TP 4 – nombres à virgule flottante

1. **Qu'affiche** le programme (à retrouver dans l'archive) ci-dessous ? Pourquoi ?

```
/**
 * @file ex1.c
 * @author : Philippe Lefebvre <philippe.lefebvre@ensicaen.fr>
 * @date : 2020/12/25
 * @brief : Programme de démonstration sur les nombres flottants
 * @version : 1.0
**/

#include<stdio.h>
int main (void) {
    float a=0.2, b=1e18, c=a/b, d= 3/7, e = 3/7.0 ;
    printf ("a = %f, b = %f, b = %e\n", a, b, b) ;
    printf ("a/b = %.20f, 3/7 = %f, 3/7.0 = %f \n", c, d, e) ;
    return(0) ;
}
```

2. **Tester** le programme suivant (à retrouver dans l'archive) et expliquez son résultat en vous appuyant sur le format IEEE-754.

```
/**
 * @file ex2.c
 * @author : Philippe Lefebvre <philippe.lefebvre@ensicaen.fr>
 * @date : 2020/12/25
 * @brief : Programme de démonstration sur la précision du type float
 * @version : 1.0
**/

#include <stdio.h>
int main(void){
    int n=0;
    float epsilon=1.0;
    float condition ;
    do {
        epsilon/=10.0;
        n++;
        condition = 1.0+epsilon ;
    } while(condition!=1.0);
    printf("epsilon = 10^-%d\n",n);
    return 0;
}
```

3. **Déclarez** la variable `condition` de type double. Que vaut n ?

4. La suite de Fibonacci est définie par la relation de récurrence suivante : $u_0=u_1=1$; $u_{n+2}=u_{n+1}+u_n$, pour tout entier naturel n. Voici le début de cette suite 1 1 2 3 5 8 13 21 34 55 89 *–correction disponible–*

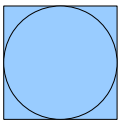
   Écrire en C des programmes n'utilisant pas de fonctions **récursives*** et répondant à chacune des situations ci-après :

   a) Demander un entier à l'utilisateur, initialiser une variable n à la valeur saisie et afficher les valeurs des n+1 premiers termes u_k ($0 \le k \le n$) correspondants de la suite de Fibonacci. A partir de quelle valeur de n le type int devient-il insuffisant ?

   b) Calculer le nombre d'or en tant que limite de la suite v_n définie par $v_n = u_n/u_{n-1}$ ($n \ge 1$) : les

valeurs de v_n seront affichées jusqu'à ce que l'écart relatif entre deux termes consécutifs devienne inférieur ou égal à la constante epsilon calculée dans l'exercice 2 ci-dessus. Comparer expérimentalement à la valeur exacte (1+√5)/2.

5. **Méthode de Monté-Carlo.** On a quelquefois besoin de recourir au hasard en informatique pour résoudre certains problèmes. On se propose ici d'utiliser la méthode de Monté-Carlo pour déterminer le nombre π. La méthode est la suivante : Imaginez une cible de cette forme *(fig. 1)* sur laquelle on lance des fléchettes au hasard :



Ceci revient à tirer 2 nombres au hasard x et y dans l'intervalle [-1,1[représentant les coordonnées d'un point. Si $x^2 + y^2 < 1$ alors le point est dans le cercle unité. La proportion de nombres dans le cercle unité est égale au rapport des surfaces du cercle unité inscrit et du carré de longueur L=2.
Ce rapport vaut : π x r^2 / L^2 = π/4.
Vous aurez besoin de la fonction `int rand ()` qui retourne un nombre entier pseudo-aléatoire compris dans [0, RAND_MAX]. Cette fonction retourne toujours la même suite de nombres. Si vous désirez tirer une autre suite de nombres, vous devez d'abord initialiser le premier terme (*seed*) de la suite au début de votre programme par la fonction `void srand( int seed )`. Une technique classique consiste à utiliser la fonction `time( NULL )` qui retourne le nombre de secondes écoulées depuis le 01/01/1970.
Écrire un programme qui donne une estimation du nombre π.

warning : implicit declaration of function srand()

Vous avez sûrement rencontré ce message. Lorsque vous utilisez des fonctions il faut que le compilateur connaisse le prototype de la fonction, c'est à dire qu'il connaisse le type des paramètres et le type de retour. Si le prototype n'est pas connu, le compilateur suppose que tout est *int..* . Pour informer le compilateur on va utiliser la directive *#include <fichier.h>.*
Dans quel *fichier.h* est incluse la fonction srand ? Pour le savoir on va utiliser la commande *man.*
Dans une console tapez « *man srand* ». Tapez « q » pour quitter le manuel.

Quelquefois man renvoie sur un synonyme. Pour connaître toutes les entrées du manuel tapez man -k. Exemple :
man -k rand
On voit ici que la fonction rand est définie dans la section 3 du manuel. Pour y avoir accès tapez :
man 3 rand

Les fichiers .h sont dans le répertoire /usr/include.
Ouvrez /usr/include/stdlib.h. Que vaut RAND_MAX ?

6. *(pour les plus rapides)* Ecrire un programme qui calcule une estimation de la constante π à l'aide de la formule de Machin :

$$\frac{\pi}{4} = 4 \times \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right), avec \arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

On calculera le développement limité de *arctan (x)* , jusqu'à ce que deux valeurs consécutives de cette somme soient égales, à la constante epsilon (de l'exercice 2) près. Comparer expérimentalement avec la valeur de la constante π nommée `M_PI` fournie dans la bibliothèque <math.h>. Vous compilerez dans ce cas de cette manière : `gcc ex4_6.c -o ex4_6.bin -lm`
Dans Geany, les options de compilations se trouvent dans « construire » puis « définir les commandes de construction». Ajoutez « -lm » à la fin de la commande Build.