Travaux dirigés
Notation en virgule flottante
Mathématiques pour l'informatique
—IMAC 2—

▸ Exercice 1. *Calculs faciles*

1. *Récupérer le programme* `calculsFaciles.cpp` *ou bien recopier les lignes suivantes sur votre éditeur puis compiler. Que constatez-vous ? Quelle est votre explication ?*

```
#include <iostream>
#include <iomanip>

int main()
{
  std::cout << std::setprecision(-1) << "1.0  + 2.0  = " << 1.0 + 2.0 << std::endl;
  std::cout << std::setprecision(20) << "1.0f + 2.0f = " << 1.0f + 2.0f << std::endl;
  std::cout << std::setprecision(20) << "1.0  + 2.0  = " << 1.0  + 2.0  << std::endl;
  std::cout << std::setprecision(20) << "1.0L + 2.0L = " << 1.0L + 2.0L << std::endl;

  std::cout << std::setprecision(-1) << "0.1  + 0.2  = " << 0.1 + 0.2 << std::endl;
  std::cout << std::setprecision(20) << "0.1f + 0.2f = " << 0.1f + 0.2f << std::endl;
  std::cout << std::setprecision(20) << "0.1  + 0.2  = " << 0.1  + 0.2  << std::endl;
  std::cout << std::setprecision(20) << "0.1L + 0.2L = " << 0.1L + 0.2L << std::endl;

  std::cout << std::endl;
  std::cout << std::setprecision(20) << "0.1f - 0.1L = " << 0.1f - 0.1L << std::endl;

  return 0;
}
```

2. *Téléchargez puis compilez le programme* `seeFloat.cpp` *sur la page de l'enseignant. Testez le avec les nombres 1,2,3 et 0.1, 0.2, 0.3 (et plein d'autres tests). Qu'observez-vous?*

3. *D'après vous, quel est le résultat de l'opération suivante ?*

```
if( (0.1 + (0.2 + 0.3) == (0.1 + 0.2) + 0.3) ) std::cout << "true" << std::endl;
else std::cout << "false" << std::endl;
```

Faites un test pour vérifier.

▸ **Exercice 2.** ***Et en C/C++ ?***

1. *Qu'est-ce que je risque en écrivant dans mon programme les lignes suivantes :*

```
float a = ... ;
float b = ... ;

if(a == b) ... ;
```

   Qu'est-ce que je devrais écrire à la place ?

2. *Suis-je en danger si j'écris :*

```
int a = ... ;
float b = a/3;
```

3. *Mon programme compile-t-il avec la ligne suivante :*

```
float a = -5.0/0;
std::cout << " a " << a << std::endl;
```

4. *Est-ce que je peux mourir si ma vie dépend de la fiabilité de ce calcul* [(1)] *:*

```
float a = powf(2.0,40);
std::cout << " a = " << a << std::endl;

int b = a;
std::cout << " b = " << b << std::endl;
```

5. *Je vais manger dès que la boucle est finie, qu'est-ce qu'on mange ?*

```
for(float a=0.0; a<10; a+=0.00000001)
  ... ;

std::cout << " à table ! "  << std::endl;
```

▸ **Exercice 3.** ***Suites***

Soit la suite définie par :

$$\begin{cases} u_0 &= \dfrac{1}{3} \\ u_{n+1} &= 4u_n - 1 \end{cases}$$

1. *calculer manuellement les 5 premiers termes.*
2. *faites un programme C++ calculant les 100 premiers termes avec des* `float` *et les 550 premiers termes avec des* `double`*.*

[(1)] question de J. Chaussard

▸ **Exercice 4.** ***Multiplication à la russe***

La multiplication à la russe $a \times b$ consiste à répéter l'opération "diviser a par 2 et multiplier b par 2 jusqu'à ce que $a = 1$", dans quel cas $a \times b = b$. Deux cas sont à considérer dans ce processus itératif : si a est pair, tout se passe bien, si a est impair, sa division par 2 génère un résidu qu'il faudra additionner au résultat final. Cette approche était utilisée sur les premières calculatrices.

Par exemple 13×320 *:*

opération	a	b	*résidu*
	13	*320*	
13/2=6 reste 1	*6*	*640*	*320*
6/2=3 reste 0	*3*	*1280*	-
3/2=1 reste 1	*1*	*2560*	*1280*

résultat : $13 \times 320 = 2560 + (320 + 1280) = 4160$

Il apparait clairement qu'il est préférable de choisir pour a la plus petite des deux valeurs à multiplier.

Coder la multiplication à la russe en C++.

▸ **Exercice 5.** ***Racine carrée***

1. *Imaginez un programme estimant la racine carrée d'un nombre à virgule flotante.*
2. *Newton a proposé une méthode pour estimer la racine carrée d'un nombre x en iterant sur la suite suivante :*
$$\begin{cases} u_0 = 1 \\ u_{n+1} = \dfrac{u_n + x/u_n}{2} \end{cases}$$
Testez cette méthode et commentez.
3. *Allez voir sur wikipedia.*