Programmation : principes de base

SUP DE PUB

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- Types et variables
- Instructions
- Branchements conditionnels
- Schémas itératifs
- Procédures et fonctions

Sommaire

- **Ordinateur : architecture et codage**
- Langages de programmation
- Types et variables
- Instructions
- Branchements conditionnels
- Schémas itératifs
- Procédures et fonctions

Ordinateur : architecture globale

- Un ordinateur est constitué de :
  - Un (ou plusieurs) processeur (CPU)
  - De la mémoire : RAM
  - De stockage des données : disque dur
  - Des périphériques :
    - D'entrée : clavier, souris
    - De sortie : écran, imprimante
    - D'entrée-sortie : clé USB
- Un ordinateur est doté d'un système d'exploitation qui fait l'interface avec les programmes : Windows, Linux, MacOS

Ordinateur : fonctionnement

- Processeur :
  - Cadencé par un cristal de quartz
  - Fréquence : nombre d'impulsions par seconde
  - Accomplit des instructions :
    - Opération
    - Opérandes

Ordinateur : fonctionnement

- Familles d'instructions :
  - Accès à la mémoire
  - Opérations arithmétiques : +, -, *, /
  - Opérations logiques : and, or
  - Contrôle : if, while

Ordinateur : codage des données

- Les données sont représentées en binaire : puissances de 2, représentées par 0 ou 1
- Par exemple :

123 = 64 + 32 + 16 + 8 + 2 + 1

123 = 2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0

123 ==> 1111011

Ordinateur : codage des données

- Les données sont traitées par « paquets » : un ordinateur 32 bits traite 32 unités binaires à la fois
- L'unité de base est l'octet : 8 bits
- Avec un octet, on peut représenter :
  - Un nombre compris entre 0 et 255
  - Une lettre en ASCII (65 ==> A, 66==> B...)
  - etc.

Ordinateur : codage hexadécimal

- L'écriture binaire est très verbeuse
- On regroupe les bits par paquets de 4 pour compter en puissances de 16 :
  1 ==> 1, 2 == 2, ... 10 ==> A, ... 15 ==> F
- Par exemple :

  123 ==> 0111 1011
  123 ==> 7B

Codage : à retenir

- On n'a généralement pas besoin de faire les conversions pour programmer
- Par contre, on doit connaître les limites (par exemple 255 est le nombre maximal qu'on peut coder avec un octet)
- L'hexadécimal est parfois utile : en HTML les composantes RGB d'une couleur sont codées sur 3 octets en hexadécimal

Sommaire

- Ordinateur : architecture et codage
- **Langages de programmation**
- Types et variables
- Instructions
- Branchements conditionnels
- Schémas itératifs
- Procédures et fonctions

Programmation : principes

- Plusieurs types de langages de programmation :
  - Langage machine
  - Langage de plus haut niveau
- Un programme écrit dans un langage de haut niveau doit être « traduit » :
  - Interprété : traduction au fur et à mesure
  - Compilé : traduction complète avant exécution

Exemples de langages

- Langages interprétés :
  - Matlab
  - PHP
- Langages compilés :
  - Pascal
  - C
  - Java (semi-compilé)

Langages : autre typologie

- Selon la manière de poser et résoudre le problème :
  - Langages procéduraux : Matlab, C, PHP
  - Langages objet : C++, Java
  - Langages formels : CAML
  - Etc.
- En première année, nous nous intéressons uniquement aux langages procéduraux !

