MOOC Init. Prog. C++
Ejercicios adicionales opcionales semana 3

Un historial de préstamos (nivel 2)

> Este ejercicio corresponde al ejercicio 8 (páginas 22 y 204) de *C++ par la pratique* (3ª edición, PPUR).

El objetivo de este ejercicio es resolver el siguiente problema:

Un banco concede un préstamo a una persona X por un importe total de **S0** euros. Esta persona devuelve una cantidad fija **r** cada mes y paga (además) un interés variable **i = ir * S**, donde **ir** es el tipo de interés mensual (fijo) y **S** es el importe pendiente de reembolso (antes de deducir la amortización mensual).

Se trata de determinar el importe de los intereses percibidos por el banco una vez reembolsado el préstamo.

Para ello, escriba el programa `pret.cc`, que calcula el importe de los intereses percibidos y el plazo de amortización en meses, y luego muestra esta información en la pantalla.

El programa también debe preguntar al usuario por los valores **S0** (estrictamente positivo), **r** (estrictamente positivo) e **ir** (estrictamente entre 0 y 1) y comprobar que son válidos.

Pruebe su programa con los valores siguientes S0=30000, r=1200, ir=0,01 (es decir, 1%). El interés total percibido (a lo largo de 25 meses) es entonces de 3900 euros.

Suite y serie (nivel 2)

a) Escribe un programa que calcule los 10 primeros términos de la sucesión U_n tal que:

$$U_0 = 1, \quad U_{n+1} = \frac{A}{n+1}$$

Tienes que averiguarlo:

$U_0 = 1$
$U_1 = 1$
$U_2 = 0.5$
$U_3 = 0.1666667$
$U_4 = 0.0416667$
$U_5 = 0.0083333$
$U_6 = 0.0013889$
$U_7 = 0,000198413$
$U_8 = 2,48016e\text{-}5$ U_9
$= 2,75573e\text{-}06$
$U_{10} = 2.75573e\text{-}07$

b) Modifica tu programa para que calcule simultáneamente la sucesión U_n y la serie V_n, donde

$$V_n = \sum_{i=0}^{n} U_i \cdot$$

Comprueba que V_n converge a $e = \exp(1) = 2,71828...$

Cifras en modo texto (nivel 1)

a) Escribe un programa que muestre los valores del 1 al 9 en una línea, utilizando un bucle `for`: `123456789`

b) Modifica el programa para que muestre 9 líneas similares, utilizando 2 bucles `for`:

```
123456789
123456789
.
.
.
123456789
```

c) ¿Cómo puedo cambiar el programa para que muestre un triángulo?

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

d) Modifica tu programa una última vez para que muestre una pirámide invertida:

```
        1
       12
      123
     1234
    12345
   123456
  1234567
 12345678
123456789
```

Triángulo (nivel 2)

Utilizar bucles para construir un triángulo isósceles formado por el carácter estrella (*).
Mostrar *n* líneas, donde *n* es introducido en el teclado por el usuario.

Ejemplo: para *n* = 5 :

```
    *
   ***
  *****
 *******
*********
```

Cálculo del PGDC (algoritmo de Euclides, nivel 1)

> Este ejercicio corresponde al ejercicio 38 (páginas 90 y 272) de *C++ par la pratique* (3ª edición, PPUR).

(PGDC = máximo común divisor)

Objetivos

Escribe el programa `pgdc.cc` que :

1. pide al usuario que introduzca dos números enteros estrictamente positivos `a` y `b`;
2. comprueba si `a` y `b` son estrictamente positivos, y vuelve a preguntar al usuario si no lo son.
3. encontrar los números enteros `u`, `v` y `p` que satisfacen la identidad de Bezout (es decir, una ecuación de valor entero): `u a + v b = p`, donde `p` es el máximo común divisor de `a` y `b`.

Método

El método utilizado es el **algoritmo de Euclides**.

Procederemos por iteración, del siguiente modo (señalando `x / y` como el cociente y `x % y` como el resto de la división entera de `x` por `y`):

0: inicialización			$u_0 = 1$	$v_0 = 0$
	$x_1 = a$	$y_1 = b$	$u_1 = 0$	$v_1 = 1$
	...	...	...	...
i+1: iteración	$x_{i+1} = y_i$	$y_{i+1} = x_i \% y_i$	$u_{i+1} = u_{i-1} - u_i(x_i / y_i)$	$v_{i+1} = v_{i-1} - v_i(x_i / y_i)$
	...	...	...	...
Valores finales	x_{k-1}	$y_{k-1} = 0$	u_{k-1}	v_{k-1}
k: condición de parada cuando $y_k = 0$	**p =** no es necesario x_k		$y_k = 0$	no es necesario

En otras palabras, calcularemos los valores de `x`, `y`, `u` y `v` paso a paso. Cada vez, calcularemos los nuevos valores en función de los anteriores (y procuraremos memorizar lo necesario para un cálculo correcto, véanse las instrucciones más abajo).

Por ejemplo, $y_{i+1} = x_i \% y_i$ significa: "el nuevo valor de `y` vale lo mismo que el antiguo valor de `x`".

modulo el valor antiguo de `y`".

Programe estos cálculos en un bucle, que se ejecuta hasta que se cumple la condición de parada.

Recuerda inicializar correctamente tus variables antes de entrar en el bucle.

Indicaciones

Dadas las dependencias entre los cálculos, deberá definir (por ejemplo) las variables : `x`, `y`, `u`, `v` **y** `q=x/y`, `r=x%y`, `prev_u`, `prev_v`, `new_u` y `new_v`.

Actualizará estas variables en cada iteración, utilizando las fórmulas de la línea *i+1* y las relaciones temporales obvias entre ellas (por ejemplo `prev_u = u`).

Comprueba que `y` es distinto de cero antes de dividir.

Ejemplo de ejecución

```
Introduzca un número entero mayor o igual que 1: 654321
Introduzca un número entero mayor o igual que 1: 210
Calcule el PGDC de 654321 y 210

  x         y         u         v
210       171         1     -3115
171        39        -1      3116
 39        15         5    -15579
 15         9       -11     34274
  9         6        16    -49853
  6         3       -27     84127
  3         0        70   -218107
PGDC(654321, 210)=3
```

Nota

- Tenga en cuenta que sólo para el cálculo de la PGDC, el cálculo de `x` e `y` por el algoritmo anterior es suficiente, no hay necesidad de `u` y `v`. Se introdujeron aquí para encontrar la ecuación de Bezout (y para que programes secuencias anidadas). Por ejemplo, en el ejemplo anterior tenemos :
  -27 * 654321 + 84127 * 210 = 3.
