INITIATION A LA PROGRAMMATION
5XPRG

L'étudiant sera capable :

face à un problème énoncé (faisant appel, par exemple, à des variables de différents types, des structures conditionnelles, des structures itératives) dans un langage simple et pouvant être résolu par un ordinogramme/algorithme,

- d'exploiter la notion de variable ;
- d'identifier et d'associer les différents type de variables et leurs contenus ;
- d'utiliser les variables dans des opérations (affectation, déclaration, opérations mathématiques, concaténation,...) ;
- d'utiliser et de représenter sous forme schématique (organigramme/algorithme) les différentes structures nécessaires à la résolution d'un problème (structures conditionnelles, itératives,..) ;
- d'identifier et utiliser les notions de procédures et de fonctions ;
- de mettre en œuvre une méthodologie permettant la résolution du problème (observation, résolution, expérimentation, validation) ;
- de réaliser l'ordinogramme/algorithme du programme permettant la résolution du problème ;
- de traduire un algorithme en langage littéraire ;
- de traduire un ordinogramme/algorithme simple de manière correcte et appropriée dans un langage donné en respectant sa syntaxe spécifique ;
- de recourir à bon escient de la documentation disponible.

CAPACITES TERMINALES

Pour atteindre le seuil de réussite, l'étudiant sera capable :

à partir d'un ou plusieurs problèmes énoncés faisant appel, par exemple, à des variables de différents types, des structures conditionnelles, des structures itératives, … de manière claire et concise en langue française,

- de construire des représentations (algorithme/ordinogramme, pseudo-code,…) nécessaires à la résolution du ou des problèmes énoncés ;
- de traduire un ou plusieurs algorithmes/ ordinogrammes dans un langage spécifié.

Pour la détermination du degré de maîtrise, il sera tenu compte des critères suivants :

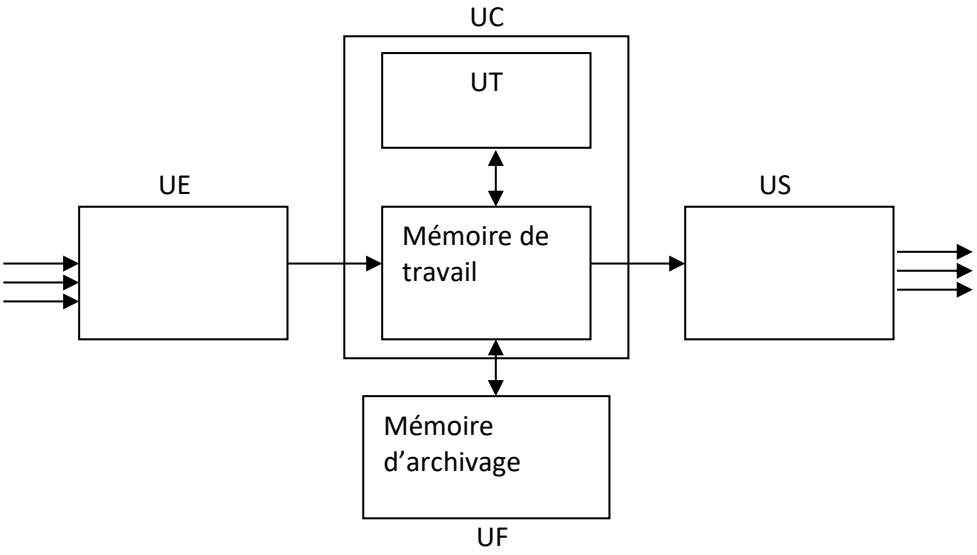
- le niveau d'efficience des solutions proposées pour la résolution du ou des problèmes posés,
- le niveau de cohérence de la traduction dans le respect de la syntaxe spécifique,
- le niveau de lisibilité du code (indentation, nomenclature, …) et des commentaires.

Table des matières

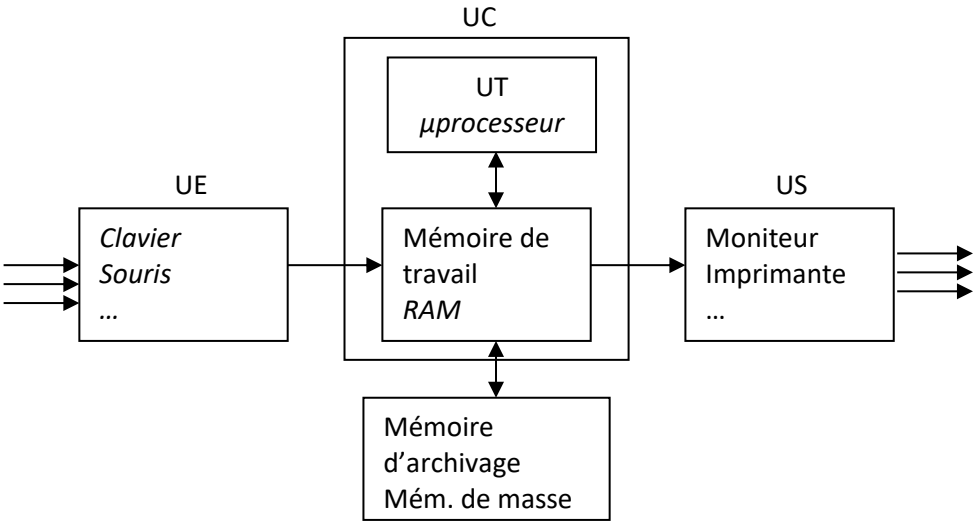
1. Introduction

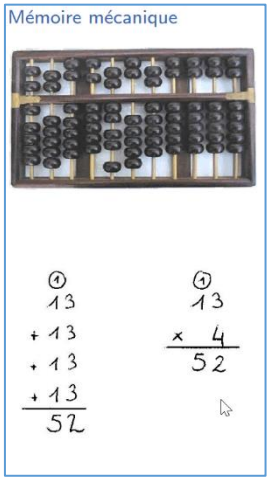
Le schéma du traitement rationnel de l'information

