Langage et Programmation – 2ème séance

Fabien Tarissan

1 Premiers pas

Exercice 1 — *Opérations simples* Écrire les expressions permettant de :
- calculer le résultat d'un expression arithmétique (par exemple $\frac{3+9}{2*3}$)
- calculer le quotient de la division euclidienne de 42 par 5
- calculer le reste de la division euclidienne de 42 par 5
- calculer le résultat de la division réelle de 42 par 5
- tester l'égalité entre 6 et $2*3$.
- concatener la chaîne "j'enseigne " avec la chaîne "la programmation dite "impérative""

Exercice 2 — *Interagir avec l'utilisateur*
- Écrire un programme qui affiche à l'écran "Bonjour !"
- Écrire un programme qui demande à l'utilisateur de rentrer son nom, le mémorise dans une variable et affiche à l'écran "Bonjour" suivi du nom contenu dans la variable
- Écrire un programme qui demande à l'utilisateur de rentrer son année de naissance et affiche à l'écran son age.

2 Fonctions simples

Exercice 3 — *Fonction mystère* Soit le code suivant :

```
double mystere (double a, double b, double c) {
  if (a<b) {
     if (a<c) return a;
     else return c;
  } else {
     if (c<b) return c;
     else return b;
  }
}
```

1. Quel est le résultat de l'application de `mystere(1.0,3.5,5.9)` ?
2. Que fait la fonction ? Renommer la fonction et lui donner une entête correcte.
3. Combien de tests différents faut-il faire pour s'assurer que la fonction est correcte ?

Exercice 4 — *Fonction de conversion*
- Écrire une fonction `farhenheitEnCelsius` qui convertit une température donnée en farhenheit en celsius ($c = \frac{5*(f-32)}{9}$. Refléchir aux types des arguments et de la valeur de retour.
- Proposer une fonction qui réalise la conversion inverse de la précédente

Exercice 5 — *Entier naturel ?* Écrire une fonction `naturel` qui teste si un entier est un entier naturel ou non.

Exercice 6 — *Divisible ?* Écrire une fonction `divisible` qui, étant donnés deux nombres entiers, teste si le premier est divisible par le second.

Exercice 7 — *Nombre premier* À l'aide de la fonction précédente, écrire une fonction `nbPremier` permettant de déterminer si un nombre donné est *premier* ou non.

Exercice 8 — *Trouver l'erreur* Dans ce qui suit, il est demandé de prédire ce que vont faire les programmes avant de les tester.

- Soit le code suivant :
```
boolean malade(double x) {
  if (x=37.5) return true;
  else return false;
}
```
Que va-t-il se passer lors de l'appel `malade(39.0)` ? Corriger le code.

- Soit le code suivant :
```
double x= 4/3;
println(x);
```
Qu'affiche ce bout de programme ? Pourquoi ?

- Soit le code suivant :
```
if (3<5) x=6/2;
else x=6/0;
```
Que fait le programme lors de l'exécution de ce code ?

- Soit maintenant la fonction :
```
int si(boolean b, int e1, int e2) {
  if (b) return e1;
  else return e2;
}
```
Que se passe-t-il lors de l'appel de fonction `si(3<5, 6/2, 6/0);` ?

Exercice 9 — *Échange de valeurs* Soit la fonction suivante :

```
void echange(int x, int y) {
  int tmp=x;
  y=x;
  x=tmp;
}
```

Que vaut la variable `a` après l'execution suivante :

```
int a = 1;
int b = 5;
echange(a,b);
}
```

Commenter le résutat

3 Les tableaux

Exercice 10 — *Premières opérations*

- Créer un tableau `t` contenant les valeurs | 1 | 3 | 5 |.
- Afficher la taille de `t`
- Écrire une fonction `afficheTableauInt` permettant l'affichage d'un tableau d'entiers passé en argument.
- Écrire un programme demandant à l'utilisateur un entier n et créant un tableau `tab0n` contenant les valeurs de 0 à n (inclus).
- Écrire un programme demandant à l'utilisateur un entier n, créant un tableau de taille n et demandant ensuite à l'utilisateur les valeurs à mémoriser dans le tableau. [1]
- Écrire une fonction qui, étant donnés un tableau d'entiers, et deux indices `i` et `j`, échange les valeurs de ces deux indices si c'est possible, ne fait rien sinon (*i.e.* si les indices ne correspondent pas à des cases du tableau).

1. il sera utile d'encapsuler primitive dans une fonction `int [] readTableauInt(int n)` afin de la réutiliser par la suite.

