fuera del aula:

- Terminar los ejercicios no completados en el aula.

Arrays

Arrays unidimensionales

Como dijimos más arriba, cada variable perteneciente al tipo estructurado de datos llamado `array` consiste en una colección numerada de variables, todas ellas del mismo tipo, al que, a su vez, llamamos tipo base.

También podemos ver un `array` como una tabla, formada por elementos de un tipo base, en la que cada uno de esos elementos tiene asociado un número natural, al que llamamos `índice`.

Su declaración más simple tiene la sintaxis:

```
<tipo> <identificador> [numero_entero];
```

Por ejemplo, para declarar un `array`, llamado "`tabla`", formado por 10 variables del tipo `int`, usaríamos la siguiente instrucción:

```
int tabla[10];
```

la cual reserva espacio de memoria para las variables indexadas `tabla[0]`, `tabla[1]`,... , `tabla[9]`.

Denominamos indexación a la operación por medio de la que nos referimos a la variable con índice `i` de dicha "`tabla`" (que representamos mediante `tabla[i]`).

Por ejemplo, para asignar a la variable que ocupa la 7ª posición del mencionado `array` "`tabla`" (es decir, `tabla[6]`) el valor de una variable entera "`a`", usaríamos la siguiente instrucción:

```
tabla[6] = a;
```

Debemos tener en cuenta que la primera variable del `array` es la variable que tiene índice `0`.

Ejercicio 1:

Ve al directorio `pr04`.

Escribe un programa (`ej1.c`) que declare y use un `array` formado por 5 variables de tipo entero.

El programa deberá asignar a cada elemento un valor igual al índice del elemento multiplicado por 10.

Por último, el programa mostrará en pantalla los valores almacenados, cada uno de ellos en una línea, y en un campo de anchura 3:

```
  0
 10
 20
 30
 40
```

Ejercicio 2:

Copia el ejercicio 1:

```
usuario@maquina:~/pr04$ cp ej1.c ej2.c
```

Modifícalo, de forma que, ahora, en vez de asignar valores fijos, el programa genere aleatoriamente 5 valores de tipo entero entre -10 y +10, y los almacene en el `array`. Se sugiere el uso de las funciones `srand()` y `rand()`.

Por último, el programa mostrará en pantalla los valores almacenados, cada uno de ellos en una línea, y en un campo de anchura 3:

```
 -9
  2
 -5
  9
  7
```

Control de la ejecución repetitiva

Ejercicio 3:

Copia el ejercicio 1 de la Práctica #3:

```
usuario@maquina:~/pr04$ cp ../pr03/ej1.c ej3.c
```

Modifícalo, utilizando el código que has desarrollado en el ejercicio 7 de la Práctica #3, para que tenga el siguiente comportamiento:

El programa presentará lo siguiente por pantalla:

```
N) New reader
A) Add book
V) Submit vote
C) Clean reader
X) Exit

Enter operation:
```

Cuando se elija alguna de las opciones `'N'`, `'A'`, `'V'` o `'C'` se presentará un mensaje indicando la opción seleccionada y se volverá a presentar el menú.

Si se introduce la opción `'X'`, el programa debe terminar. Pero, antes, el programa deberá solicitar una confirmación como la siguiente:

```
Confirm exit? (YN):
```

Si el usuario contesta afirmativamente (`'Y'` o `'y'`), se saldrá del programa, y, si contesta negativamente (`'N'` o `'n'`), se volverá a presentar el menú principal. Con cualquier otra contestación, se volverá a mostrar el mensaje de confirmación.

Arrays bidimensionales

Los `arrays` que vimos antes tienen un único índice; sin embargo el tipo `array` admite múltiples índices, lo que nos permite mantener tablas multidimensionales.

Concretamente, las tablas bidimensionales (es decir, los `arrays` con 2 índices) son muy útiles, ya que nos permiten representar de una forma sencilla matrices de MxN elementos de un determinado tipo base.

