Unité et diversité des langages

Devoir à la maison à rendre au format PDF sur pronote pour le vendredi 11 mars.
Tout autre format est interdit.
Travail seul ou à deux.
La collaboration entre groupes est interdite et sera sanctionnée.

Nous allons voir comment on écrit l'algorithme de Fisher-Yates dans différents langages. Celui-ci donne une méthode simple pour mélanger (*to shuffle* en Anglais) une liste. Le voici écrit en langage naturel : La fonction

- prend en entrée une liste d'éléments numérotés de 0 à n-1;
- parcourt cette liste et à chaque étape échange l'élément courant avec un élément choisi au hasard parmi ceux qui le précèdent;
- ne renvoie aucune valeur.

Langage naturel

```
fonction shuffle(L : liste)
    variables
       n, i, j : entiers
    début
        n = longueur(L)
        pour i allant de 1 à n - 1:
            choisir un j nombre au hasard entre 0 et i
            si j < i
                echanger L[i] et L[j]
        fin
    fin
```

Tu peux visualiser un exemple d'application de la fonction en vidéo ici :

https://youtu.be/-WLu_OqyT5s.

I Comprendre l'algorithme

On se donne la liste `lst = [1, 2, 3, 4, 5]` et on veut la mélanger en utilisant la fonction `shuffle` donnée ci-dessus (c'est à dire que le paramètre `L` prend la valeur de ;`lst`, c'est-à-dire `[1, 2, 3, 4, 5]`).
On va examiner en détail ce qui se passe lors de l'appel `shuffle(lst)`.

1. Que vaut la variable `n` de la ligne 5 ? (1pt)
2. Quelles sont les valeurs que va prendre la variable `i` de la ligne 6 ? (1pt)
3. En choisissant toi-même le nombre `j` au hasard comme indiqué, explique étape par étape le déroulement de la boucle `pour` et donne à chaque fois la valeur de `L`. (2pts)
4. Suivant ta méthode, quelle valeur de `lst` obtiens-tu après avoir appelé la fonction `shuffle(lst)` ? (1pt)

II Traduire en Python

Complète le script suivant pour écrire la fonction précédente en PYTHON. Tu pourras utiliser la fonction `randint` : `randint(a, b)` renvoie un entier au hasard compris entre `a` et `b` inclus. (2pts)

Code Python

```
from random import randint

def shuffle(L : list) -> None
    # compléter
```

III Observer et classer

En annexe, tu trouveras des versions de la fonction `shuffle` dans différents langages. Tu vas répondre aux questions suivantes en citant des exemples et en argumentant. Voici deux choses à ne pas faire :

- Répondre « Pour certains langages oui, pour d'autres non » (aucun intérêt) ;
- Répondre en faisant une liste exhaustive et détaillée (trop long).

1. Suivant les langages :

a. Quelles sont les notations pour signifier une *affectation* ? (1pt)

b. Les éléments de la liste sont-ils toujours numérotés de 0 à n-1 ? (1pt)

c. Comment indique-t-on un bloc d'instructions (à l'intérieur d'une boucle ou dans le cas d'un test) ? (1pt)

d. Faut-il déclarer les variables avant de commencer à programmer la fonction ? (1pt)

e. Faut-il indiquer les types des variables ? (1pt)

2. Peux-tu regrouper quelques langages qui se ressemblent ? (1pt)

3. Y a-t-il un ou des langages très proches de PYTHON ? (0,5pt)

4. Y en a-t-il un ou plusieurs qui sont très différents ? (0,5pt)

IV Établir une chronologie

Deux critères peuvent t'aider à classer les langages selon leur « âge ». Ce ne sont pas des règles générales, simplement des tendances.

- Les langages de programmation les plus anciens sont souvent assez verbeux : beaucoup de lignes sont nécessaires pour décrire la structure, préciser les variables, les types de données. Avec les progrès des analyseurs de code, un grand nombre de ces informations peut être inféré (déduit par le compilateur ou l'interpréteur), ce qui permet de ne pas les écrire explicitement.

- Avec le temps, la gestion des variables locales se simplifie considérablement. Au début, elles devaient toutes être déclarées au début de la fonction (parfois dans une section spécifique) avec leur type. Ensuite, deux évolutions sont apparues : elles ont pu être déclarées « en cours de route » (voire pas du tout) et la spécification de leur type est parfois devenue non systématiquement nécessaire.

1. À l'aide des critères précédents, essaye d'établir une chronologie des langages. Le but n'est pas d'avoir une classification exacte : tu peux dans un premier temps essayer de trouver les langages les plus anciens ainsi que les plus récents (sans aller chercher sur Internet), puis ensuite aller vérifier sur Internet et décrire ce que tu avais bien vu ainsi que les erreurs que tu avais commises. (2pts)

2. Choisis un (et un seul) des langages de la liste suivante et rédige un petit document (entre une demi page et une page) qui explique l'histoire du langage, ses spécificités et des exemples de projets réalisés dans ce langage (s'il y en a).
   Tu veilleras à éviter les « copier-coller » d'Internet. Il ne faudra pas non plus utiliser des mots que tu ne comprends pas : s'il le faut tu dois pouvoir donner une définition simple des termes employés. (3pts)

Liste des langages : C++, C#, JAVA, JAVASCRIPT, PHP, PYTHON, GO, RUBY.