(inspiré de von Neuman, mathématicien d'origine hongroise, 1903-1957, un des pères du calculateur E.N.I.A.C.)

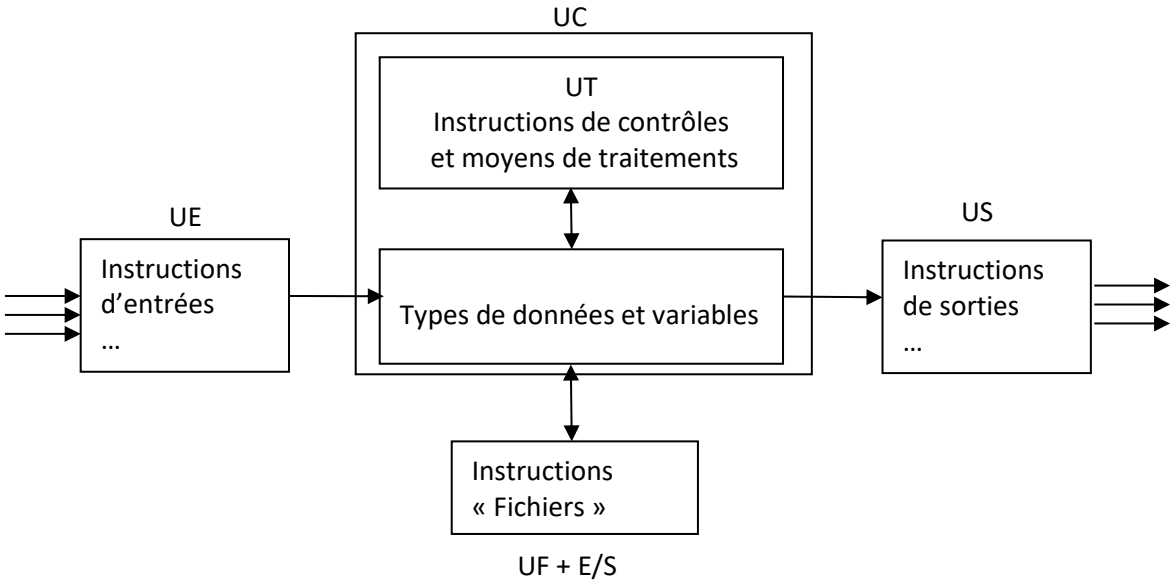


Application du schéma au système informatique

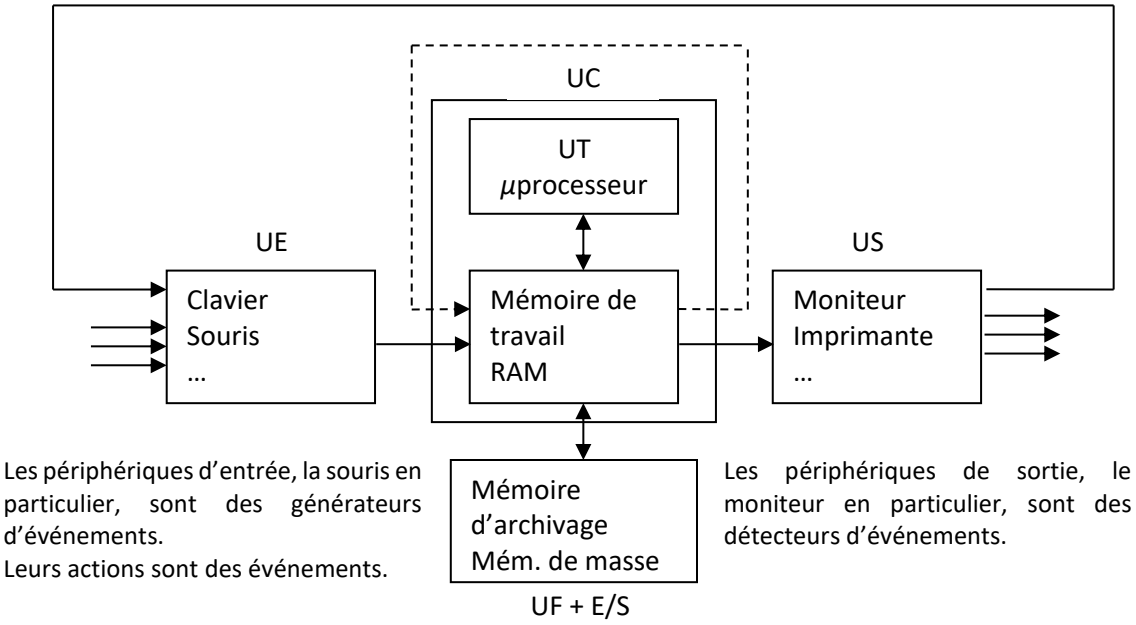




Application du schéma aux langages informatiques



Extension du schéma à la programmation événementielle



Les périphériques d'entrée, la souris en particulier, sont des générateurs d'événements.
Leurs actions sont des événements.

