Práctica 4: Estructuras de control iterativas (2 sesiones)

Programación 1. Grado en Ingeniería Multimedia

10 de octubre de 2016

Objetivos:
• Conocer y manejar con habilidad los distintos tipos de estructuras de control iterativas.

1.

Realiza un programa que pida un número por teclado y muestre la tabla de multiplicar de dicho número, el programa deberá pedir al final si se quiere volver a repetir.

2.

Juego de suma 10: consiste en acertar el máximo número de sumas de un total de 10.

Para ello:

- Debemos generar dos números aleatorios n1 y n2 entre 0 y 9
- Debemos mostrar la suma que tiene que acertar el usuario
- Debemos pedir el resultado de la suma al usuario.
- Si el resultado es correcto aumentaremos los aciertos y diremos que ha acertado. Si no ha acertado, diremos que ha fallado.
- Al terminar de preguntar las diez sumas mostraremos el número de aciertos.

Además, al finalizar preguntaremos al usuario si quiere volver a jugar y si es así repetiremos el juego desde el principio. A continuación mostramos varios ejemplos de ejecución:

Ejemplo de programa que genera 10 números aleatorios entre 0 y 9:

```
5 + 8 = 13
Acierto!
4 + 5 = 9
Acierto!
3 + 5 = 7
Error!
8 + 7 = 15
Acierto!
5 + 5 = 10
Acierto!
5 + 3 = 8
Acierto!
1 + 7 = 8
Acierto!
```

3.

Implementa un algoritmo que lea un conjunto de números reales hasta que se introduzca el valor cero y, a continuación, calcule y visualice la media y la varianza de los números introducidos.

$$media = \frac{\sum_{i=1}^{n} x_i}{n} \qquad varianza = \frac{\sum_{i=1}^{n} x_i^2}{n} - media^2$$

4.

Implementa un programa que represente gráficamente un polinomio de tercer grado f(x) = ax3+bx2+cx+d, de forma invertida (es decir, los valores de x en el eje vertical y los de f(x) en el horizontal). Para ello el programa solicitará en primer lugar el número de valores que se quieren representar (empezando en 0) y el valor de cada uno de los cuatro coeficientes del polinomio, que serán enteros. El programa debe controlar que el número de valores y los coeficientes sean todos mayores o iguales que 0.

Comprobación: representar el polinomio x^2+4

```
Introduce número de valores: 6
Introduce coeficiente a: 0
Introduce coeficiente b: 1
Introduce coeficiente c: 0
Introduce coeficiente d: 4

0  |    *
1  |     *
2  |        *
3  |             *
4  |                   *
5  |                            *
```

5.

Implementa un programa en C que pida por teclado un número n, que deberá cumplir que sea mayor que 0 y menor o igual a 9. En caso de que la entrada de dicho dato sea incorrecta, el programa deberá volver a solicitarlo hasta que sea correcto.

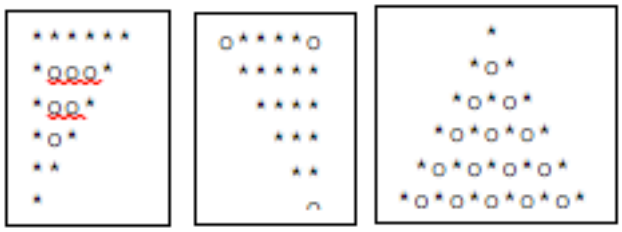
Después se deberá imprimir por pantalla todas las posibles parejas de valores (i,j), hasta (n,n).

Comprobación:

```
pc-rosana:pruebasC rosana$ ./ej
Introduce N (0<N<=9): -1
El valor introducido es incorrecto. Vuelve a introducirlo
Introduce N (0<N<=9): 10
El valor introducido es incorrecto. Vuelve a introducirlo
Introduce N (0<N<=9): 6
(1,1) (1,2) (1,3) (1,4) (1,5) (1,6)
(2,1) (2,2) (2,3) (2,4) (2,5) (2,6)
(3,1) (3,2) (3,3) (3,4) (3,5) (3,6)
(4,1) (4,2) (4,3) (4,4) (4,5) (4,6)
(5,1) (5,2) (5,3) (5,4) (5,5) (5,6)
(6,1) (6,2) (6,3) (6,4) (6,5) (6,6)
pc-rosana:pruebasC rosana$
```

6.