Por ejemplo, para declarar un `array`, llamado "`matriz`", formado por 10x8 enteros (es decir, es una matriz de 10 filas x 8 columnas, cuyo el tipo base es `int`), lo haríamos con la siguiente instrucción:

```
int matriz[10][8];
```

Obviamente, aquí, la indexación requiere de 2 índices.

Por ejemplo, para asignar al 4º elemento de la 7ª fila de dicho `array` "`matriz`" el valor de una variable entera "`a`", lo haríamos con la siguiente instrucción:

```
tabla[6][3] = a;
```

Recuerda que los índices empiezan en 0.

Ejercicio 4:

Copia el ejercicio 2:

```
usuario@maquina:~/pr04$ cp ej2.c ej4.c
```

Modifícalo para que el programa declare y use un `array` formado por 5x3 variables de tipo entero.

A continuación el programa deberá almacenar en ese `array` valores aleatorios entre -10 y +10.

Finalmente el programa deberá mostrar por pantalla, en forma de matriz, los 5x3 valores almacenados.

Por ejemplo, el programa podría mostrar:

```
-2  4  0
 1  2 -3
 0  3 -2
-2  4 -9
-4 -2  3
```

Arrays de cadenas

Un array de cadenas es un array bidimensional de caracteres:

```
char palabras[3][21];
```

Con esa declaración, creamos un array formado por 3 cadenas de hasta 20 caracteres de longitud.

En él, para acceder a un carácter individual, utilizamos los 2 índices:

```
printf ("%c", palabras[1][12]);
```

Con esa instrucción, mostramos el 13º carácter de la 2ª cadena.

Pero, para acceder a una de las cadenas que forman el array, utilizamos solo 1 índice:

```
printf ("%s", palabras[2]);
```

Con esa instrucción, mostramos la 3ª cadena.

Ejercicio 5:

Escribe un programa (`ej5.c`) que:

1) Declare un array formado por 5 cadenas de 25 caracteres (ojo, hay que dejar espacio para el `'\0'`).

2) Le pida al usuario que introduzca 5 cadenas de hasta 25 caracteres, y las almacene en dicho array de cadenas.

3) A continuación, le pida al usuario que introduzca un número entre 1 y 25.

4) Pase a mayúsculas el carácter que ocupa la posición dada por ese número en las 5 cadenas del array. Es decir, si se da un 1, se pasará a mayúsculas el 1ª carácter de cada cadena (ojo a los índices).

5) A continuación, le pida al usuario que introduzca un número entre 1 y 5.

6) Finalmente, muestre por pantalla la cadena introducida en la posición dada por ese número. Es decir, si se da un 1, se mostrará la cadena introducida en la 1ª posición (ojo a los índices).

```
Escribe una cadena (1-25 char): hola
Escribe una cadena (1-25 char): amigo
Escribe una cadena (1-25 char): como
Escribe una cadena (1-25 char): estas
Escribe una cadena (1-25 char): hoy
Posicion del caracter a convertir [1-25]: 2
Posicion de la cadena a mostrar [1-5]: 5

La cadena en la posicion 5 es hOy
```

Entrada de datos con formato

Ejercicio 6:

Escribe un programa (`ej6.c`) que le solicite al usuario que introduzca una línea con el siguiente formato:

```
ID Nombre
```

Donde `"ID"` es un número entero no negativo, y `"Nombre"` es una cadena formada por cualquier carácter no blanco, por ejemplo:

```
10 Iago_Aspas
```

Y, finalmente, muestre por separado el ID y el nombre recibidos. Por ejemplo:

```
Escribe la linea con el formato "ID nombre": 10 Iago_Aspas
ID: 10
nombre: Iago_Aspas
```

Comprueba con tu programa, y reflexiona: ¿qué ocurre si el usuario introduce una línea que no respeta el formato pedido?

Resumen

Los principales resultados esperados de esta práctica son:

- Conocer el tipo estructurado de datos "array".

Como posibles líneas de trabajo adicional del alumno, se proponen las siguientes:

- Terminar los ejercicios no completados en clase.