TPs: Nombres et calculs en général

Van Oudenhove Didier

11 novembre 2023

Première partie

Exercices divers

1 Faire la somme des nombres compris dans un intervalle donné:

Dans le cadre du cours, nous avons réalisé l'exercice qui consistait à faire la somme des nombres entiers de 0 à n soit:

$$s = \sum_{i=0}^{n} i$$

Nous l'avions réalisé de 2 façons différentes:

1. la première façon consistait à utiliser une boucle pour :$s \leftarrow 1 \; pour \, i : 2 \rightarrow n \, faire \, s \leftarrow s + i$
2. la deuxième façon était plus intéressante puisque la performance de dépendait pas de n: $s = \frac{n^2-n}{2} + n = \frac{n*(n+1)}{2}$

Algorithme 1 Version1 de la somme de 1 à n

```
fonction calculSomme(n)
    s ← 1
    pour i : 2 → n faire
        s ← s + i
    fpour
    retour s
ffonction
```

Hyp: n est un nombre entier >=1

Fonction: long calculSomme(n)

- IN: n le nombre
- OUT: la somme des entiers de 1 à n

Variables:

- s: un entier; la somme des nombres
- i: un entier; pour aller de 1 à n

1.1 Réalisez une fonction qui calcule la somme des nombres de n à m

Dans cet exercice, on vous demande de réaliser un programme similaire à celui vu au cours mais pour une somme de n à m:

$$s = \sum_{i=n}^{m} i$$

Hypothèses: $n, m \in \mathbb{Z} \mid n <= m$

Fonction: long calculSomme(int n,int m)

In	n et m les deux entiers où n<=m
Out	un grand entier qui contiendra la somme

Algorithme: vous pouvez envisager les 2 versions:

- ☛ version qui utilise une boucle « pour » mais pensez à optimiser !!!
- ☛ version à partir d'une formule mathématique

1.1.1 Implémentation

Après avoir écrit les pseudo-codes, implémentez les dans les 2 langages:

Java: Réalisez la fonction en Java dans votre classe « MyMath » et testez la avec un test unitaire dans « TestMyMath ».

Python Réalisez la fonction en Python dans le module « mymath » et le test unitaire dans le module « test_mymath ».

2 Calculs liés à la suite de Fibonacci

2.1 Calculez le $n^{ème}$ nombre de Fibonacci

La suite de Fibonacci doit certainement vous dire quelque chose: 0, 1, 1, 2, 3, 5, 8,... le nombre suivant est calculé à partir de la somme des 2 nombres précédents:

$$F_n = \begin{cases} 0 & , n = 0 \\ 1 & , n = 1 \\ F_{n-2} + F_{n-1} & , n \geq 2 \end{cases}$$

Hypothèses: $n \in \mathbb{N} \mid n \geq 0$

Fonction: int fibo(int n)

In	n un entier où n>=0
Out	le $n^{ème}$nombre de Fibonacci

Implémentation

Après avoir écrit les pseudo-codes, implémentez les dans les 2 langages:

Java: Réalisez la fonction en Java dans votre classe « MyMath » et testez la avec un test unitaire dans « TestMyMath ».

Python Réalisez la fonction en Python dans le module « mymath » et le test unitaire dans le module « test_mymath ».

2.2 Calculez le nombre d'or à partir de la suite de fibonacci

Le nombre d'or appelé aussi le nombre Phi où $\varphi = \frac{1+\sqrt{5}}{2}$. Il est également possible d'obtenir le nombre d'or à partir de la suite de Finonacci avec

$$\varphi = \lim_{1 \to \infty} \frac{F_{n+1}}{F_n}$$

Itérations	n	F_{n+1}	F_n	$\frac{F_{n+1}}{F_n}$
1	2	1	1	1
2	3	2	1	2
3	4	3	2	1.5
4	5	5	3	1.6666...
5	6	8	5	1.6
...	...	...	...	...

2.2.1 Réalisez une fonction qui donnera le nombre d'or après la $n^{ème}$itération

Fonction: double nombreOr(int n)

In n un entier où n>=1

Out un réel, qui représentera le nombre d'or obtenu après le $n^{ème}$itération

Implémentation

Implémentez votre pseudo-code dans les 2 langages ainsi que le test unitaire associé

2.2.2 Réalisez une fonction qui donnera le nombre d'itération nécessaires pour obtenir le nombre d'or avec une précision donnée

Fonction: int nombreOr(double epsilon)

In n une réel qui indique une précision

Out un entier qui indiquera le nombre d'itérations nécessaires pour obtenir le nombre d'or avec une précision epsilon

Algorithme:

Votre algorithme devra d'abord calculer le nombre d'or à partir de la formule : $\varphi = \frac{1+\sqrt{5}}{2}$, ensuite il devra appliquer l'approximation $\frac{F_{n+1}}{F_n}$et s'arrêter dès que le résultat sera proche de φ d'une précision epsilon:

$$minimum\ de\ n \mid \left| \frac{1+\sqrt{5}}{2} - \frac{F_{n+1}}{F_n} \right| <= epsilon$$

Exemple: si epsilon= 0.001 votre fonction devra renvoyé 9 car c'est la première itération où

$$|\varphi - \frac{F_{10}}{F_9}| < 0.001$$

Implémentation

Implémentez votre pseudo-code dans les 2 langages ainsi que le test unitaire associé

3 Nombres premiers

Un nombre premier est un nombre entier supérieur à 1 qui n'est divisible que par lui-même et par un.

3.1 Écrivez une fonction qui vérifie si un nombre est premier

Fonction: booleéan estPremier(int n)

In	n un entier où n>=2
Out	un booléen qui sera à vrai si le nombre est un nombre premier

Algorithme:

Pour savoir si un nombre est premier, vous devez tester qu'il n'est pas divisible par plusieurs diviseurs. Le nombre de diviseurs à tester, peut être fortement réduit en sachant que si A divise N alors il existe un « B » tel que A*B=N. Donc, en supposant que A<=B, le plus grand diviseur à tester sera A tel que $A^2 = N$

Implémentation

Implémentez votre pseudo-code dans les 2 langages ainsi que le test unitaire associé