Les périphériques de sortie, le moniteur en particulier, sont des détecteurs d'événements.

Bit : Plus petite unité de mesure en informatique, 0 ou 1
Bytes : Octets en français, il s'agit d'une suite de 8 bits

1 kilo-octet (ko ou Ko) = 2^{10} octets = 1 024 octets (et pas 1 000 octets comme on pourrait le supposer), soit 2 à la puissance 10.
1 méga-octet (Mo) = 2^{20} octets = 1 024 ko = 1 048 576 octets.
1 giga-octet (Go) = 2^{30} octets = 1 024 Mo = 1 073 741 824 octets.
1 téra-octet (To) = 2^{40} octets = 1 024 Go = 1 099 511 627 776 octets.
1 péta-octet (Po) = 2^{50} octets = 1 024 To = 1 125 899 906 842 624 octets.
1 exa-octet (Eo) = 2^{60} octets = 1 024 Po = 1 152 921 504 606 846 976 octets.
1 zetta-octet (Zo) = 2^{70} octets = 1 024 Eo = 1 180 591 620 717 411 303 424 octets.
1 yotta-octet (Yo) = 2^{80} octets = 1 024 Zo = 1 208 925 819 614 629 174 706 176 octets.

2. Introduction au PHP

Dans le cadre de ce cours d'initiation à la programmation, nous utiliserons le php, car :
« Le PHP est le langage serveur le plus utilisé.

Ce qui fait de lui, lorsqu'on est débutant dans le développement web, le langage parfait pour l'apprentissage du développement côté back-end.

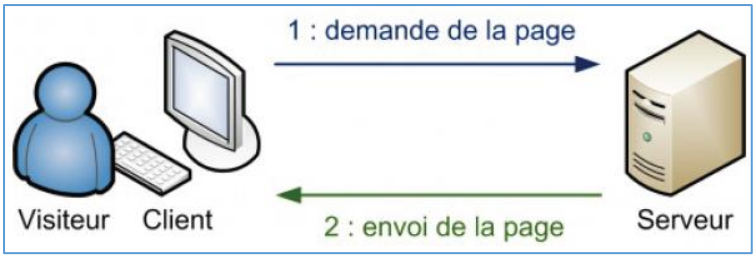
> *Utilisé dans 78,4% des sites web incluant du back-end, ce langage est donc le plus utilisé pour ce type de développement. Bien que cette statistique soit un peu biaisée – car PHP est le langage utilisé dans la plupart des CMS et sites e-commerce (WordPress, PrestaShop, etc.).*

De plus, si vous comptez vous spécialiser dans le développement de CMS et de sites e-commerce, PHP est évidemment le langage web qu'il faut connaître, au même titre que MySQL, HTML et JavaScript.

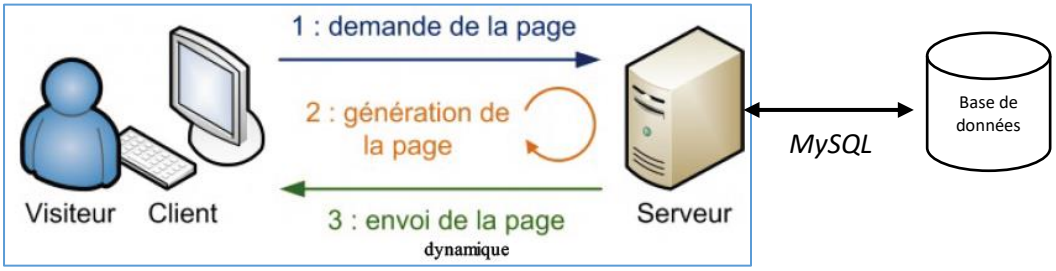
C'est également un bon choix pour tout développeur souhaitant devenir freelance. En effet, la demande en développeurs PHP, surtout ceux maitrisant un framework comme Symfony ou Laravel, est toujours croissante. »[1]

2.1 Langage dynamique

Les sites statiques : ce sont des sites réalisés uniquement à l'aide des langages HTML et CSS. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés.



Les sites dynamiques : plus complexes, ils utilisent d'autres langages en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster.



Le **PHP** est un langage exécuté par le serveur. Il permet de personnaliser la page en fonction du visiteur, de traiter ses messages, d'effectuer des calculs, etc. Il génère une page HTML.

2.2 Préparer son ordinateur

Il existe des paquetages tout prêts pour Windows. WAMP Server, (LAMP pour linux, MAMP pour mac), XAMPP, EasyPHP, Laragon, etc.
Ces paquetages contiennent :

- **Apache** : c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.
- **PHP** : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP. En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.
- **MySQL** : c'est le logiciel de gestion de bases de données dont je vous ai parlé en introduction. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site). Nous n'en aurons pas besoin immédiatement, mais autant l'installer de suite.

