Facultad de Informática – Universidad Complutense
1º curso de los Grados

Fundamentos de la programación

Curso 2012–2013

Hoja de ejercicios del Tema 11

1. Escribe un subprograma recursivo que reciba un número entero positivo $n \geq 1$ y muestre sus cifras en orden inverso en la pantalla. Por Ejemplo:

Número: `1234` Al revés: `4321`

2. Escribe un subprograma recursivo que reciba un número entero positivo $n \geq 1$ y devuelva su número de dígitos.

3. Escribe una función recursiva `invierte()` que acepte caracteres por teclado hasta recibir un INTRO y los imprima en orden inverso al de lectura. Los caracteres se leerán uno a uno y no deben almacenarse en un array; basta emplear una sola variable local a la función `invierte()` de tipo `char`.

4. Escribe una función recursiva que diga si una cadena contiene un palíndromo.

5. Escribe una función recursiva `pascal(i,j)` que calcule el elemento `i`, `j` del *Triángulo de Pascal* (o *de Tartaglia*), que sigue el siguiente patrón: los elementos en el borde del triángulo son unos y el resto de los elementos son iguales a la suma de los dos elementos que hay sobre ellos:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
1   6  15  20  15   6   1
          ...
```

6. Escribe un subprograma recursivo `antes()` que visualice en la pantalla todos los números pares que existan en los `m` valores enteros anteriores a `n`, incluido éste. Por ejemplo, la ejecución de la llamada `antes(8, 3)` debe visualizar 2 0 –2 –4; la ejecución de la llamada `antes(3, 12)` debe visualizar 12 10; la ejecución de la llamada `antes(0,4)` no debe visualizar ningún valor.

7. Dado el siguiente subprograma recursivo:

```
void acme(int n) {
   if (n > 0) {
       cout << setw(n) << "A" << endl;
       acme(n - 1);
       cout << setw(n) << "B" << endl;
       acme(0);
   }
}
```

a) Indica y razona la salida exacta producida al efectuar la llamada `acme(5)`.

b) Escribe una versión iterativa de ese subprograma.

8. Sean los términos de una serie A_0, A_1, A_2, ..., A_{n-1}, A_n, donde A_i es el término i-ésimo y los términos se definen como:

$A_0 \rightarrow d$
$A_1 \rightarrow A_0 + d$
$A_2 \rightarrow A_0 + 2 * d$
.
.
$A_n \rightarrow A_0 + n * d$

A la hora de programar una posible suma de los n primeros términos de la serie se nos ocurre los siguiente:

```
entero suma(A, D, N : enteros)
   si N = 0  → A
   si N <> 0 → A + suma(A + D, D, N - 1)
```

a) Implementa una función recursiva para sumar los n primeros términos.

b) Implementa una función para sumar los n primeros términos de la serie, usando un método iterativo.

c) Implementa una función sin iteración ni recursión

Una manera de expresar la potencia es mediante la definición recurrente que sigue:

$$x^n = \begin{cases} 1 & si \quad n = 0 \\ (x * x)^{\frac{n}{2}} & si \quad n > 0 \;\; y \;\; n\ es\ par \\ x \cdot x^{n-1} & si \quad n > 0 \;\; y \;\; n\ es\ impar \end{cases}$$

Dados x, un número real, y n ≥ 0, un entero positivo, escribe una función recursiva `potRec2()` que calcule la potencia de acuerdo con esa definición y pruébala en un programa principal.

9. Dado el siguiente subprograma recursivo:

