**Travaux pratiques 6**

**Exercice 1 : calculatrice basique**

**Notre objectif est de réaliser une calculatrice basique pouvant calculer une somme, une soustraction, une multiplication, une division, le reste d’une division entière, une puissance, une factorielle, le PGCD et le PPCD.**

**Celle-ci attendra une entrée formatée suivant la** [**notation polonaise inverse**](http://fr.wikipedia.org/wiki/Notation_polonaise_inverse)**. Autrement dit, les opérandes d’une opération seront entrés *avant* l’opérateur, par exemple comme ceci pour la somme de quatre et cinq : 4 5 +.**

4+5 infixée 4 5 + post fixé enregistrer 4; enregistrer 5 plus additionner

**Elle devra également retenir le résultat de l’opération précédente et déduire l’utilisation de celui-ci en cas d’omission d’une opérande. Plus précisément, si l’utilisateur entre par exemple 5 +, vous devrez déduire que la première opérande de la somme est le résultat de l’opération précédente (ou zéro s’il n’y en a pas encore eu).**

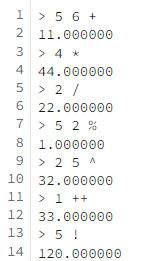
**Chaque opération sera identifiée par un symbole ou une lettre, comme suit :**

* **addition : + ;**
* **soustraction : - ;**
* **multiplication : \* ;**
* **division : / ;**
* **reste de la division entière : % ;**
* **puissance : ^ ;**
* **factorielle : ! ;**
* **PGCD : g ;**
* **PPCD : p.**

ou bien une constante numérique #define p = 10;

**Le programme doit s’arrêter lorsque la lettre « q » est spécifiée comme opération (avec ou sans opérande).**

Exemple d’exécution



autre exemple

30 12 pgcd 2 ^ 5% 70 5 + q

stocker 30 dans a

stocker 12 dans b

stocker 6 dans a

stocker 2 dans b

a exp b = 36

stocker 36 dans a

stocker 5 dans b

36 %5 dans a

stocker 1 dans a

##### stocker 70 dans b

stocker 5 dans a

additionner a et b dans a

stocker 75 dans a

afficher 75 (dernier résultat)

**Indices.**

PPCM (a, b) = (axb) / PGCD (a, b)

vous pouvez utiliser une boucle infinie et l’interrompre par exit

While (1)

{

switch(op)

{

case ‘+’:

…….

break;

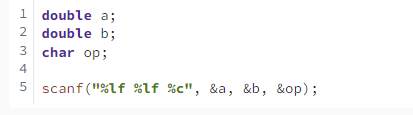
case ‘q’: exit

}

}

##### Les puissances

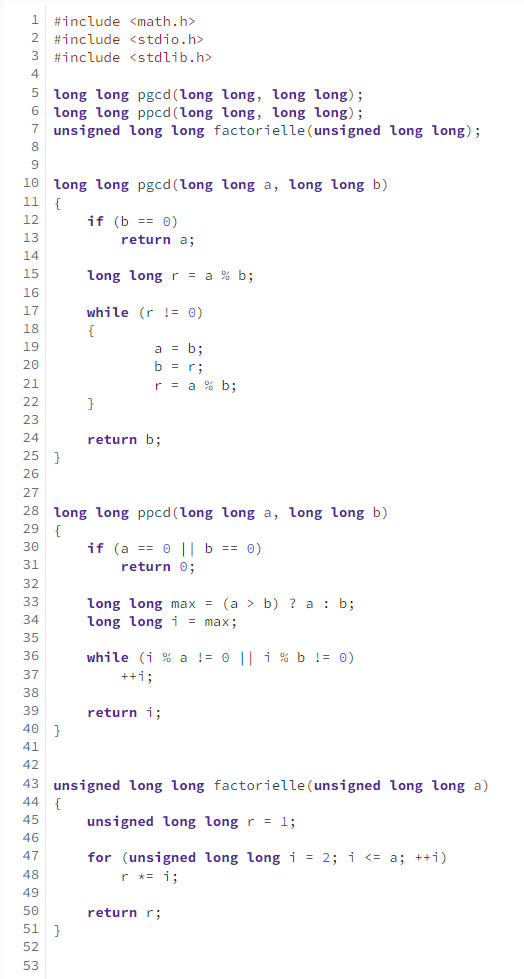
Décomposer en fonctions le plus que possible (PGCD, PPCM, Fact, …)