[1] Source https://talks.freelancerepublik.com/langage-php-syntaxe-apprendre/

2.3 Premiers pas avec PHP

- Les balises PHP commence par **<?php** et se termine par **?>** ; c'est à l'intérieur que l'on mettra du code PHP.
- Les commentaires longs s'écrivent en utilisant /* */

```
/*Ceci est un commentaire trèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèè èèèèèèèèèèèèèèèèèèèèèèèèèèèèèèès long*/
```

- Les commentaires courts, sur une ligne s'écrivent //

```
//Ceci est un commentaire court
```

- Toutes les instructions PHP se terminent par un point-virgule.

L'instruction echo

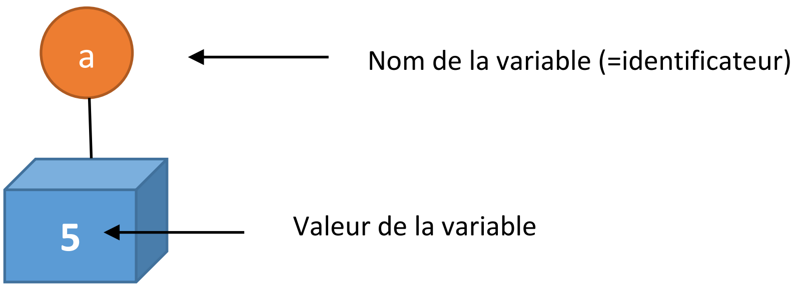
Pour afficher du texte en PHP, on utilise l'instruction echo.

```
< ?php
    echo " Bonjour ";
?>
```

Enregistrer le fichier avec l'extension .php

3. Les variables

Une variable informatique peut être vue comme une boîte avec un nom (étiquette, identificateur) qui peut contenir différentes valeurs.



La variable est un **conteneur** dont le contenu peut changer.

La variable doit être prédestiné à un **type** donné de valeurs.

En informatique, il existe deux grandes familles de valeurs : les valeurs **Numériques** et les valeurs **Alphanumériques**. Chacune de ces familles se subdivise en plusieurs types dont le nombre, les noms et les plages de valeurs sont dépendantes des langages et des systèmes utilisés.

Exemples :

Numérique

Types	*plages(en fonction des langages informatiques)*
Octet	de 0 à 255
Entier	de –32768 à +32767
Réel	de $-3.4*10^{38}$ à $-1.4*10^{-45}$ et de $1.4*10^{45}$ à $3.4*10^{38}$

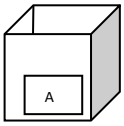
Alphanumérique

Types	*plages(en fonction des langages informatiques)*
Caractère	tout caractère, mais un seul
Alphabétique	un ou plusieurs caractères parmi A … Z et a … z
Chiffre	un ou plusieurs caractères parmi 0 … 9
Chaîne	de 0 à 65535 caractères quelconques

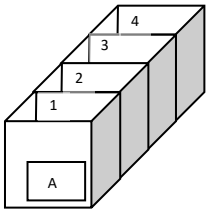
Une variable ne peut contenir qu'**une seule valeur à la fois**. Donner une valeur à une variable revient à remplacer toute autre valeur qu'elle pouvait avoir.

Assemblage de variables

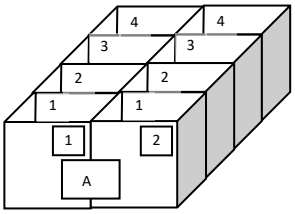
Une variable élémentaire :



Une structure Table à 1 dimension (=Vecteur) regroupant 4 variables élémentaires :



Une structure Table à 2 dimensions (=Matrice) regroupant 8 variables élémentaires :



Les opérateurs :

Opérateurs d'affectation =

Attribuer 5 à la variable « A »
A ← 5

Exemples :
Total = Quantité * Prix_Unitaire
Pi = 3.141592
Adresse = "Rue de la Limite, 6"
Infos = "Les inscriptions aux cours ont lieu du lundi au vendredi, de 19 à 21H. "

Opérateurs de concaténation &

La concaténation de chaînes de caractères, c'est-à-dire leur mise bout à bout
x = "Bon" & "jour"
➔
x = "Bonjour"

Opérateurs arithmétiques

^	élévation à la puissance (x = 2 **^** 3 ➔ x = 8)
*****	multiplication (x = 3 ***** 15 ➔x = 45)
/	division réelle (x = 7 **/** 2 ➔ x = 3.5)
****	division entière (x = 7 **** 2 ➔ x = 3)
%	modulo = reste de la division entière (x = 7 **%** 2 ➔ x = 1)
+	somme (x = 7 **+** 5 ➔ x = 12)
–	soustraction (x = 19 **–** 13 ➔ x = 6)

Opérateurs relationnels (ou opérateurs de comparaison)

<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
= et **==**	égal à
<> et **!=**	différent de

*Les opérateurs relationnels et les opérateurs logiques permettent la définition des **conditions**.*

Opérateurs logiques (table de vérité : 0 = Faux, 1 = Vrai)

NOT et **!**	NEGATION (contraire)
OR et **||**	OU (un OU l'autre OU les deux) (disjonction)
AND et **&&**	ET (les deux) (conjonction)
XOR	OU EXCLUSIF (un OU l'autre, mais pas les deux)

La hiérarchie des opérateurs

Lorsqu'une expression arithmétique ou logique est évaluée, ses composantes prioritaires sont calculées d'abord, les autres ensuite. Bien que tous les langages ne proposent pas le même jeu d'opérateurs, la hiérarchie entre ces opérateurs est généralement toujours la même.

