Programmation : TP 5

Juliusz Chroboczek

17 octobre 2023

Exercice 1.

1. Écrivez une fonction de prototype

   ```
   int max(int x, int y);
   ```

   qui retourne le plus grand de deux entiers. Écrivez un programme (une fonction `main`) pour tester votre fonction.
2. En utilisant la fonction écrite ci-dessus, écrivez une fonction

   ```
   int max3(int x, int y, int z);
   ```

   qui retourne le plus grand de trois entiers. Écrivez un programme pour tester votre fonction.

Exercice 2.

1. Écrivez une fonction itérative (qui ne fait pas d'appels récursifs) de prototype

   ```
   int syracuse(int n);
   ```

   qui calcule le temps de vol de la n-ième suite de Syracuse (voir TP 3). Écrivez un programme qui affiche les temps de vol des entiers de 1 à 100.
2. Écrivez un programme qui lit un entier k puis affiche un entier n tel que le temps de vol de n soit maximal parmi les temps de vol des entiers de 1 à k.

Exercice 3 (Suite de Fibonacci)**.** Leonardo di Pisa, dit Fibonacci, élève des lapins. Les lapins de Fibonacci se comportent de façon un peu particulière : sa première année, un lapin est immature, et ne se reproduit donc pas. À partir de sa deuxième année, un lapin produit chaque année un autre lapin. Il y a donc un lapin l'année zéro, un lapin la première année, deux lapins la deuxième année, trois lapins la troisième année, etc.

La *suite de Fibonacci* (f_n) est définie par la récurrence suivante :

$$\begin{aligned} f_0 &= 1; \\ f_1 &= 1; \\ f_n &= f_{n-2} + f_{n-1} && \text{pour } n \geq 2. \end{aligned}$$

1. Écrivez une fonction de prototype

```
long long int fib_r(int n);
```

qui calcule f_n en faisant une suite d'appels récursifs. Écrivez un programme pour tester votre fonction (vous pouvez afficher une valeur de type « `long long` » à l'aide du descripteur de format « `%lld` »).

2. Ajoutez un appel à `printf` au corps de la fonction `fib_r` et comptez combien de fois elle est appelée lors d'un appel à `fib_r(3)`, `fib_r(4)`, `fib_r(5)`. Que constatez-vous ?
3. Écrivez une fonction

```
long long int fib_i(int n);
```

qui calcule f_n itérativement en espace constant (sans utiliser de tableaux ou faire d'appels récursifs). Écrivez un programme pour tester votre fonction.

4. Enlevez l'appel à `printf` ajouté à la fonction `fib_r` puis recompilez vos programmes à l'aide de la commande « `gcc -Wall -O2` ». À l'aide de la commande `time` comparez le temps d'exécution des deux programmes. Que constatez-vous ? Explication ?

Exercice 4. Une *fraction* est une paire d'entiers (n, d) $(d \neq 0)$ qui représente le rationnel n/d. On dit que deux fractions sont *équivalentes* si elles représentent le même rationnel. On dit qu'une fraction (n, d) est sous *forme canonique* si $d > 0$ et $\text{pgcd}(n, d) = 1$. Dans cet exercice, nous représenterons les fractions par des structures :

```
struct fract {
    long long int num, dem;
}
```

1. Écrivez une fonction

```
void print(struct fract x)
```

qui affiche une fraction.

2. Écrivez une fonction

```
int pgcd(long long int a, long long int b);
```

qui retourne le PGCD de deux entiers en utilisant l'algorithme d'Euclide. Testez votre fonction.

3. Écrivez une fonction

```
struct fract canonique(struct fract x);
```

qui retourne une fraction canonique équivalente à `x`. Testez votre fonction.

4. Écrivez une fonction

```
struct fract add(struct fract x, struct fract y);
```

qui ajoute deux fractions et retourne un résultat canonique. Testez votre fonction.

5. Écrivez une fonction

```
struct fract div(struct fract x, struct fract y);
```

qui divise x par y et retourne un résultat canonique.

6. Pour tout rationnel a, on définit la fonction

$$f^{(a)}(x) = \frac{1}{2}(x + \frac{a}{x}).$$

Pour quelles valeurs de a la fonction $f^{(a)}$ admet-elle un point fixe dans **Q** ? Dans **R** ?

7. Pour tout rationnel a, on définit la suite $(x_n^{(a)})$:

$$\begin{aligned} x_0^{(a)} &= a \\ x_n^{(a)} &= f^{(a)}(x_{n-1}) \qquad\qquad (n > 0) \end{aligned}$$

Écrivez un programme qui lit une fraction représentant a puis affiche la liste des 20 premiers termes de $(x^{(a)})$ sous forme de fractions canoniques ainsi que leurs approximations en virgule flottante [1].

Exercice 5. Un étudiant place 1 zł dans une banque. Cette somme sera rémunérée au taux annuel de 100%, l'étudiant se retrouvera donc en possession de 2 zł au bout d'une année. Un deuxième étudiant choisit de placer son złoty dans une banque lui offrant un taux de rémunération de 50% tous les six mois. Ce dernier se retrouvera en possession de 2, 25 zł à la fin de l'année.

Écrivez une fonction qui calcule la valeur e_n que possède au bout d'une année l'étudiant n, qui à placé son złoty dans une banque lui offrant un taux de rémunération de $1/n$ toutes les $1/n$ années. Écrivez un programme qui affiche les valeurs de e_n pour n allant de 1 à 100. Devinez la limite de la suite (e_n) ; sa convergence vous semble-t-elle rapide ?

1. On appelle cet algorithme la *méthode babylonienne* de calcul des racines carrées, et c'est une application immédiate de la méthode de Newton-Raphson.