Annexe
Implémentation de l'algorithme dans divers langages

code Ada

```
procedure Shuffle (Tab : in out Array_Type) is
    package Discrete_Random is new
        Ada.Numerics.Discrete_Random(Result_Subtype =>Integer);
    use Discrete_Random;
    J : Integer;
    G : Generator;
    TMP : Element_Type;
begin
    Reset (G);
    for I in Tab'Range loop
        J := (Random(G) mod I) + 1;
        if J < I then
            TMP := Tab(I);
            (I) := Tab(J);
            Tab(J) := TMP;
        end if;
    end loop;
end Shuffle;
```

code Algol

```
procedure shuffle (tab, n);
    array tab; integer n;
    begin
        integer i, j, tmp;
        for i := 1 step 1 until n do
        begin
            j := random(i + 1);
            if j < i then
            begin
                tmp := tab[i];
                tab[i] := tab[j];
```

```
                tab[j] := tmp;
            end
        end
    end shuffle
```

code Basic

```
100 FOR I = 1 TO LONGUEUR
110 J = INT(RND(1) * I + 1)
120 IF I = J THEN GOTO 160
130 TMP = TAB(I)
140 TAB(I) = TAB(J)
150 TAB(J) = TMP
160 NEXT I
170 END
```

code C

```
void shuffle(int tab[], int n) {
    int i;
    for (i = 1; i < n; ++i) {
        int j = random(i + 1);
        if (j < i) {
            int tmp = tab[i];
            tab[i] = tab[j];
            tab[j] = tmp;
        }
    }
}
```

caml

```
let shuffle tab =
    for i = 1 to Array.length tab - 1 do
        let j = Random.int (i + 1) in
```

```
        if j < i then (
            let temp = tab.(i) in
            tab.(i) <- tab.(j);
            tab.(j) <- temp
        )
    done
;;
```

code Cobol

```
IDENTIFICATION DIVISION.
PROGRAM-ID. shuffle.

DATA DIVISION.
LOCAL-STORAGE SECTION.
01 i            PIC 9(8).
01 j            PIC 9(8).
01 temp         PIC 9(8).

LINKAGE SECTION.
78 Table-Len    VALUE 10.
01 ttable-area.
03 ttable       PIC 9(8) OCCURS Table-Len TIMES.

PROCEDURE DIVISION USING ttable-area.
    PERFORM VARYING i FROM 2 BY 1 UNTIL i = Table-Len
        COMPUTE j =
            FUNCTION MOD(FUNCTION RANDOM * 10000, Table-Len) + 1
        If j < i
            MOVE ttable (i) TO temp
            MOVE ttable (j) TO ttable (i)
            MOVE temp TO ttable (j)
        END-IF
    END-PERFORM
    GOBACK
    .
```

code Fortran

```
subroutine shuffle (tab, n)
    integer n, tab(*)
    integer i, j, temp
    real r

    do 10 i = 2, n
        call random_number(r)
        j = int(r * i) + 1
        if (j < i) then
            temp = tab(j)
            tab(j) = tab(i)
            tab(i) = temp
        endif
10  continue
    return
lend
```

code Go

```
func shuffle(tab []int) {
    for i := 1; i < len(tab); i++ {
        j := rand.Intn(i + 1)
        if j < i {
            tmp := tab[i]
            tab[i] := tab[j]
            tab[j] := tmp
        }
    }
}
```

code Java

```
public static void shuffle (int[] tab) {
    for (int i = 1; i < tab.length; i++) {
        int j = gen.nextInt(i + 1);
        if (j < i) {
```

```
            int temp = tab[i];
            tab[i] = tab[j];
            tab[j] = temp;
        }
    }
}
```

code javascript

```
function shuffle(tab) {
    for (var i = 1; i < tab.length; i++) {
        var j = Math.floor((i + 1) * Math.random());
        if (j < i) {
            var temp = tab[rand];
            tab[rand] = tab[i];
            tab[i] = temp;
        }
    }
}
```

code kotlin

```
fun shuffle(tab: Array<Int>) {
    for (i in 1 until tab.size) {
        val j = (0..i).random()
        if (j < i) {
            val tmp = tab[i]
            tab[i] = tab[j]
            tab[j] = tmp
        }
    }
}
```

code Ruby

```ruby
def shuffle(tab)
    tab.each_index do |i|
        j = rand(i + 1)
        if j < i
            tab[i], tab[j] = tab[j], tab[i]
        end
    end
end
```

code Pascal

```pascal
procedure shuffleList(var tab: tableau);
var
    i, j, tmp : integer;
begin
    for i := 1 to high(tab) - low(tab) do begin
        j := random(i + 1);
        if j < i then
            tmp := tab[i + low(tab)];
            tab[i + low(tab)] := tab[j + low(tab)];
            tab[j + low(tab)] := tmp
        end
    end
end;
```

code Scala

```scala
def shuffle(tab: Array[Int]) = {
    for (i <- 1 to tab.size - 1) {
        val j = util.Random nextInt(i + 1)
        if (j < i) {
            val tmp = tab(i)
            tab(i) = tab(j)
            tab(j) = tmpwd
        }
    }
```