MOOC Init. Prog. C++
Exercices supplémentaires facultatifs semaine 3

Une histoire de prêt (niveau 2)

> Cet exercice correspond à l'exercice n°8 (pages 22 et 204) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

L'objectif de cet exercice est de résoudre le problème suivant :

Une banque fait un prêt à une personne X pour un montant total de **S0** euros. à une personne X. Cette personne rembourse chaque mois un montant fixe **r** et paye (en plus) un intérêt variable **i = ir * S**, où **ir** est le taux d'intérêt mensuel (fixe) et **S** la somme restant à rembourser (avant déduction du remboursement mensuel).

Le but est de déterminer la somme des intérêts encaissés par la banque une fois le prêt remboursé.

Écrivez pour cela le programme `pret.cc` qui calcule la somme des intérêts encaissés et la durée en mois du remboursement, puis qui affiche ces informations à l'écran.

Le programme doit en outre demander à l'utilisateur les valeurs **S0** (strictement positif), **r** (strictement positif) et **ir** (compris strictement entre 0 et 1) et s'assurer de leur validité.

Testez votre programme avec les valeurs suivantes : S0=30000, r=1200, ir=0.01 (i.e. 1%). La somme des intérêts encaissés (sur 25 mois) est alors de 3900 euros.

Suite et série (niveau 2)

a) Écrivez un programme qui calcule les 10 premiers termes de la suite U_n telle que:

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

Vous devez trouver:

$U_0 = 1$
$U_1 = 1$
$U_2 = 0.5$
$U_3 = 0.1666667$
$U_4 = 0.0416667$
$U_5 = 0.0083333$
$U_6 = 0.0013889$
$U_7 = 0.000198413$
$U_8 = 2.48016e\text{-}5$
$U_9 = 2.75573e\text{-}06$
$U_{10} = 2.75573e\text{-}07$

b) Modifiez votre programme pour qu'il calcule simultanément la suite U_n et la série V_n, où

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

Vérifiez que V_n converge vers $e = \exp(1) = 2.71828...$

Figures en mode texte (niveau 1)

a) Ecrivez un programme qui affiche les valeurs 1 à 9 en ligne, à l'aide d'une boucle `for`:

```
123456789
```

b) Modifiez le programme pour qu'il affiche 9 lignes similaires, à l'aide de 2 boucles `for`:

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

c) Comment modifier le programme pour qu'il affiche un triangle ?

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

d) Modifiez une dernière fois votre programme, pour qu'il affiche une pyramide inversée:

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

Triangle (niveau 2)

Utilisez des boucles afin de construire un triangle isocèle formé par le caractère étoile (*). Affichez-en *n* lignes, où *n* sera entré au clavier par l'utilisateur.

Exemple: pour *n* = 5 :

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

Calcul de PGDC (algorithme d'Euclide, niveau 1)

> Cet exercice correspond à l'exercice n°38 (pages 90 et 272)
> de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

(PGDC = plus grand diviseur commun)

Buts

Écrivez le programme `pgdc.cc` qui :

1. demande à l'utilisateur d'entrer deux entiers strictement positifs `a` et `b`;
2. teste si `a` et `b` sont bien strictement positifs, et dans le cas contraire les redemande à l'utilisateur.
3. trouve les entiers `u`, `v` et `p` satisfaisant l'identité de Bezout (i.e. une équation à valeurs entières) : `u a + v b = p`, avec `p` le plus grand commun diviseur de `a` et `b`.

Méthode

La méthode utilisée est **l'algorithme d'Euclide**.

On procédera par itération, comme suit (en notant `x / y` le quotient et `x % y` le reste de la division entière de `x` par `y`) :

0 : initialisation			$u_0 = 1$	$v_0 = 0$
	$x_1 = a$	$y_1 = b$	$u_1 = 0$	$v_1 = 1$
	...	...	...	...
i+1 : itération	$x_{i+1} = y_i$	$y_{i+1} = x_i \% y_i$	$u_{i+1} = u_{i-1} - u_i(x_i / y_i)$	$v_{i+1} = v_{i-1} - v_i(x_i / y_i)$
	...	...	...	...
Valeurs finales	x_{k-1}	y_{k-1} **!= 0**	u_{k-1}	v_{k-1}
k : condition d'arrêt quand $y_k = 0$	**$p = x_k$**	$y_k = 0$	inutile	inutile

C'est-à-dire que l'on va calculer de proche en proche les valeurs de `x`, `y`, `u` et `v`. En calculant à chaque fois les nouvelles valeurs en fonction des anciennes (et en faisant bien attention à mémoriser ce qui est nécessaire à un calcul correct, voir les indications ci-dessous).

Par exemple, $y_{i+1} = x_i \% y_i$ veut dire : "la nouvelle valeur de `y` vaut l'ancienne valeur de `x` modulo l'ancienne valeur de `y`".

Programmez ces calculs dans une boucle, qui s'execute tant que la condition d'arrêt n'est pas vérifiée.

Pensez à initialiser correctement vos variables avant d'entrer dans la boucle.

Indications

Vu les dépendances entre les calculs, vous aurez besoin de définir (par exemple) les variables : `x`, `y`, `u`, `v` **et** `q=x/y`, `r=x%y`, `prev_u`, `prev_v`, `new_u` et `new_v`.

Vous mettrez ces variables à jour à chaque itération, à l'aide des formules de la ligne *i+1* et des relations temporelles évidentes entre elle (par exemple `prev_u = u`).

Testez si `y` est non nul avant d'effectuer les divisions !

Exemple d'execution

```
Entrez un nombre entier supérieur ou égal à 1 : 654321
Entrez un nombre entier supérieur ou égal à 1 : 210
Calcul du PGDC de 654321 et 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
```

Note

- Remarquez que pour le seul calcul du PGDC, le calcul de `x` et `y` par l'algorithme ci-dessus suffit, pas besoin de `u` et `v`. Ils ont été introduits ici pour trouver l'équation de Bezout (et vous faire programmer des suites imbriquées). Par exemple sur l'exemple précédent on a :
  -27 * 654321 + 84127 * 210 = 3.
