Exercice 13

OBJECTIF : FONCTIONS ET STRUCTURES

A l'issue de la réalisation de cet exercice, les étudiants doivent être capables de réaliser des fonctions capables de recevoir et retourner plusieurs valeurs en utilisant des structures.

PREPARATION DU PROJET

L'exercice 2 sert de mode d'emploi pour la création du projet.

- Lancez le "Microsoft Visual Studio 2015
- Créez un nouveau projet :
  - Projets Visual C++
  - Application console Win32 (Dans la fenêtre modèle)
  - Sous emplacement: introduisez votre répertoire d'exercices
  - Décocher "Créer le répertoire pour la solution"
  - Sous Nom du projet : **Ex13**
- Paramètre de l'application :
  - Application Console
  - Projet Vide
- **Copie du fichier Ex13.c :**
  Il faut copier le fichier Ex13.C de K:\ES\Maitres-Eleves\SLO\Modules\SL124_LOGA\Exercices\Ex13 dans le répertoire Ex13. (**Usage de l'explorateur Windows**)
- Ajoutez au projet le fichier Ex13.c :
  - Depuis le Visual Studio 2015, sélectionnez "**Fichiers Sources**", avec un clique-droit obtenez le menu pour Ajouter un élément existant.
- Test de compilation : vous devez obtenir :

```
Génération : 1 a réussi, 0 a échoué, 0 a été ignore
```

DONNEES DU PROBLEME

Le fichier fournis correspond à la solution de l'Ex11. Il s'agit de transformer les fonctions en utilisant des structures, conformément à la demande, pour chaque test.

ACTION DU TEST A

Il faut réaliser la saisie de 2 valeurs de type int, la première est le dividende, le deuxième le diviseur.

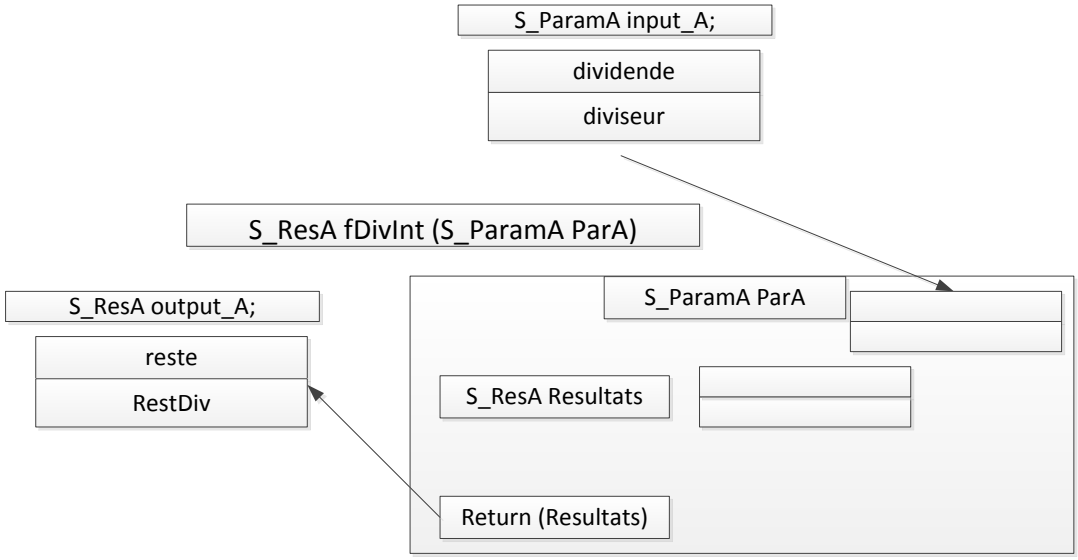
Si le diviseur = 0, il ne faut pas appeler la fonction, par contre il faut afficher :

TestA : erreur division par 0 !

Il faut appeler la fonction **fDivInt,** ayant comme paramètres une structure du type **S_ParamA** comportant 2 champs de type int, le numérateur et le diviseur.