Voici, dans leur ordre décroissant des degrés de priorité, les opérateurs les plus courants :

( )	les parenthèses : leur contenu est toujours évalué en premier.
^	l'opérateur arithmétique d'élévation à une puissance.
* / \ %	les opérateurs arithmétiques de multiplication, de divisions, de modulo.
+ -	les opérateurs arithmétiques d'addition et soustraction.
< =< >= >	les opérateurs relationnels.
AND	l'opérateur logique ET
OR	l'opérateur logique OU

Les règles arithmétiques et algébriques d'associativité, de commutativité et de distributivité sont également d'application en informatique.

Exercice 1 – règles des opérateurs

```
1+2*3= ?
```

Les variables en php

Une **variable**, c'est une information stockée en mémoire **temporairement**. Elle n'a pas une grande durée de vie. En PHP, la variable (l'information) existe tant que la page est en cours de génération. Dès que la page PHP est générée, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien. Ce n'est donc pas un fichier qui reste stocké sur le disque dur mais une petite information temporaire présente en mémoire vive.

- En Php, toutes les variables commencent par **$**
- En Php, les variables ne sont **pas typées** par défaut

Chaque variable contiendra un **nom** : pour pouvoir la reconnaître (*Par exemple age_du_visiteur*) et une **valeur** : c'est l'information qu'elle contient, et qui peut changer. (*Par exemple : 17)*

```
<?php
/**
 * Remarque : les noms de variables ne peuvent contenir ni espace, ni caractères spéciaux, ni accents,
 * à l'exception du underscore (_) qui fait office d'espace.
 */
```

```
// Déclaration d'une variable
```

```
$nom_de_la_variable;
```

```
// Affectation d'une valeur dans une variable
$age = 17; //important! L'affectation se termine comme pour une instruction par un point-virgule.
```

Types de données

String (chaîne de caractères)

```
<?php
$nom_du_visiteur = "toto21";
$nom_du_visiteur = 'toto21';
?>
```

Remarque : si vous voulez insérer un guillemet simple alors que le texte est entouré de guillemets simples, il faut « l'échapper » en insérant un antislash devant. Il en va de même pour les guillemets doubles.

```
<?php
$variable = "Mon \"nom\" est toto21";
$variable = 'Je m\'appelle toto21';
?>
// si l'antislash est omis, PHP vous renvoie une PARSE ERROR.
```

Ou encore, inverser les guillements :

```
<?php
$variable = 'Mon "nom" est Mateo21';
$variable = "Je m'appelle Mateo21";
?>
```

int (nombre entier)

```
<?php
$age_du_visiteur = 17;
?>
```

Le type float (nombre décimal)

Vous devez écrire votre nombre avec un point au lieu d'une virgule.

```
<?php
$poids = 57.3;
?>
```

Le type bool (booléen)

Pour dire si une variable vaut vrai ou faux, vous devez écrire le mot true ou false.

```
<?php
$je_suis_un_zero = true;
$je_suis_bon_en_php = false;
?>
```

Afficher le contenu d'une variable

La fonction **echo** permet d'afficher le contenu d'une variable

```
<?php
$age_du_visiteur = 17;
echo $age_du_visiteur;
?>
```

```
<?php
$age_du_visiteur = 17;
echo "Le visiteur a ";
echo $age_du_visiteur;
echo " ans";
?>
```

Cette instruction peut aussi s'écrire en une seule ligne :

```
<?php
$age_du_visiteur = 17;
echo "Le visiteur a $age_du_visiteur ans";
?>
```

Ou encore :

```
< ?php
$age_du_visiteur = 17;
echo 'Le visiteur a ' . $age_du_visiteur . ' ans';
?>
```

Les tableaux

Il existe 2 sortes de tableaux :

- scalaires quand on travaille avec les indices ou des numéros
- associatif quand on travaille avec le label des champs.

Tableaux scalaires

```
$a[ ] ="titi" ;
$a[ ] ="toto" ;
```

Il n'est pas nécessaire de spécifier l'indice pour le remplissage du tableau, mais bien pour l'affichage :

```
echo $a[0] ;
```

Un tableau numéroté ou à indices est un tableau ou chaque case est identifiée par un numéro. Ce numéro est appelé clé. Attention, un tableau numéroté commence toujours à la case n°0

```
// Initialisation du tableau
$prenoms = array('Mathilde', 'Pierre', 'Amandine', 'Florian');

//Affichage des données
echo "Affichez le prénom qui a pour clé 0 <br>";
echo $prenoms[0] ."<br>";
// affichera Mathilde
echo $prenoms[2] ."<br>";
// affichera Amandine
```

Tableaux associatif

Les tableaux associatifs fonctionnent sur le même principe, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.

```
$tab['nom']=  "Rémi" ;
echo $tab['nom'] ;
```

