Language de programmation L2

Le langage C

TD1

Visibilité des variables

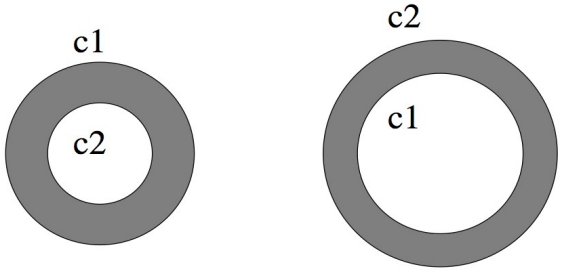
Donner le résultat du *printf* de la fonction *main.*

```
#include <stdio.h>
#define X 25
#define Y 20

int f1 (int x) {
 int y = 3;
 {
    int x = 2, y;
    y = x * x;
 }
 return x * y;
}

int f2 (int x, int y) {
  x = 3;
  {
    int x = 2;
    int y = 3;
    {
      int z = y;
      int y = x;
      {
        return (z + y + x);
      }
    }
  }
}

int main () {
  int x, y, res1, res2;
  y = 30;
  x = Y;
  res1 = f1 (y);
  res2 = f2(y, x);
  printf("%d %d %d %d\n",x,y,res1,res2);
}
```

Calcul de la surface d'une couronne

Soient 2 cercles c1 et c2 de rayons respectifs r1 et r2 et de même centre. On souhaite calculer la surface de la couronne ayant pour limite extérieure le cercle de plus grand rayon et pour limite intérieure le cercle de plus petit rayon (voir figure 1). En plus des opérateurs arithmétiques, vous pouvez utiliser la fonction fabs qui permet de calculer la valeur absolue d'un réel, et dont l'entête est : double fabs(double x). Pour toutes les questions suivantes vous utiliserez uniquement des variables de type double.

1. Ecrire une fonction *scercle* qui prend en paramètre le rayon d'un cercle et retourne la surface du cercle correspondant. On suppose que le nombre pi est défini dans une variable PI qu'il n'est pas nécessaire de déclarer ni d'initialiser.
2. Ecrire une fonction *scouronne* qui retourne l'aire de la couronne telle que définie précédemment, en fonction des rayons r1 et r2, et en utilisant la fonction *scercle*. Les rayons r1 et r2 sont quelconques (r1 peut être supérieur, inférieur ou égal à r2).
3. Ecrire la fonction principale d'un programme qui utilise *scouronne* pour calculer la couronne pour 2 cercles de rayons 3,1 et 7,8 et affiche le résultat. Pour définir la valeur des rayons, vous utiliserez 2 variables initialisées correctement.



Boucles

Pour chacun des exercices suivants, choisissez quelle structure de boucle utiliser puis programmez-le.

1. Ecrire la fonction *nb_chiffres* qui retourne le nombre de chiffres d'un entier décimal donné en paramètre. Par exemple : *nb_chiffres(6789)* retourne l'entier 4.
2. Ecrire la fonction *nb_div* qui retourne le nombre de diviseurs distincts d'un entier strictement positif donné en paramètre. Par exemple : *nb_div*(10) retourne l'entier 4.
3. Ecrire la fonction *somme_chiffres* qui retourne la somme des chiffres d'un entier décimal donné en paramètre. Par exemple : *somme_chiffres*(6789) retourne l'entier 30.
4. Ecrire la fonction *nb_div_pair_impair* qui calcule le nombre de diviseurs pairs et impairs d'un entier strictement positif donné en paramètre, et affiche ces 2 nombres.

Tableaux

1. Écrire la fonction *int detecter(int val, int tab[], int t)*, qui retourne l'indice correspondant à la première case ayant la valeur *val*, dans le tableau donné en paramètre (*tab*, de taille *t*). Si la valeur n'est pas présente, on retourne -1 par défaut.
2. Ecrire la fonction *int compare (int tab1[], int tab2[], int t)* qui compare les contenus de 2 tableaux (*tab1* et *tab2*) de taille *t*. Si il y a une différence entre les 2 tableaux, la fonction retourne la valeur faux et vrai sinon.

Palindrome

1. Écrire la fonction *taille* qui prend en paramètre une chaîne de caractère et retourne la taille de la chaîne.
2. Écrire la fonction *palindrome* qui prend en paramètre une chaîne de caractère et retourne vrai si cette chaîne de caractères est un palindrome (en utilisant la fonction taille) .