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

> Exercice n°38 (pages 90 et 272) de l'ouvrage *C++ par la pratique*.
> (pages 92 et 272 dans la 2e édition).

(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
```

Notes

1. Ne vous préoccupez pas de la forme de l'affichage ci-dessus ; faites au plus simple avec vos connaissances actuelles (on ne vera que en semaine 11 comment faire ce joli affichage bien aligné) ;
2. Vous pouvez réutiliser la fonction `demander_nombre()` de l'exercice 11 de la semaine 4 du MOOC (semaine 5 d'ICC).
3. 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 résoudre l'équation de Bezout (et vous faire programmer des suites imbriquées). Par exemple sur l'exemple précédent on a :
   -27 * 654321 + 84127 * 210 = 3.

Dernière mise à jour le 8 septembre 2021
Last modified: Wed Sep 8, 2021