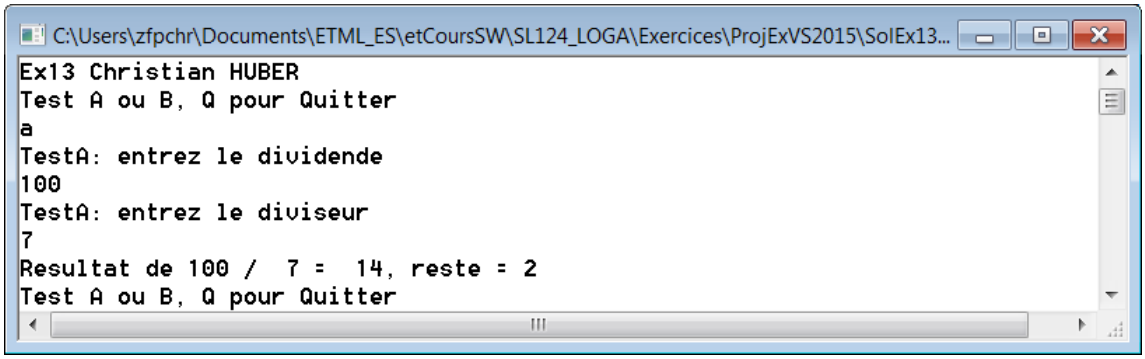
La fonction retourne une structure du type **S_ResA** comportant 2 champs de type int, le reste et le résultat de la division entière.

La fonction et son environnement :



Après l'appel de la fonction **fDivInt,** il faut afficher sur 1 ligne:
Resultat de a / b = y, reste = r
(a représente le dividende, b le diviseur, y le résultat et r le reste)

Exemple de résultat :



ACTION DU TEST B

La partie fournie par le fichier de canevas affiche : Test B : Entrez un nombre de 0 à 999. La réponse de l'utilisateur est stockée dans la variable int ValB.
Si la valeur de ValB est > 999 alors on utilise 999 et on affiche: "ValB limitée a 999 !",

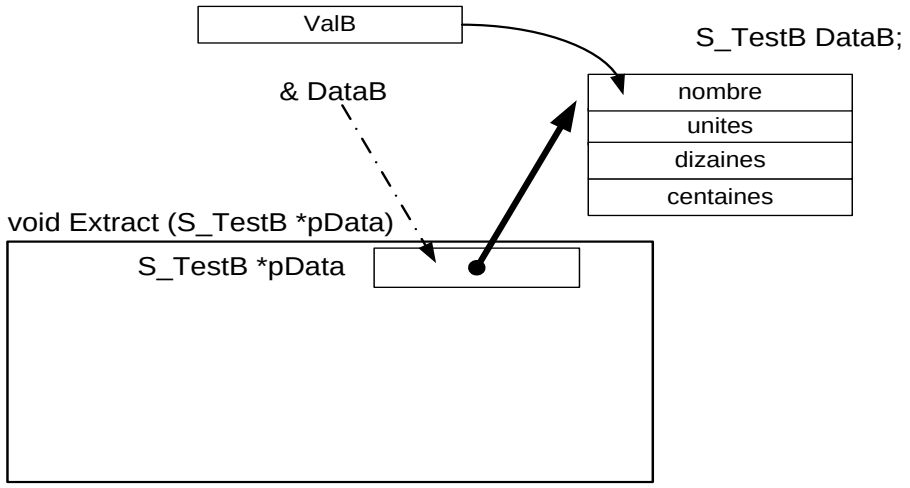
L'objectif du testB est d'extraire le chiffre des centaines, le chiffre des dizaines et le chiffre des unités du nombre stocké dans ValB. Par exemple 851 doit donner 8 centaines, 5 dizaines et 1 unité.

L'extraction ce fait en utilisant une seule fonction **Extract**

Cette fonction n'a qu' un paramètre qui est un pointeur sur une structure du type S_TestB. La structure du type S_TesB comporte 4 champs.

int nombre (nombre reçu)
short int unites
short int dizaines
short int centaines

Avant l'appel de la fonction Extract, il faut placer ValB dans le champ nombre de la structure. Lors de l'appel il faut fournir l'adresse de la structure !.



Après l'appel de la fonction Extract, la structure est mise à jour et contient le chiffre des unités, des dizaines et des centaines.

Il faut afficher : "ValB = bbb centaine = c dizaine = d unite = u"
Avec bbb la valeur saisie, c le nombre de centaine, d le nombre de dizaine et u le nombre d'unité.

Exemple de résultat à obtenir :