Langages procéduraux

- Des suites d'instructions élémentaires sont regroupées en procédures ou fonctions
- L'enchaînement global est réalisé dans une procédure d'entrée
- Les instructions sont exécutées les unes à la suite des autres, sauf si :
  - Condition d'exécution : if
  - Répétition d'exécution : for, while

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- **Types et variables**
- Instructions
- Branchements conditionnels
- Schémas itératifs
- Procédures et fonctions

Les données du programme

- Un programme effectue des opérations sur des données :
  - Opérations arithmétiques
  - Autres opérations de transformation (ex : sur les chaînes de caractères)
  - Opérations logiques : comparaison
- Ces données sont :
  - Des constantes : invariables
  - Des variables

Typage

- Le programme doit connaître le type de données manipulées
- Ce typage peut être :
  - Explicite : Pascal, C
  - Implicite : Matlab, PHP
- Un typage implicite signifie que le langage détermine le type automatiquement en fonction des données affectées dans la variable sinon il faut « déclarer » la variable

Les variables

- Une variable est un moyen de représenter et manipuler une donnée d'un programme
- Une variable correspond à un espace de la mémoire de l'ordinateur
- Il peut y avoir des « conventions de nommage » : par exemple le nom d'une variable commence par $ en PHP

Affectation des variables

- Une variable n'a de sens que si elle contient une donnée
- Le fait de mettre une valeur dans une variable s'appelle l' « affectation »
- On peut affecter dans une variable :
  - Une valeur (ex : 2, "toto")
  - Une autre variable
  - Le résultat d'une opération

Affectation des variables

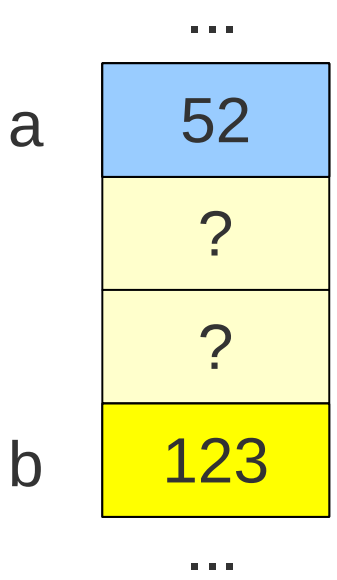
- En général, c'est le signe = qui est utilisé (:= en Pascal)
- L'affectation se fait toujours de la droite vers la gauche !!

```
$a = 123

$b = $a

$a = 52

54 = $a
```



Nombres et chaînes de caractères

- La conversion d'un nombre en chaîne de caractère est généralement implicite
- Par contre, la conversion d'une chaine de caractères en nombre demande généralement l'utilisation d'une fonction du langage
- Ainsi :
  2 + 1 ==> 3
  "2" + "1" ==> "21"
- Cependant, en PHP l'opérateur de concaténation (.) est distinct de l'addition (+)

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- Types et variables
- **Instructions**
- Branchements conditionnels
- Schémas itératifs
- Procédures et fonctions

Enchaînement des instructions

- Par convention, on met une instruction par ligne
- Un séparateur peut être nécessaire : souvent le ;
- Les instructions peuvent être regroupées dans des blocs : une variable déclarée dans un bloc n'existe pas à l'extérieur de celui-ci (notion de « portée » d'une variable)

Quelques instructions en PHP

```
$a = 12; // Les noms de variables
         // commencent par $ en PHP
{ // Début de bloc

   $b = $a * $a;
   echo "Carré de $a = $b";

} // Fin de bloc

echo "Carré de $a = $b"; // Impossible !
```

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- Types et variables
- Instructions
- **Branchements conditionnels**
- Schémas itératifs
- Procédures et fonctions

Branchements conditionnels

- Les branchements conditionnels permettent de modifier l'exécution du programme selon les valeurs des variables
- Typiquement : si condition
  alors bloc vrai
  sinon bloc faux

Branchements conditionnels

- La condition peut être une variable ou toute opération qui renvoie vrai ou faux (comparaison arithmétique par exemple)
- Le bloc vrai est exécuté quand la condition est vrai, sinon c'est le bloc faux
- Le bloc faux est facultatif

Conditions : à savoir

- L'égalité doit se distinguer de l'affectation : généralement == au lieu de =
- Négation : ~ en Matlab, ! en PHP
- On peut combiner plusieurs conditions :
  - opérateurs and, or, xor
  - emploi des parenthéses possible
  - ne pas oublier que (notation PHP) :
    - ! (A and B) = !A or !B
    - ! (A or B) = !A and !B