Vous remarquez qu'on écrit une flèche (=>) pour dire « associé à ».

```
// Initialisation du tableau
$ages = ['Mathilde' => 27, 'Pierre' => 29, 'Amandine' => 21];
```

```
//Affichage des données
echo "Affichez l'âge de Pierre<br >";
echo $ages['Pierre']."<br>";
```

Remarque :
Nous verrons plus loin que lors de l'envoi des données d'un formulaire par la méthode POST, la variable $_POST est un tableau associatif et chaque élément du tableau correspond au nom du champ du formulaire

```
echo $_POST ['nom'] ;
```

Cette instruction permettra de récupérer la valeur du tableau

Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau dont les valeurs peuvent elles-mêmes être des tableaux qui vont à nouveau pouvoir contenir d'autres tableaux et etc.

```
// Initialisation du tableau
$utilisateurs = [
   ['nom' => 'Tom', 'mail' => 'tom@ifosup.wavre.be'],
   ['nom' => 'Pierre', 'mail' => 'pierre@ifosup.wavre.be '],
   ['nom' => 'Amandine', 'mail' => 'amandine@ifosup.wavre.be']
];

//Affichage des données
echo "Affichez le nom de l'utilisateur qui a pour clé le nombre 2 <br>";
echo $utilisateurs[2]['nom']. '<br>';
// affichera Amandine
echo $utilisateurs[2]['mail']. '<br>';
// affichera amandine@ifosup.wavre.be
```

Exercice 2 – Les variables en php

- Réalisez un programme qui affiche le message: « Bonjour je suis un script PHP » – exe2.1.php

- Qu'affichera le programme suivant ? - exe2.2.php

```
<?php
$nombre = 2 + 4;
echo $nombre ;
$nombre = 5 - 1;
echo $nombre ;
$nombre = 3 * 5;
echo $nombre ;
$nombre = 10 / 2;
echo $nombre ;
$nombre = 3 * 5 + 1;
echo $nombre ;
$nombre = (1 + 2) * 2;
echo $nombre ;
?>
```

- Qu'affichera le programme suivant ? - exe2.3.php

```
<?php
$nombre = 10;
$resultat = ($nombre + 5) * $nombre;
echo $resultat
?>
```

- Qu'affichera le programme suivant ? - exe2.4.php

```
<?php
$nombre = 10 % 5;
echo $nombre;
$nombre = 10 % 3;
echo $nombre;
?>
```

- Réalisez un programme qui affichera en fonction de deux variables (jour et langue) le jour de la semaine dans la bonne langue. (*Indice* : il faut utiliser un tableau associatif à deux dimensions) – exe2.5.php

4. Les choix ou alternatives

4.1 Condition

Ces instructions sont essentiellement de la forme

```
SI condition_réalisée ALORS action(s)
```

La condition vérifie le résultat d'une expression logique et *action(s)* peut être n'importe quelle instruction, blocs d'instructions, y compris une autre alternative.

Meilleure écriture :

```
Si condition_réalisée Alors
  action
Fin si
```

```
Si condition_réalisée Alors
  action1
Sinon
  action2
Fin si
```

```
Si condition_réalisée Alors
  premier_groupe_d'actions
Sinon
 deuxième_groupe_d'actions
Fin Si
```

Multi-conditions :

```
Si condition1_réalisée Alors
        Si condition2_réalisée Alors
                premier_groupe_d'actions
        Sinon
                deuxième_groupe_d'actions
        Fin Si
        autres_groupes_d'actions_éventuelles
Sinon   Si condition3_réalisée Alors
                troisième_groupe_d'actions
        Sinon
                Si condition4_réalisée Alors
                        quatrième_groupe_d'actions
                Sinon
                        cinquième_groupe_d'actions
                Fin Si
        autres_groupes_d'actions_éventuelles
        Fin Si
        autres_groupes_d'actions_éventuelles
Fin Si
```

Autant de **`Fin Si`** que de **`Si`**
Quand il n'y a pas d'action entre le **`Sinon`** et son *sous*-Si, il vaut mieux utiliser le **`SinonSi`** :

Si *condition1_réalisée* **Alors**
    premier_groupe_d'actions
SinonSi *condition2_réalisée* **Alors**
    deuxième_groupe_d'actions
SinonSi *contion3_réalisée* **Alors**
    troisième_groupe_d'actions
End If

Exemple de conditions en php

```
<?php
$age = 8;
if ($age <= 12)
{
echo "Salut gamin !";
}?>
```

```
$age = 8;
if ($age <= 12) // SI l'âge est inférieur ou égal à 12
{
echo "Salut gamin ! Bienvenue sur mon site !<br />";
}else // Sinon
{
echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer.!<br />";
}
?>
```

```
$age = 8;
if ($age <= 12)
{
    echo "Salut gamin !<br />";
}
elseif ($age<=18)           // Sinon Si…
{
    echo "salut l'ado!<br />";
}
else
{
    echo "salut le vieux!<br />";
}
?>
```

Condition condensée en php

```
 $age = 24;
if ($age >= 18)
{
$majeur = true;
}else
{
$majeur = false;
}
?>
```

Peut s'écrire en condensé :

```
<?php
$age = 24;
$majeur = ($age >= 18) ? true : false;
?>
```

La condition testée est $age >= 18. Si c'est vrai, alors la valeur indiquée après le point d'interrogation (ici true) sera affectée à la variable $majeur. Sinon, c'est la valeur qui suit le symbole « deux-points » qui sera affectée à $majeur.