Implementa un programa que visualice en pantalla cada una de las siguientes figuras, preguntando al usuario el número de filas n que deben tener las figuras (En el ejemplo mostrado, n = 6). Ten en cuenta que la solución que propongas debe contener sentencias de salida por pantalla en las que se impriman exclusivamente uno solo de los siguientes caracteres: blanco ' ', asterisco '*' y letra 'o'.



7.

Escribe un programa que pida reiteradamente una fecha (día, mes y año) hasta que ésta sea válida y muestre el día siguiente a dicha fecha. Para ello, se debe calcular la cantidad de días que tiene el mes del año, teniendo en cuenta los años bisiestos. Recuerda que un año es bisiesto si es divisible por 400 o bien si es divisible por 4 y no es divisible por 100.

8.

El otro día recibí un email que decía lo siguiente:

EL TRUCO DE UN MATEMATICO RUSO QUE LOS CASINOS NO QUIEREN QUE CONOZCAS!!!!

Como sabes, en la ruleta puedes apostar a rojo o negro. Si no sale el color elegido, pierdes la cantidad apostada pero si sale ganas el doble. Pues bien, AQUÍ TIENES UN TRUCO PARA GANAR SIEMPRE: lo único que debes hacer es apostar siempre al rojo. Si, como lo oyes, es tan simple como eso. Pero… ¿y si no sale el rojo?. Muy sencillo: lo único que debes hacer es apostar de nuevo la cantidad perdida pero multiplicándola por 2.5 (y redondeando por arriba).

Por ejemplo imagínate que apuestas 1€ al rojo y sale negro. No pasa nada, vuelves a apostar 3€ al rojo (1x2.5 redondeado por arriba). Supón que vuelves a tener mala suerte y sale negro otra vez. Apostarías ahora 8€ (3x2.5 redondeado) al rojo. Supón que ahora sí sale rojo. Has perdido 1+3+8=12, pero has ganado 16!!!. Un beneficio de 4!!. Ponte tú mismo otros casos y verás como SIEMPRE ganas algo, ya que en algún momento TIENE QUE SALIR el rojo!!!.

No obstante, antes de ir al casino *online*, decides escribir un programa que simule esta estrategia de juego, para ver qué pasaría. Para empezar pide por teclado el dinero inicial que tiene el jugador. La apuesta inicial debe ser 1.

Puedes representar el rojo como 0 y el negro como 1 por lo que en cada tirada debes generar un valor aleatorio 0 ó 1. Mientras salga negro y el jugador tenga todavía dinero, sigue el bucle de tiradas. En cada tirada primero debes restar la apuesta actual al dinero que tiene el jugador. Si gana, súmale el doble de lo apostado. Si pierde, actualiza la apuesta multiplicándola por 2.5 y redondeando por arriba.

Al final el programa debe imprimir el dinero con el que queda el jugador tras terminar el bucle de juego.

Recuerda que debes inicializar la semilla de los valores aleatorios con srand ya que si no siempre dará el mismo resultado.

Ejecútalo manualmente varias veces y comprueba si esta estrategia funciona. Antes de ir al casino a jugarte tus ahorros, modifica el programa anterior para que se ejecute todo un número muy alto de veces, digamos 10000. Al principio de cada iteración de estas el jugador empieza la simulación con el capital inicial y apostando 1€. Al final contabiliza si el jugador ha acabado con capital positivo o negativo. Imprime el número de veces que pasa cada cosa para poder verificar si es verdad que esta estrategia SIEMPRE gana.

AVISO de los profes de P1: esto NO SIEMPRE funciona. Podría salir negro un número muy elevado de veces y hacer que te quedes sin dinero para seguir apostando. Y esto es improbable pero no tanto como parece. Esta estrategia de juego no es de ahora sino que se conoce desde el siglo XVIII como "martingala" (podéis buscar un artículo que lo explica perfectamente, buscad en Google "martingala microsiervos").

ANEXO: GENERAR NÚMEROS ALEATORIOS

```
#include <stdlib.h> //se debe añadir esta librería
        int i,n;
        srand((unsigned)time(NULL)); //semilla
        for (i=0;i<10;i++)
        {
            n=rand()%10; //número entre 0 y 9
            cout << n<<endl;
        }
```