```
void f(int num, int div) {
   if (num > 1) {
      if ((num % div) == 0) {
         cout << div << endl;
         f(num / div, div);
      } else
         f(num, div + 1);
   }
}
```

a) Dado un entero cualquiera `x`, ¿qué muestra la llamada `f(x, 2)`? ¿Cuál sería un nombre más adecuado para la función `f()`?

b) Implementa una versión iterativa equivalente.

10. En un programa en C++ nos prohíben utilizar las instrucciones `do-while` y `for`. ¿Cómo reescribirías los siguientes fragmentos?

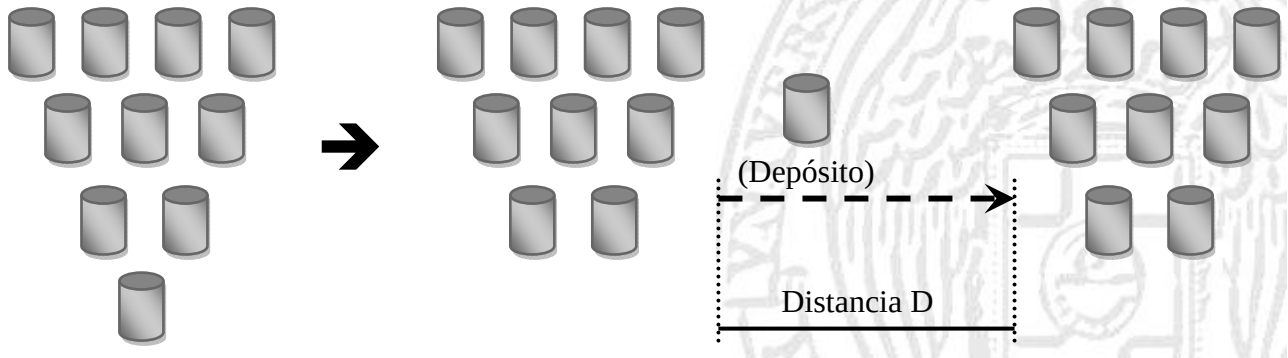
```
do {
   Instrucción1;
   Instrucción2;
} while condicion;
```

```
for(int i = 1; i < n; i++) {
   Instrucción1;
   Instrucción2;
}
```

11. *El problema del jeep*: En un oasis del desierto se encuentra una furgoneta y junto a ella hay N bidones de gasolina. En el depósito de la furgoneta cabe exactamente un bidón y la furgoneta sólo puede transportar un bidón cada vez. Además, sólo se puede llenar el depósito cuando éste esté vacío. Con cada depósito (un bidón) la furgoneta puede recorrer una distancia A (en kilómetros).

Escribe un subprograma recursivo que tome como entrada el número de bidones que tenemos y devuelva la distancia máxima que podemos recorrer con la furgoneta (en kilómetros).

Pista: ¿qué distancia podemos recorrer para transportar N–1 bidones lo más lejos posible con otro bidón en el depósito?



12. Dada las siguientes declaraciones:

```
int const N = 100;
typedef int tTabla[N];
```

Escribe subprogramas recursivos para:

a) Calcular el valor máximo de los elementos de una tabla de tipo `tTabla`.

b) Calcular la media de los elementos de una tabla de tipo tTabla.

c) Comprobar si todos los elementos de una tabla de tipo tTabla son iguales.

d) Comprobar si están ordenados los elementos de una tabla de tipo tTabla.

e) Contar el número de ocurrencias de un valor en la tabla de tipo tTabla.

13. *El problema de las ocho reinas*: Se quiere colocar ocho reinas en un tablero de ajedrez de forma que ninguna de ellas *coma* a alguna otra. Obviamente, cada una estará en una fila distinta:

Debes desarrollar un programa que obtenga una posible distribución de las reinas que sea una solución correcta. Para hacerlo, busca, para cada fila, una posición posible de la siguiente reina que sea segura (no le come ninguna reina anterior). Si no es posible, se volverá a la reina anterior para probar otras posiciones posibles. Si se consigue colocar la octava reina, se habrá resuelto el problema.

```
bool colocar(tTablero &tablero, int reina);
```

El subprograma intentará colocar ese número de `reina` en la fila `reina` sin que le coman. Si lo consigue, hará una llamada recursiva para que se intente colocar la siguiente reina (`reina + 1`). Si se consigue colocar la octava, terminará devolviendo `true`. Si no se consigue, devolverá `false`.

¿Cuándo se comen entre sí dos reinas?

	0	1	2	3
0			R	
1				
2				
3			R	

Misma columna

	0	1	2	3
0				R
1				
2				
3	R			

Igual diferencia entre filas y entre columnas:

3 - 0 y 0 - 3

	0	1	2	3
0		R		
1				
2				R
3				

Igual diferencia entre filas y entre columnas:

2 – 0 y 3 - 1

O sea: o están en la misma columna, o tienen igual diferencia, en valor absoluto, entre las filas y las columnas ($|3 - 0| = 3$, $|0 - 3| = 3$ o $|2 - 0| = 2$, $|3 - 1| = 2$).