Exercice 11 — *Moyenne* Écrire une fonction `moyenne` qui calcule la moyenne des éléments d'un tableau d'entiers.

Exercice 12 — *Min et max* Soit un tableau d'entiers naturels non vide. Écrire une fonction `maxTableau` qui calcule le maximum contenu dans le tableau. Donner la variante calculant le minimum.

Exercice 13 — *Appartenance* Écrire une fonction `appartient` qui teste l'appartenance d'un entier dans un tableau.

Exercice 14 — *Occurence* Écrire une fonction `nbOccurence` qui compte le nombre d'occurence d'un entier dans un tableau.

Exercice 15 — *Occurence du maximum* Écrire une fonction `nbOccurenceMax` qui compte le nombre d'occurence du maximum dans un tableau. Quelle est la complexité de votre fonction. Peut-on faire mieux ?

Exercice 16 — *Égalité* Qu'affiche le code suivant :

```
int [] t1={1,2,3};
int [] t2={1,2,3};
println(t1==t2);
```

Expliquer pourquoi. Proposer une fonction qui teste l'égalité de deux tableaux d'entiers.

Exercice 17 — *Même contenu* Écrire une fonction qui teste si deux tableaux d'entiers de même taille contiennent les mêmes valeurs. On supposera dans un premier temps que tous les éléments d'un tableau sont distincts. Que pensez-vous de la complexité de votre fonction. Peut-on faire mieux ?

Refléchir à la variante dans laquelle les éléments peuvent être répétés.

Exercice 18 — *Tri* Implémenter les algorithmes de tris vus en cours d'algorithmique.

4 Les fonctions récursives

Exercice 19 — *Somme des n premiers entiers* Écrire une fonction récursive `sommeN` qui, étant donné un entier n, renvoie la valeur de $\sum_{i=1}^{n} i$.

Exercice 20 — *Calcul du* PGCD On rappelle que l'algorithme d'euclide permet de déterminer le PGCD de deux entiers à l'aide des divisions euclidiennes en remarquant que :
- $\text{pgcd}(n_1, n_2) = n_1$ si $n_2 = 0$.
- $\text{pgcd}(n_1, n_2) = pgcd(n_2, r)$ sinon où r est le reste de la division euclidienne de n_1 par n_2 (on suppose $n_2 \leqslant n_1$).

À partir de ce principe, écrire une fonction récursive `pgcd` calculant le PGCD de deux entiers.

En déduire une fonction `premiersEntreEux` testant si deux entiers sont premiers entre eux.

Exercice 21 — *Puissance* Écrire une fonction récursive `puissance` qui, étant donnés un nombre x et un entier n, renvoie la valeur de x^n.

Écrire également sa variante itérative.

Combien d'appels de fonction (ou de passages dans la boucle) sont faits lors de l'appel à `puissance(x,n)` ?

Exercice 22 — *Puissance efficace* En s'appuyant sur la définition ci-dessous, proposer une fonction récursive `puissanceEfficace` plus efficace que la précédente pour calculer x^n :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ (x^{n/2})^2 & \text{si } n \text{ est pair} \\ x * (x^{n/2})^2 & \text{si } n \text{ est impair} \end{cases}$$

Combien d'appels de fonction sont faits lors de l'appel à `puissanceEfficace(x,n)` ?

Exercice 23 — *Approximation de la racine carrée* Le but de cet exercice est d'écrire une fonction qui calcule une valeur approchée de la racine carrée d'un nombre. Soit x un nombre réel positif, une valeur approchée de $\sqrt{x}$ est donnée par le calcul des valeurs de la suite suivante :

$$u_n = \begin{cases} 1 & \text{si } n = 0 \\ \frac{u_{n-1} + \frac{x}{u_{n-1}}}{2} & \text{sinon} \end{cases}$$

Écrire une fonction qui, étant donnés un nombre x et un entier n, renvoie l'approximation au rang n de $\sqrt{x}$. Attention à l'efficacité de votre fonction [2]

Exercice 24 — *Fibonacci* Écrire une fonction récursive `fibonacci` qui, étant donné un entier n, renvoie la valeur de la suite de Fibonacci au rang n :

$$\text{fib}_n = \begin{cases} 1 & \text{si } n = 0 \text{ ou } n = 1 \\ \text{fib}_{n-1} + \text{fib}_{n-2} & \text{sinon} \end{cases}$$

Donner la variante itérative. Que pensez-vous de la complexité des deux variantes ?

2. plein de variantes avec d'autre valeurs : approximation de π, de e, ...