Workshop I

Algorithmique

Un algorithme est une suite d'instructions à effectuer chronologiquement pour élaborer quelque chose.

Il prend des données en entrée, applique un traitement particulier et fournit des données en sortie.

On utilise inconsciemment des algorithmes tous les jours :

- Une recette de cuisine est composée d'ingrédients, d'une série d'étapes à appliquer et d'un plat à élaborer
- Un trajet est composé de personnes et/ou de bagages, d'une série de chemins à emprunter et d'un endroit à atteindre

Algorithme informatique

Un algorithme informatique est un programme dans lequel on explique à la machine comment effectuer une tâche à notre place.

Généralement, un algorithme répond à un problème ou un énoncé.

L'écriture algorithmique est un exercice qui ne dépend d'aucun langage informatique. Son but est d'illustrer un schéma compréhensible.

Dans ce cours nous utiliserons d'une part une écriture schématique et d'autre part une écriture Javascript pour pouvoir tester les algorithmes.

Algorithme informatique

Il y a 3 principes qu'il est important de définir :

- La complexité :
  - En combien de temps un algorithme va-t-il atteindre le résultat attendu
  - De quelle mémoire a-t-il besoin ?
- La calculabilité :
  - Existe-t-il un algorithme pour une tâche défini ?
  - Existe-t-il des tâches pour lesquelles il n'existe aucun algorithme ?
- La correction :
  - Peut-on être sûr qu'un algorithme effectue correctement la tâche pour laquelle il a été élaboré ?

Les données

Les données en entrée sont des éléments nécessaires à l'élaboration d'une tâche, il est important d'en définir suffisamment pour que les données en sortie soit le plus correcte possible.

En informatique, une donnée est appelée variable et on utilise différentes instructions (selon le langage informatique employé) pour déclarer une variable.

La déclaration de variable permet de réserver un espace de stockage identifié par un nom et parfois par un type de donnée.

Les donnée : Les type de données

Les types de données prédéfinis :

- Les booléens : true (1) && false (0)
  - Les booléens sont des variables à deux états destinés à représenter des valeurs de vérité.
- Les entiers relatifs : -100 ; 0 ; 42 ; etc...
  - Les entiers relatifs sont des nombres positif ou négatif sans chiffres derrière la virgule.
- Les entiers réels : -100 ; -10,42 ; 0 ; 42,42 ; etc...
  - Les entiers réels sont des nombres positif ou négatif avec ou sans chiffres derrière la virgule.
- Les chaînes de caractères : 'H3ll0 w0rld !'
  - Les chaînes de caractères peuvent contenir des lettres, des chiffres et des symboles.

- Les autres types :
  - Il existe de nombreux autres types (selon le langage informatique employé) qui serviront pour des cas spécifiques. (tableaux, objets, liste, énumération, etc...)

Les données : Déclaration de variable

Pour déclarer une variable avec Javascript, on utilisera var avec l'opérateur d'affectation égal (=) :

```
// Déclaration de maVariable avec une valeur numérique
var maVariable = 42;

// Modification de la valeur numérique de maVariable
maVariable = 100;

// Déclaration d'une seconde variable numérique
var maVariable2 = 21;

// Déclaration d'une variable en utilisant les valeurs d'autres variables
var maVariable3 = maVariable + maVariable2;
// maVariable3 vaut 121

// Déclaration d'une variable alphanumérique
var maVariable4 = 'additionner maVariable et maVariable2 est égal à ' + maVariable3;
// maVariable4 contient 'additionner maVariable et maVariable2 est égal à 121'

// Déclaration d'une variable tableau
var maVariable5 = ['Hello', 'world !'];
// maVariable5 contient un tableau avec deux éléments (item) alphanumérique

// Déclaration d'une variable objet
var maVariable6 = {ville: 'Paris', departement: 75};
// maVariable6 contient un objet avec deux éléments (item), chaque item a une clé (key)
```

Les données : Opérateurs d'affectation

Il existe différents opérateurs d'affectation :

```
// Affection
var maVariable = 42;

// Affectation après adidtion, soustraction, multiplication et division
maVariable += 100;
// maVariable vaut 142
maVariable -= 42;
// maVariable vaut 100
maVariable *= 2;
// maVariable vaut 200
maVariable /= 10;
// maVariable vaut 20

// Affectation du reste
maVariable %= 18;
// maVariable vaut 2

// Affection après exponentiation
maVariable **= 2;
// maVariable vaut 4
```

Les données : Opérateurs arithmétiques

Il existe différents opérateurs arithmétiques :

```
// Affection
var maVariable = 42;

// Incrémentation : Ajoute 1
maVariable++;
// maVariable vaut 43

// Décrémentation : Enlève 1
maVariable--;
// maVariable vaut 42

// Négation unaire : Renvoie l'opposé
maVariable = -maVariable;
// maVariable vaut -42
maVariable = -maVariable;
// maVariable vaut 42

// Reste : Renvoie le reste de la division
maVariable = maVariable % 5;
// maVariable vaut 2

// Opérateur d'exponentiation : Calcul un nombre élevé à une puissance
maVariable = maVariable ** 2;
// maVariable vaut 4
```

Les instructions

Les instructions sont la partie principale de l'algorithme. Les données en entrée sont analysées et traitées pour retourner les données de sortie.

Exemple d'énoncé d'algorithme :

- Parmi une sélection de personne, combien sont majeur ?
- Quel est le carré du nombre 42 ?
- Enlever toutes les voyelles de cette phrase.

