Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr

TD n°4

Objectifs: Renforcer l'usage de boucles à compteur. Écrire des premiers programmes utilisant une boucle à événements. Choisir les types de répétition les plus appropriés en fonction de la tâche à réaliser.

Exercice 1 : Nombre d'occurrences d'un caractère [REP-FOR-STR, REP-ACC-NUM]

Écrire un programme qui calcule le nombre d'occurrences d'un caractère dans une chaîne de caractères.
Le programme demande d'abord la chaîne à analyser, puis le caractère dont les occurrences doivent être comptées.
Voici quelques exemples d'exécutions :

```
totoro
o
3
```

```
totoro
a
0
```

```
a
0
```

Le dernier exemple correspond à la saisie d'une chaîne vide.

Exercice 2 : Première occurrence d'un caractère [REP-WHILE-STR*]

Écrire un programme qui indique à quel indice se situe la première occurrence d'un caractère dans un texte saisi par l'utilisateur. Le programme demande d'abord la chaîne à analyser, puis le caractère à rechercher.
Voici quelques exemples d'exécutions :

```
 Hello world !
o
Première occurrence à l'indice 4
```

```
 Phonographe
f
Ce caractère est absent.
```

Exercice 3 : Saisie d'une note valide [REP-WHILE-SAISIE*]

1. Écrire un programme permettant de saisir une note valide, c'est-à-dire comprise entre 0 et 20. Pour cela, utilisez une boucle `do while` qui effectue la saisie d'un nombre jusqu'à ce qu'il appartienne à l'intervalle $[0..20]$, le programme s'arrête alors.
2. Faire l'algorigramme du programme et la trace d'exécution si les nombres saisis en entrée sont -5, 25, 15.
3. Améliorer votre algorithme pour qu'il affiche `"Notes négatives non admises"` si le nombre est inférieur à zéro, et `"Notes supérieures à 20 non admises"` si le nombre est supérieur à 20, et enfin `"Note valide"` lorsque la note est bien dans l'intervalle.

Exercice 4 : La suite de Syracuse [REP-WHILE]

Prenez un entier positif ; s'il est pair, divisez-le par 2 ; s'il est impair, multipliez-le par 3 et ajoutez lui 1… Réitérez ce processus sur plusieurs exemples : que semble-t-il se passer ?

Partons de l'entier 7, et regardons la suite alors construite : 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1…. Cette suite devient cyclique, puisque l'obtention de la valeur 1 fait "boucler" indéfiniment l'algorithme.

On conjecture que l'on finit toujours par trouver la valeur 1 au fil des calculs quel que soit l'entier de départ... C'est la conjecture de Syracuse (encore appelée *problème* $3n + 1$) ... qui attend toujours une preuve !

En utilisant une boucle `while`, écrire le programme calcule et affiche la suite de Syracuse pour un nombre donné par l'utilisateur/utilisatrice.

Par exemple, si le nombre saisi est **7** alors le programme devra afficher :

```
7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1,
```

Pour vous aider à décomposer le problème, répondre à ces questions:

1. À chaque tour de boucle, on doit calculer le prochain élément de la suite. En supposant que la variable `x` contient l'élément courant de la suite, écrire le code qui permet de calculer la nouvelle valeur de `x` en fonction de sa valeur actuelle.
2. Quelle est la condition de continuation de la boucle ?
3. Écrire le programme affichant la suite de `syracuse`.
4. Dessiner l'organigramme du programme et faire la trace d'exécution avec en entrée la valeur 5, puis la valeur 1.

Prolongement

Exercice 5 : Somme de nombres saisis au clavier

— Écrire un programme qui permet de saisir 10 nombres, puis d'afficher leur somme.
— Que faut-il modifier dans ce programme pour qu'il demande d'abord à l'utilisateur le nombre de nombres à saisir.