Développement initiatique

Sujet 2 : instructions conditionnelles

Notions à acquérir

— conditions (expressions booléennes) ;
— instruction « `si alors sinon` » ;
— instruction « `si alors` » ;
— instruction « `si alors sinonSi sinon` ».

1 Expressions booléennes

Les expressions suivantes sont-elles bien de type booléen ? (C'est à dire, peut-on utiliser ces expressions dans la condition d'un `si` ?)

— $(x < y)$ *et* $(z == 4)$ (avec x, y, et z entiers)
— $(x < y)$ *ou* $(z$ *et* $(t == 4))$ (avec x, y, z, et t entiers)

Dans les expressions qui suivent, indiquez les types des variables utilisées afin que ces expressions soient booléennes, si cela est possible.

— x *ou* $(y$ *et* $(z < 5))$
— x *ou* $(y$ *et* $(x < 5))$

2 Syntaxe du « `si alors sinon` »

Les instructions 1 et 3 sont incorrectes, les instructions 2 et 4 les corrigent. Réindentez le programme pour y voir plus clair, et expliquer dans quels cas on exécute `<instructions1>` ou `<instructions2>` :

1. **si <condition1> alors si <condition2> alors <instructions1> sinon <instructions2> finSi**
2. **si <condition1> alors si <condition2> alors <instructions1> sinon <instructions2> finSi finSi**
3. **si <condition1> alors si <condition2> alors <instructions1> finSi finSi sinon <instructions2>**
4. **si <condition1> alors si <condition2> alors <instructions1> finSi sinon <instructions2> finSi**

3 Algorithmes équivalents ?

Les deux algorithmes `piegeux` et `nonPiegeux` sont-ils équivalents ?

```
Algo nonPiegeux
Variables
   n : entier
Debut
   afficher(...) ; saisir(n)
   si n mod 2 != 0 alors // n impair
      n <- 3*n + 1
   sinon
      n <- n div 2
   finSi
   afficher("Resultat = ", n)
Fin piegeux
```

```
Algo piegeux
Variables
   n : entier
Debut
   afficher(...) ; saisir(n)
   si n mod 2 != 0 alors
      n <- 3*n + 1
   finSi
   si n mod 2 == 0 alors
      n <- n div 2
   finSi
   afficher("Resultat = ", n)
Fin nonPiegeux
```

4 Paire, brelan, carré...

Etant donnés quatre nombres entiers, écrire un algorithme qui affiche :

"C'est un carré !"	si les quatre nombres sont égaux ;
"C'est un brelan !"	si trois des nombres sont égaux et le quatrième distinct ;
"C'est une paire !"	si deux nombres sont égaux, les deux autres étant distincts des deux premiers et entre eux ;
"C'est une double paire !"	si deux nombres sont égaux, les deux autres étant distincts des deux premiers et égaux entre eux ;
"Pas de chance !"	si tous les nombres sont distincts.

5 Nombre de valeurs paires

Etant donnée une suite de nombres entiers positifs ou nuls, saisie à partir du clavier, et qui se termine par le nombre 0, écrire un algorithme qui calcule le nombre de valeurs paires contenues dans cette suite.

6 Valeur maximum

Ecrire un algorithme qui demande 20 nombres entiers relatifs (valeurs négatives possibles) à l'utilisateur, et qui affiche ensuite à l'écran la valeur du plus grand des nombres saisis.

7 Racines de polynômes

1. Ecrire un algorithme qui recherche les solutions réelles d'une équation polynomiale de la forme :
   $a.x^2 + b.x + c = 0$
   On suppose que a, b, c sont des coefficients réels donnés par l'utilisateur.
2. Ecrire un algorithme qui recherche les solutions *entières* entre -100 et 100 d'une équation polynomiale de la forme :
   $a.x^3 + b.x^2 + c.x + d = 0$
   On suppose que a, b, c, d sont des entiers donnés par l'utilisateur.

8 Un peu de lecture

Considérons une phrase écrite avec seulement des lettres minuscules et qui est sans ponctuation, mis à part un point final. Cette phrase est constituée de mots séparés les uns des autres par une espace [1] et une seule. La phrase contient au moins un mot. Elle commence par la première lettre du premier mot. Elle se termine par le point final qui est accolé à la dernière lettre du dernier mot.

Pour gérer la lecture des différentes lettres, les algorithmes ci-dessous devront effectuer une boucle qui à chaque itération saisit un seul caractère à la fois au clavier (ce qui rend les algorithmes plus subtils), jusqu'à la lecture du point final.

1. Ecrire un algorithme qui calcule le nombre de lettres 'm' présentes dans la phrase.
2. Ecrire un algorithme qui calcule le nombre de mots constituant cette phrase.
3. Ecrire un algorithme qui calcule le nombre de syllabes "le" présentes dans la phrase.
4. Ecrire un algorithme qui calcule la longueur du dernier mot de la phrase.
5. Ecrire un algorithme qui calcule quelle est la longueur du plus long mot de la phrase.

9 Somme la plus proche d'un nombre

Ecrire un algorithme qui saisit au clavier un nombre `nb` strictement positif, puis additionne les premiers nombres entiers (naturels) jusqu'à obtenir une somme la plus proche possible de `nb`.

En cas d'écarts relatifs égaux, prendre la somme trouvée par valeur inférieure.

Envisager les différents cas de figure.

10 Année bissextile

Une année est bissextile si elle est divisible par 4 mais pas par 100, à moins qu'elle ne soit divisible par 400. Ecrire un algorithme qui saisit une année et affiche *année bissextile* si elle est bissextile, et *année non bissextile* sinon.

1. Pour une implantation dans un langage de programmation, au moins en Java, choisir un autre caractère que l'espace, un ' :' par exemple.

11 Validité d'une date

On dit qu'une date écrite sous la forme de trois entiers (j, m, a), j représentant le jour, m le mois et a l'année, est valide si et seulement si $a > 0$, $1 \leq m \leq 12$ et $1 \leq j \leq nombreDeJoursDuMois$.

Ecrire un algorithme qui saisit une date et affiche `vrai` si elle est valide, et `faux` sinon.

12 Nombre de jours depuis le début d'année

(Exercice à réaliser en TP plutôt)

A partir d'une date $d = (j, m, a)$, écrire un algorithme qui calcule le nombre de jours écoulés entre le 1^{er} janvier de l'année a et la date d (ces 2 jours compris).

Par exemple, si $d = (5, 3, 2016)$ le nombre de jours écoulés est $31+29+5 = 65$ et si $d = (5, 3, 2018)$ le nombre de jours écoulés est $31 + 28 + 5 = 64$.