Colle

Pi

Responsable Astek astek_resp@epitech.eu

Table des matières

Consignes

- Le sujet peut changer jusqu'à une heure avant le rendu.
- Vos exercices doivent être à la norme.
- Votre programme devra compiler avec un Makefile.
- Travaillez en local !
  C'est à dire que pour chaque exercice vous devez le compiler sur votre compte linux puis, une fois qu'il fonctionne, le copier sur votre compte AFS.
  Ceci dans le simple but de ne pas surcharger les serveurs car vous êtes nombreux.
- Le binaire s'appellera `pi`
- Vous avec le droit d'utiliser les fonctions `malloc`, `free` , `exit`, `write`, `read`
- Vous n'avez pas le droit à votre `libmy`
- Vous n'avez pas le droit à internet
- Rendu sur le compte du chef de groupe :
  `/afs/epitech.net/users/group/login/rendu/cpe/colle/pi`

`Attention aux droits de vos fichiers et de vos répertoires`

Sujet

L'objectif de cette colle est de calculer le nombre PI avec une expression très longtemps utilisée.
L'opération que nous allons réaliser est la suivante :

Pi = 4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + 4/13 - 4/15...

Cependant cette suite converge très doucement. Il sera alors nécessaire de faire un calcul avec une très grande profondeur (au moins 300 termes) pour avoir un résultat cohérent.

Il est évident qu'à la main le calcul sera très long et fastidieux. Mais en C, avec vos machines, seules quelques secondes suffiront.

Il ne reste plus qu'à programmer tout ça.

1. Le programme et ses paramètres
   `-p [profondeur]` : Par défaut 300
   `-P [précision]` : Nombre de chiffres après la virgule, par défaut 20
   `-a [nb_1] [nb_2]` : réalise l'addition de deux nombres positifs passés sous forme de chaine de caractère
   `-s [nb_1] [nb_2]` : réalise la soustraction de deux nombres passés sous forme de chaine de caractères avec nb_1 toujours supérieur à nb_2
   `-d [nb_1] [nb_2]` : réalise la division euclidienne entre deux nombres positifs

2. L'addition
   Réaliser l'addition de deux nombres positifs.
   Il sera possible de stipuler la précision de l'opération pour pouvoir obtenir une infinité de nombre après la virgule
   ex : ./pi -P 10 -a "2,1111111111111" "10,0001"
   retournera : résultat : 12,1112111111

3. La soustraction
   Réaliser la soustraction de deux nombres positifs.
   Il sera possible de préciser la précision de l'opération pour pouvoir obtenir une infinité de nombre après la virgule.
   ex : ./pi -P 5 -s "401,15" "0,0001"
   retournera : résultat : 401,14990

   
   Indices `A bien y refléchir, je me demande si je n'ai pas déja presque fait la soustraction quand j'arrive à cette étape...`

4. La division
   Réaliser la division euclidienne de deux entiers naturels.
   Il sera possible de préciser la précision de l'opération pour pouvoir obtenir une infinité de nombre après la virgule.

Rappel :
- C'est la division que l'on fait au primaire
- Deux entiers naturels a et b, avec b non nul, la division euclidienne associe un quotient q et un reste r, tous deux entiers naturels, vérifiant :
a = b.q+r avec r < b.

- Algorithme donné à titre indicatif

```
1  division := proc(a,b)
2    local r, q, u ;
3    r := a ;
4    q := 0 ;
5    while (r >= b)
6       do
7          r := r -- b ;
8          q := q + 1 ;
9       od ;
10   u := [q, r] ;
11   return(u) ;
12 end ;
```

Indices Quand le reste est plus petit que le diviseur, il suffit de le multiplier par 10, de rajouter une virgule au dividende pour poursuivre la division euclidienne. On pourra multiplier alors par 10 indéfiniement le reste obtenu pour poursuivre la division.

5. PI
Réaliser le calcul de PI.

ex : ./pi -p 600 -P 100
résultat : 3,14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 79

6. Bonus :
   - Gérer les opérations avec des nombres négatifs
   - Controler la redondance du nombre avec un paramètre `-r [nb_element]`
     ex : 1,332334 redondance de 2 éléments donne 2. // redondance de 3 éléments donne 0.
     ex : 1,123123 redondance avec 3 éléments donne 2.
   - Générer un calcul de PI avec une profondeur maximale, SANS REDONDANCE, en moins de 3 secondes.