Exemple : calcul de racine

```
// On suppose une suite d'instruction
// ayant conduit à ce que la variable
// $a contienne un nombre réel

if ($a >= 0) {
   $r = sqr($a);
   echo "Racine de $a : $r";
}
else {
   echo $a . " n'est pas un nombre
      positif, pas de calcul de racine !";
}
```

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- Types et variables
- Instructions
- Branchements conditionnels
- **Schémas itératifs**
- Procédures et fonctions

Schémas itératifs

- Les schémas itératifs permettent de répéter l'exécution d'un bloc d'instructions
- 2 grands types de schémas :
  - schéma for :
    lorsque le nombre de répétitions est connu à l'avance, gère intrinséquement un compteur
  - schéma while :
    lorsque la fin des répétitions dépend d'une condition logique, ne gère pas de compteur

Schémas itératifs

- Si on souhaite avoir un compteur dans un schéma while, il faut le gérer soi-même et penser à le modifier à chaque itération
- Un schéma for peut toujours être transformé en schéma while, mais la réciproque n'est pas vraie

Exemple : calcul de factorielle

```
// On suppose une suite d'instruction
// ayant conduit à ce que la variable
// $a contienne un nombre entier positif

$f = 1;
for ($i=1; $i<=$a; $i++) {
   $f = $f*$i;
}


$f = 1;
$c = 1;
while ($c <= $a) {
   $f = $f * $c;
   $c = $c + 1;
}
```

Les collections

- Il arrive souvent qu'on ait à gérer un grand nombre de données similaires et à leur appliquer des traitements similaires
- On peut alors gérer ces données dans des collections qui permettront de facilement les manipuler dans des itérations
- 2 grands types de collection :
  - les tableaux : indicés à partir de 0 ou 1
  - les tableaux indexés : on peut alors choisir de faire correspondre une valeur à un index choisi

Exemple de tableau

- Remplir puis afficher un tableau des 10 premiers multiples strictements positifs de 3 :

```
for ($i=1; $i<=10; $i++) {
   $multiples[$i] = $i * 3;
}
foreach ($multiples as $index => $valeur) {
   echo "$index multiple : $valeur\n";
}
```

Exemple de tableau indexé

- On associe les capitales aux pays :

```
$capitales["France"] = "Paris";
$capitales["Royaume Uni"] = "Londres";
$capitales["Allemagne"] = "Berlin";
foreach ($capitales as $index => $valeur) {
   echo "$index a pour capitale $ville\n";
}
```

Sommaire

- Ordinateur : architecture et codage
- Langages de programmation
- Types et variables
- Instructions
- Branchements conditionnels
- Schémas itératifs
- **Procédures et fonctions**

Procédure et fonctions

- Les procédures et fonctions permettent l'organisation et la réutilisation du code
- Contrairement à une procédure, une fonction renvoie un résultat (plusieurs possibles avec Matlab)
- Les procédures et fonctions peuvent prendre des paramètres d'entrée
- Bien sûr les variables déclarées dans une procédure ou fonction n'existent pas en dehors de celle-ci

Gestion des variables

- Soit une fonction pour calculer la factorielle d'un nombre entier positif :

```
function factorielle($nombre) {
  $resultat = 1;
  for ($i = 2; $i <= $nombre ; $i++) {
     $resultat = $resultat * $i;
  }
  return $resultat;
}
```

Gestion des variables

- Paramètre d'entrée : `$nombre`
  `$nombre` contiendra ce qu'on passe au moment de l'appel de la fonction : directement un nombre ou une autre variable
- Variables locales à la fonction :
  `$resultat`
  `$i`
- Paramètre de retour : la valeur contenue dans `$resultat`

Exemples d'appels

```
$resultat = factorielle(4);
    // $nombre prendra la valeur 4

$b = 5;
$r = factorielle($b);
    // $nombre prendra la valeur de $b :
  5
```

On n'a pas besoin de connaître le nom des variables utilisées dans la fonction !

Seuls comptent leur nombre et l'ordre !