4.2 Sélection (Selon Que…)

Lorsque les conditions de d'alternatives en cascade concernent différentes valeurs d'une même variable ou expression, il est préférable d'utiliser l'instruction de choix multiple (ou sélection).
Cette instruction est de la forme

```
SELON variable
CAS_OU variable = valeur_1
action(s)_1
CAS_OU variable = valeur_2
action(s)_2
FIN_SELON
```

L'instruction se comporte comme une commande d'aiguillage effectuant la connexion vers la voie appropriée selon la ligne demandée par la valeur de la variable.

Exemple de sélection en php

Voici un exemple avec un if assez lourd à la lecture…

```
$nombre=5 ;
if ($nombre<4){
   echo "le nombre est < que 4";
}elseif ($nombre <7){
          echo "le nombre est >3 et < 7";
         }elseif ($nombre =7){
                  echo "le nombre est 7";
                  }elseif ($nombre =8){
                           echo "le nombre est 8";
                           }elseif ($nombre =9){
                                    echo "le nombre est 9";
                                    }elseif ($nombre >9){
                                             echo "le nombre est >9"
                                             }
```

Celui-ci peut avantageusement être remplacé par un switch : (Selon que)

```
switch($nombre){
case 1:
case 2:
case 3:echo  "le nombre est < que 4";
          break;
case 4:
case 5:
case 6: echo "le nombre est >3 et < 7";
          break;
case 7: echo "le nombre est 7";
           break;
case 8: echo "le nombre est 8";
          break;
case 9: echo "le nombre est 9";
           break;
default: echo "le nombre est >9";
           break;
}
```

Exercice 3 : les conditions

- 3.1 Ecrivez un programme (en utilisant un if) qui, en fonction d'une moyenne affichera le grade obtenu de l'étudiant. – exe3.1.Php

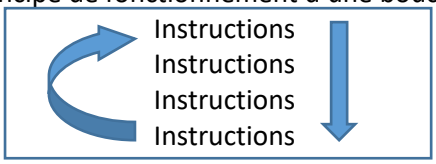
Exemple :

0≤moyenne <10 : refusé
10≤moyenne <14 : satisfaction
14≤moyenne <16 : distinction
16≤moyenne <18 : grande distinction
18≤moyenne <20 : la plus grande distinction

- 3.2 Même question en utilisant un switch – exe3.2.Php

5. Les répétitives

Principe de fonctionnement d'une boucle :



Les boucles permettent la répétition d'une ou plusieurs actions. On distincte la boucle arithmétique, dont la condition d'arrêt est liée à un compteur (compte-tours), des boucles logiques dont les conditions d'arrêt sont exprimées par une expression logique.

La boucle arithmétique est essentiellement de la forme

```
POUR variable VARIANT_DE valeur_initiale A valeur_finale AU_PAS_DE valeur_pas
          REPETER action(s)
```

Les boucles logiques sont essentiellement de la forme

TANT_QUE *condition* REPETER *action(s)*

ou

REPETER *action(s)* JUSQU'À *condition*

Les actions seront exécutées un nombre indéterminé de fois si la condition est préalablement réalisée

5.1 Boucle logique en php

```
<?php
while ($continuer_boucle == true)
{
// instructions à exécuter dans la boucle
}
?>
```

while peut se traduire par « tant que ». Ici, on demande à PHP : TANT QUE $continuer_boucle est vrai, exécuter ces instructions.

Par exemple, afficher les chiffres de 1 à 10 :

```
$chiffre= 1;
while ($chiffre <= 10)
{
echo $chiffre ;
$chiffre ++;            // $chiffre=$chiffre +1
}
?>
```

```
<?php

$nombre_de_lignes = 1;
while ($nombre_de_lignes <= 10) {
   echo 'je dois pratiquer régulièrement le PHP pour progresser.<br />';
   $nombre_de_lignes++; // $nombre_de_lignes = $nombre_de_lignes + 1
}
?>
```

Remarque : Il faut TOUJOURS s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini ! PHP refuse normalement de travailler plus d'une quinzaine de secondes. Il s'arrêtera tout seul s'il voit que son travail dure trop longtemps et affichera un message d'erreur.

5.2 Boucle arithmétique en php

Les actions sont exécutées un nombre prédéterminé de fois.

for et while donnent le même résultat et servent à la même chose : répéter des instructions en boucle.
L'une peut paraître plus adaptée que l'autre dans certains cas…

Le compteur sert à l'**initialisation**. C'est la valeur que l'on donne au départ à la variable

La **condition** : comme pour le while, tant que la condition est remplie, la boucle est réexécutée. Dès que la condition ne l'est plus, on en sort.