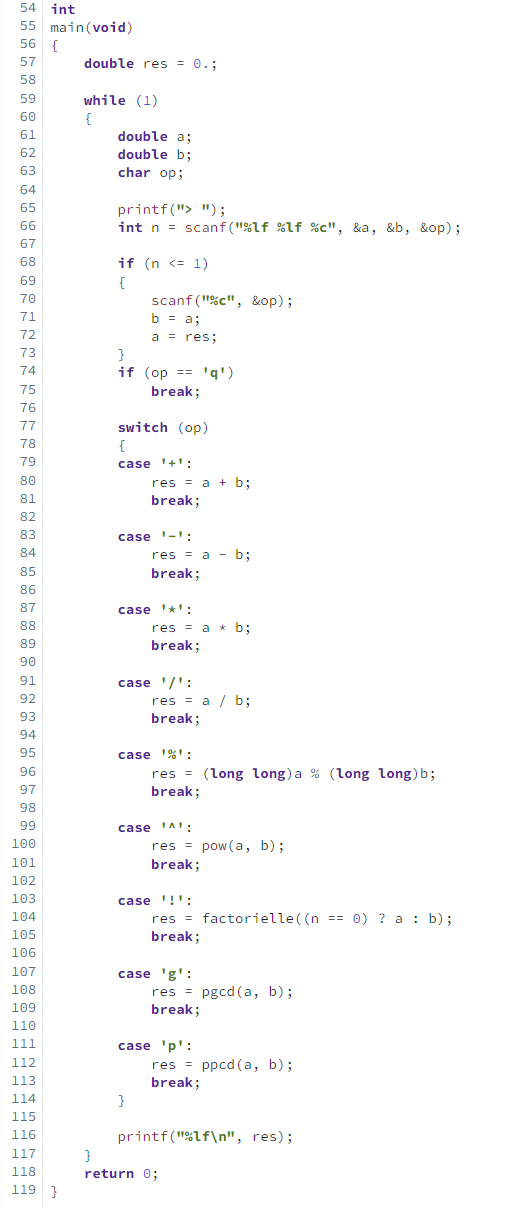


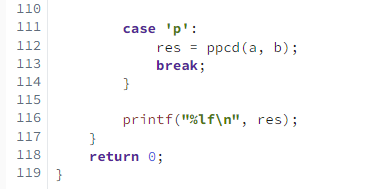
Dans le code ci-dessus, si l’utilisateur entre 7 \*, la fonction scanf() retournera 1 et n’aura lu que le nombre 7. Il sera nécessaire de l’appeler une seconde fois pour que le symbole \* soit récupéré.

Utiliser le switch case of pour choisir une opération à faire

Corrigé.







Commençons par la fonction main(). Nous définissons plusieurs variables :

* res, qui correspond au résultat de la dernière opération réalisée (ou zéro s’il n’y en a pas encore eu) ;
* a et b, qui représentent les éventuels opérandes fournis ;
* op, qui retient l’opération demandée ; et
* n, qui est utilisée pour retenir le retour de la fonction scanf().

Ensuite, nous entrons dans une boucle infinie (la condition étant toujours vraie puisque valant un) où nous demandons à l’utilisateur d’entrer l’opération à réaliser et les éventuelles opérandes. Nous vérifions ensuite si un seul opérande est fourni ou aucun (ce qui se déduit, respectivement, d’un retour de la fonction scanf() valant un ou zéro). Si c’est le cas, nous appelons une seconde fois scanf() pour récupérer l’opérateur. Puis, la valeur de a est attribuée à b et la valeur de res à a.

Si l’opérateur utilisé est q, alors nous quittons la boucle et par la même occasion le programme. Notez que nous n’avons pas pu effectuer cette vérification dans le corps de l’instruction switch qui suit puisque l’instruction break nous aurait fait quitter celui-ci et non la boucle.

Enfin, nous réalisons l’opération demandée au sein de l’instruction switch, nous stockons le résultat dans la variable res et l’affichons. Remarquez que l’utilisation de conversions explicites n’a été nécessaire que pour le calcul du reste de la division entière. En effet, dans les autres cas (par exemple lors de l’affectation à la variable res)

**Exercice 2 : Utilisation d’une pile**

Développer une pile (structure contiguë - Tableau) avec ses opérations (empiler, dépiler, pile pleine, pile vide et initialisation ) et implémenter le calcul arithmétique en utilisant la structure pile définie.

**Exercice 3 : Petites coupures**

##### En petites coupures ?

**Pour ce dernier exercice, vous allez devoir réaliser un programme qui reçoit en entrée une somme d’argent et donne en sortie la plus petite quantité de coupures nécessaires pour reconstituer cette somme.**

**Pour cet exercice, vous utiliserez les coupures suivantes :**

* **des billets de 100€ ;**
* **des billets de 50€ ;**
* **des billets de 20€ ;**
* **des billets de 10€ ;**
* **des billets de 5€ ;**
* **des pièces de 2€ ;**
* **des pièces de 1€ ;**

**Ci dessous un exemple de ce que devra donner votre programme une fois terminé.**

**Entrez une somme : 285**

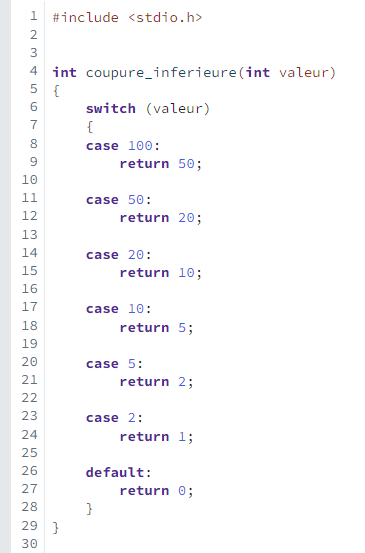
**2 billet(s) de 100.**

**1 billet(s) de 50.**

**1 billet(s) de 20.**

**1 billet(s) de 10.**

**1 billet(s) de 5.**

****