Les instructions

Il y a différentes instructions :

- Les conditions :
  - Une condition permet de comparer une donnée à quelque chose. Il y a différents opérateurs de comparaison (==, !=, ===, !==, <, <=, > et >=).
  - Une condition peut comparer une à plusieurs données avec les opérateurs logiques (&& et ||).
- Les boucles :
  - Une boucle permet de répéter plusieurs fois une ou plusieurs instructions. On appelle une répétition, une itération.
- Les fonctions :
  - Une fonction est un ensemble d'instructions.
  - On peut passer des paramètres à une fonction.
  - Une fonction peut retourner une valeur.
- Les autres instructions :
  - Il existe de nombreuses autres instructions (selon le langage informatique employé) qui serviront pour des cas spécifiques. (tri, manipulation de donnée, etc...)

Les instructions : Les blocs d'instructions

Les conditions, les boucles et les fonctions sont des blocs d'instructions.

Généralement les blocs sont délimités par des accolades ({…}).

Il est possible d'avoir un ou plusieurs bloc d'instructions dans un bloc d'instructions. On parle alors de bloc parent et de bloc enfant.

```
{
//bloc parent
   {
   // bloc enfant
   }
}
```

Exemple schématique :

- Parmi une sélection de personne, combien sont majeur ?

```
Boucle sur la sélection de personne {
   Condition si l'age est supérieur à la majorité {
      Incrémentation du nombre de personne majeure
   }
}
```

Les instructions : Les conditions

Il existe plusieurs structures conditionnelles :

if...else (Si...sinon) :

```
var age = 42;
if (age >= 18) {
    console.log('Cette personne est majeure');
}
else {
    console.log('Cette personne est mineure');
}
```

Écriture schématique :

```
varAge = 42
Si varAge est supérieur ou égal à 18 {
    Affichage de 'Cette personne est majeure'
}
Sinon {
    Affichage de 'Cette personne est mineure'
}
```

Les instructions : Les conditions

Il existe plusieurs structures conditionnelles :

else if (sinon si) :

```
var date = 'demain';
if (date == 'hier') {
    console.log('On est dans le passé');
}
else if (date == 'maintenant') {
    console.log('On est dans le présent');
}
else {
    console.log('On est dans le futur');
}
```

Écriture schématique :

```
varDate = 'demain'
Si varDate est égal à 'hier' {
    Affichage de 'On est dans le passé'
}
Sinon si varDate est égal à 'maintenant' {
    Affichage de 'On est dans le présent'
}
Sinon {
    Affichage de 'On est dans le futur'
}
```

Il existe plusieurs méthodes pour faire des boucles :

while (Tant que) :

```
var seconde = 0;
while (seconde < 60) {
    // Tant que la variable seconde est inférieur à 60, on incrémente la variable.
    seconde++;
}
// seconde vaut 60
```

Écriture schématique :

```
varSeconde = 0
Boucle tant que varSeconde est inférieur à 60 {
    Incrémentation de varSeconde
}
```

Il existe plusieurs méthodes pour faire des boucles :

for (Pour) :

```
// for (initialisation ; condition ; incrémentation)
for (var seconde = 0 ; seconde < 60 ; seconde++) {
    // Tant que la variable seconde est inférieur à 60, on incrémente la variable.
    // A chaque itération, on affiche la seconde
    console.log(seconde);
}
// => 0 à 59
```

Écriture schématique :

```
varSeconde = 0
Boucle pour que varSeconde s'incrémente tant que varSeconde est inférieur à 60 {
    Affichage de varSeconde
}
```

Les instructions : Les boucles

Il existe plusieurs méthodes pour faire des boucles :

for...of (Pour...de) : Avec une variable tableau

```
var tableau = ['Hello', 'world'];
for (var item of tableau) {
    // A chaque itération, on affiche l'item
    console.log(item);
}
// => Hello
// => world
```

Écriture schématique :

```
varTableau = ['Hello', 'world']
Boucle pour chaque varItem de varTableau {
    Affichage de varItem
}
```

Il existe plusieurs méthodes pour faire des boucles :

for...of (Pour...de) : Avec une variable objet

```
var objet = {
    ville: 'Paris',
    departement: 75
};
for (var key of Object.keys(objet)) {
    // A chaque itération, on affiche l'item avec la key
    console.log(objet[key]);
}
// => Paris
// => 75
```

Écriture schématique :

```
varObjet = [ville: 'Paris', departement: 75]
Boucle pour chaque varKey de varObjet {
    Affichage de varObjet[varKey]
}
```

Les instructions : Les boucles

Il existe plusieurs méthodes pour faire des boucles :

for...of (Pour...de) : Avec une variable chaîne de caractères

```
var chaine = 'Hello';
for (var item of chaine) {
    // A chaque itération, on affiche l'item
    console.log(item);
}
// => H
// => e
// => l
// => l
// => o
```

Écriture schématique :

```
varChaine = 'Hello'
Boucle pour chaque varItem de varChaine {
    Affichage de varItem
}
```

Les instructions : Les fonctions

Pour créer une fonction on utilise function, return permet de retourner une valeur :

```
function addition(param1, param2) {
    return param1 + param2
}
// On passe 2 paramètres à la fonction addition et on affiche la valeur retournée par la fonction addition
console.log(addition(40, 2));
// => 42
```

Écriture schématique :

```
Fonction fonctionAddition (param1 et param2) {
    Retourne la valeur de l'Addition de param1 et param2
}
Affichage de fonctionAddition(40, 2)
```

Javascript

Javascript est un langage de programmation de script interprété orienté objet.

Principalement utilisé pour le web côté client et/ou serveur.

```
<html>
...
<body>
    ...
    <script>
        // instruction;
    </script>
    <script src="./monFichier.js"></script>
</body>
</html>
```