Programación I

Práctica #3 Curso 2024-2025 1º curso

Las instrucciones de iteración en C

Introducción

En esta práctica, en primer lugar, aprenderemos a usar las instrucciones de control de flujo que nos proporciona el lenguaje C para codificar la **iteración**, es decir, la ejecución **repetitiva** de una instrucción o de un bloque de instrucciones:

- `do-while`
- `while`
- `for`

También aprenderemos a utilizar dos instrucciones que nos permiten alterar el control de flujo del programa:

- `break`
- `continue`

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Aprender a utilizar las instrucciones de control de flujo del lenguaje C para la ejecución repetitiva (while, do-while, for)
- Aprender a utilizar las instrucciones para alterar el control de flujo (break, continue)

Planificación

Sesión 3 de laboratorio (3 horas)

1. Realización de los ejercicios 1 a 4
2. Realización de los ejercicios 5 a 7.
3. Realización de los ejercicios 8 y 9.

Trabajo fuera del aula:

- Terminar los ejercicios no completados en el aula.
- Seguir mejorando el código del ejercicio 8.

Instrucciones de control de repetición

Las instrucciones de control de repetición (`while`, `do-while` y `for`), también denominadas "**bucles**", se utilizan para ejecutar varias veces un bloque de instrucciones. A cada ejecución de dicho bloque de instrucciones la denominamos **una iteración**.

La instrucción `do-while`

La sintaxis de la instrucción `do-while` es la siguiente:

```
do {
   <bloque de instrucciones>
} while (<expresión>);
```

La instrucción de repetición `do-while` se utiliza cuando tenemos un bloque de instrucciones que queremos que se ejecute varias veces. El número de veces que se ejecutará el bloque dependerá de la evaluación de una condición, de tal forma que mientras dicha condición sea cierta (es decir, tenga un valor distinto de cero), el bloque se continuará ejecutando.

Puesto que la condición se evalúa después de ejecutar el bloque de instrucciones, dicho bloque siempre se ejecutará al menos una vez. Por tanto utilizaremos esta instrucción cuando queramos que el bloque de instrucciones se ejecute **una o más veces**.

Por ejemplo, para imprimir 10 líneas con los números enteros consecutivos entre 0 y 9, podríamos hacer:

```
i = 0;
do {
   fprintf (stdout, "%d\n", i);
   i++;
} while (i < 10);
```

El bloque de instrucciones se ejecutará mientras la condición indicada por la expresión se cumpla. Observe que el signo ';' al final del `do-while` es obligatorio.

Ejercicio 1:

Ve al directorio pr03. Copia el ejercicio 6 de la Práctica #2:

```
usuario@maquina:~/pr03$ cp ../pr02/ej6.c ej1.c
```

Modifícalo para que presente al usuario el menú:

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

Enter operation:
```

El programa presentará los mismos mensajes indicados en dicho ejercicio 6, pero, ahora, tras presentar el mensaje correspondiente a la opción elegida, volverá a presentar el menú, y continuará repitiendo este modo de operación hasta que el usuario seleccione la opción `'X'`, momento en que finalizará la ejecución del programa.

La instrucción `while`

La sintaxis de esta instrucción es la siguiente:

```
while (<expresión>) {
   <bloque de instrucciones>
}
```

La instrucción de repetición `while` es similar a la anterior. La única diferencia es que la condición que regula la ejecución del bucle se comprueba antes de ejecutar dicho bloque. Por tanto, el bloque de instrucciones se ejecutará **cero o más veces**, en función del valor de la condición.

Por ejemplo, para obtener la misma salida del ejemplo de la instrucción `do-while`, haríamos:

```
i = 0;
while (i < 10) {
   fprintf (stdout, "%d\n", i);
   i++;
}
```

Al igual que en el caso anterior, el bloque de instrucciones se ejecutará mientras la condición indicada por la expresión sea cierta (tenga un valor distinto de 0).

Ejercicio 2:

Reflexiona:

¿Qué instrucción de repetición es más conveniente para el ejercicio 1: `do-while` o `while`?

Ejercicio 3:

Escribe un programa (ej3.c) que solicite al usuario un valor entero (`N`) y, a continuación, usando la instrucción `while`, le vaya solicitando `N` números reales y mostrando su valor al cuadrado..

Por ejemplo, una posible ejecución del programa sería:

```
Dame el numero de valores: 4
Dame un numero real: 1.25
Cuadrado de 1.25 = 1.56
Dame un numero real: 80
Cuadrado de 80.00 = 6400.00
Dame un numero real: -4.7
Cuadrado de -4.70 = 22.09
Dame un numero real: 100
Cuadrado de 100.00 = 10000.00
```

¿En este caso, qué es más conveniente: `do-while` o `while`?

Ejercicio 4:

Copia el ejercicio 3 de la Práctica #2:

```
usuario@maquina:~/pr03$ cp ../pr02/ej3.c ej4.c
```

Modifícalo para que repita la invitación a introducir el número hasta que el usuario entregue un entero no menor que 0 y no mayor que 100. Una vez obtenido un número correcto, el programa mostrará su raíz y terminará.

Por ejemplo:

```
Dame un entero [0-100]: 147

Dame un entero [0-100]: -10

Dame un entero [0-100]: 75
RAIZ:      8.66
```

Antes de codificar: ¿qué instrucción de repetición es más conveniente para este ejercicio: `do-while` o `while`?

Ejercicio 5:

Copia el ejercicio 5 de la Práctica #2:

```
usuario@maquina:~/pr03$ cp ../pr02/ej5.c ej5.c
```

Modifícalo para que repita la invitación a introducir una cadena hasta que el usuario entregue una cadena con una longitud no mayor de 25 caracteres. Una vez obtenida una cadena correcta, el programa mostrará la longitud de la cadena y terminará.

Por ejemplo:

```
Dame una cadena (1-25 char): Programacion_en_lenguaje_C

Dame una cadena (1-25 char): Programacion

La cadena tiene 12 caracteres
```

Antes de codificar: ¿qué instrucción de repetición es más conveniente para este ejercicio: `do-while` o `while`?

Ejercicio 6:

Copia el ejercicio 4 de la Práctica #2:

```
usuario@maquina:~/pr03$ cp ../pr02/ej4.c ej6.c
```

Modifícalo para que repita la invitación a introducir un carácter hasta que el usuario entregue un carácter incluido en la cadena `"NAVCSX"`. Una vez obtenido un carácter correcto, el programa mostrará dicho carácter y terminará.

Por ejemplo:

```
Dame un caracter (NAVCSX): 2

Dame un caracter (NAVCSX): E

Dame un caracter (NAVCSX): s
La letra s en mayusculas es S
```

Antes de codificar: ¿qué instrucción de repetición es más conveniente para este ejercicio: `do-while` o `while`?

Ejercicio 7:

Copia el ejercicio 1 de la Práctica #2:

```
usuario@maquina:~/pr03$ cp ../pr02/ej1.c ej7.c
```

Modifícalo para que repita la invitación a tomar una decisión, hasta que el usuario introduzca una respuesta válida (`'y'`, `'n'`, `'Y'` o `'N'`). Una vez obtenida una respuesta correcta, el programa informará de la decisión y terminará.

Por ejemplo:

```
Confirm exit? (YN): p
Confirm exit? (YN): u
Confirm exit? (YN): n
You have chosen "No"
```

¿Qué instrucción de repetición es más conveniente para este ejercicio: `do-while` o `while`?

La instrucción `for`

La sintaxis de la instrucción `for` es la siguiente:

```
for (<inicialización>;<continuación>;<actualización>) {
    <bloque de instrucciones>
}
```

La instrucción de repetición `for`, en su formato más típico, permite ejecutar un bloque de instrucciones un número determinado de veces.

Su comportamiento habitualmente está controlado por una[1] variable que denominamos "de control" y tres expresiones:

- la primera expresión (`inicialización`) asigna un valor inicial a la variable de control,
- la segunda expresión (`continuación`) es la condición que debe cumplir dicha variable para que el bucle continúe y
- la tercera expresión (`actualización`) indica cómo se debe actualizar la variable de control en cada iteración del bucle[2].

Esta instrucción tiene muchas posibilidades, pero, en el caso más común, se usa para repetir un bloque de instrucciones un número determinado de veces.

El mismo ejemplo que hemos visto con `do-while` y `while`, en el caso de la instrucción `for` sería:

```
for (i=0; i<10; i++) {
     fprintf (stdout, "%d\n", i);
}
```

En este ejemplo, el bloque de instrucciones se ejecutaría 10 veces.

Ejercicio 8:

Escribe un programa (`ej8.c`) que solicite al usuario un valor entero (`N`) y reproduzca por pantalla `N` líneas, cada una de ellas con 50 signos `'-'`.

Así, por ejemplo, una posible ejecución del programa sería:

```
Dame el numero de líneas: 4

--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
```

[1] Aunque no necesariamente. Una instrucción `for` puede estar controlada por múltiples variables de control.

[2] Las tres expresiones que intervienen en la instrucción `for` pueden ser mucho más complejas.

Las instrucciones `break` y `continue`

Asociado a las sentencias de repetición (`while`, `do-while` y `for`) podemos utilizar otras dos instrucciones útiles en determinados casos:

- **`continue`**: da por finalizada la iteración actual. En el caso del `for`, pasa a evaluar la expresión de actualización, y, después, la de continuación. En el caso del `while` o del `do-while`, pasa directamente a evaluar la expresión de continuación

- **`break`**: finaliza el bucle de forma inmediata, y pasa a ejecutar la instrucción siguiente dentro del programa.

Ejercicio 9:

Copia el ejercicio 5:

```
usuario@maquina:~/pr03$ cp ej5.c ej9.c
```

Modifícalo, haciendo uso de la instrucción **`break`**, de forma que finalice:
- o bien cuando el usuario haya introducido una cadena correcta,
- o bien cuando el usuario haya introducido 5 cadenas inválidas.

Resumen

Los principales resultados esperados de esta práctica son:

- Conocer el manejo de las instrucciones de iteración del lenguaje C.
- Conocer las instrucciones para alterar el control de flujo en C.

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

- Terminar los ejercicios no completados en clase.
- Seguir practicando con los ejercicios de los Anexos II y III de la Práctica #1