Développement web S1 (R112) – TP #1

Dans cette séance de TP vous allez apprendre à décrire des algorithmes en pseudo-code, sans utiliser un vrai langage de programmation.

Liens utiles : CM #1

Il n'y a pas d'exercices à rendre pour cette séance de TP. Faites vos exercices sur feuille, puisqu'un éditeur de texte n'est pas adapté à ces exercices.

Exercices

Pour chaque exercice, nous allons concevoir un algorithme et le décrire sous deux formes : avec un organigramme de programmation et en pseudo-code.

Ce sont des langages informels, qui ne sont pas exécutés par une machine. L'objectif est de bien décrire l'algorithme sans ambiguïté. Ces techniques permettent de bien réfléchir à un algorithme avant d'écrire le code en langage informatique.

Exercice 1

Décrivez un algorithme qui calcule le nombre le plus petit parmi deux valeurs.

Vous avez deux variables, `x1` et `x2`, qui contiennent des nombres. Il faut donner la plus petite valeur des deux à une variable appelée `min`. Vous aurez besoin d'une structure conditionnelle.

Exercice 2

Décrivez un algorithme qui calcule le nombre le plus petit parmi trois valeurs.

Cette fois vous avez trois variables en entrée : `x1`, `x2` et `x3`.

Exercice 3

Décrivez un algorithme qui calcule le nombre le plus petit dans une liste avec 10 nombres.

La variable qui contient la liste s'appelle `liste`. Pour accéder au nombre dans la position i de la liste, vous écrivez `liste[i]`. Cette fois vous aurez besoin aussi d'une boucle.

Exercice 4

Décrivez un algorithme qui trie une liste de 10 nombres.

La variable qui contient la liste s'appelle `liste`. La liste avec les nombres triées s'appelle `liste_triée`.

Exercice 5

Décrivez un algorithme qui joue à deviner un nombre.

Il y a un nombre secret entre 0 et 100. L'utilisateur doit donner un nombre, et on lui dit si le nombre secret est plus grand, plus petit, ou si c'est le bon. Quand il trouve le bon nombre, on lui affiche le nombre d'essais.

La variable qui contient le nombre secret s'appelle `nombre_secret`. On peut afficher une valeur ou un texte avec l'instruction `écrire`. On peut lire un nombre avec l'instruction `lire`. Par exemple :

```
écrire "Entre un nombre entier entre 0 et 100"
lire n
```