L'**incrémentation** vous permet d'ajouter 1 à la variable à chaque tour de boucle.

```
for (compteur; condition; modification du compteur) {
 liste d'instructions
}
```

Exemple :

```
<?php
for ($nombre_de_lignes = 1; $nombre_de_lignes <= 100; $nombre_de_lignes++)
{
echo 'Ceci est la ligne n°' . $nombre_de_lignes . '<br />';
}
?>
```

5.3 La boucle foreach en php

C'est une sorte de boucle for spécialisée dans les tableaux.

```
<?php
$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique','Benoît');

foreach($prenoms as $element)
{
echo $element . '<br />'; // affichera $prenoms[0], $prenoms[1] etc.
}
?>
```

L'avantage de foreach est qu'il permet aussi de parcourir les tableaux associatifs.

```
<?php
$coordonnees = array (
'prenom' => 'François',
'nom' => 'Dupont',
'adresse' => '3 Rue du Paradis',
'ville' => 'Marseille');

foreach($coordonnees as $element)
{
    echo $element . '<br />';
}
?>
```

Toutefois, avec cet exemple, on ne récupère que la valeur. Or, on peut aussi récupérer la clé de l'élément. On doit dans ce cas écrire foreach comme ceci :

```
<?php foreach($coordonnees as $cle => $element) ?>
```

Remarque :

La fonction « print_r » permet d'afficher rapidement un tableau (c'est une sorte d'echo spécialisé dans les arrays).

Exercice 4 : répétitive

- 4.1 Affichez la table de multiplication par 8 (en utilisant une répétitive!) – exe4.1.php
- 4.2 Écrire un programme qui calcule la somme des entiers de 1 à 100 – exe4.2.php
- 4.3 Écrire un programme qui calcule le produit des entiers de 1 à 100 – exe4.3.php

6. Les fonctions

Comme les boucles, les fonctions permettent d'éviter d'avoir à répéter du code PHP que l'on utilise souvent.

Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur. En général, dès que vous avez besoin d'effectuer des opérations un peu longues dont vous aurez à nouveau besoin plus tard, il est conseillé de vérifier s'il n'existe pas déjà une fonction qui fait cela pour vous. Et si la fonction n'existe pas, vous avez la possibilité de la créer.

L'intérêt des fonctions réside dans le fait de clarifier le code et la possibilité de **réemploi** de ces fonctions.
Une fonction peut posséder plusieurs arguments qui seront séparés par des virgules. Une fonction renvoie généralement le résultat d'une demande : un nombre, une chaîne de caractère, ou une valeur booléenne (true, false). Elle doit être placée au début du script.

```
Function NomFonction(Argument1, Argument2)
        {
         liste d'instructions ;
        return $résultat ;
}
```

Les fonctions peuvent être assez complexes. Il est dès lors indispensable de les **commenter** en vue d'une utilisation ultérieure. (Détails pour les arguments, ce que la fonction retourne….)

Appel d'une fonction

```
Nom_De_La_Fonction();
```

Voici la liste des (nombreuses !) fonctions en php :
http://php.net/manual/fr/indexes.functions.php
exemple de fonction php :

Traitement des chaînes de caractères

- strlen : longueur d'une chaîne
- str_replace : rechercher et remplacer
- strtolower : écrire en minuscules
- strtoupper : écrire en majuscules
- …

Récupérer la date

```
H = Heure
i = Minute
d = Jour
m = Mois
Y = Année
```

Pour afficher l'année :

```
<?php
$annee = date('Y');
echo $annee;
?>
```

```
<?php
// Enregistrons les informations de date dans des variables
$jour = date('d');
$mois = date('m');
$annee = date('Y');
$heure = date('H');
$minute = date('i');
// Maintenant on peut afficher ce qu'on a recueilli
echo 'Bonjour ! Nous sommes le ' . $jour . '/' . $mois . '/' . $annee . 'et il est ' . $heure. ' h ' . $minute;
?>
```

Création et utilisation d'une fonction

```
<?php
// Ci-dessous, la fonction qui calcule le volume du cône
function VolumeCone($rayon, $hauteur)
{
```

```
$volume = $rayon * $rayon * 3.14 * $hauteur * (1/3);        // calcul du volume
return $volume;        // indique la valeur à renvoyer, ici le volume
}
$volume = VolumeCone(3, 1);
echo 'Le volume d\'un cône de rayon 3 et de hauteur 1 est de ' .
$volume;
?>
```

Autres exemples de fonctions :

```
<?php
//déclaration de la fonction
function add($x,$y)
{
  $total=$x+$y;
  return $total;
}
//utilisation de la fonction
echo add(1,16);
?>
```

```
<?php
function div($nb1,$nb2)
{
  if ($nb2!=0)
    return $nb1/$nb2;
  else
    return "impossible";
}
//utilisation de la fonction
echo div(5,0)."</br>";
?>
```

```
<?php
function mult($a,$b)
{
  return $a*$b;
}
//utilisation de la fonction
echo mult(10,5)."</br>";
?>
```

Afin de ne pas surcharger le programme de base, il est souhaitable de regrouper les fonctions d'un programme dans un fichier séparer et le nommer lib.inc (=librairie)
A l'aide de l'instruction include ces fonctions peuvent être intégrées directement au programme

```
include ("lib.inc");
```

Exercice 5 : Les fonctions

- 5.1 Créez une fonction qui renvoie une chaîne de caractère en fonction de l'heure ("bonjour", "bon après-midi", "bonsoir", "bonne nuit", …) – exe5.1.php
- 5.2 Créer une fonction factorielle qui calcul la factorielle d'un nombre. – exe5.2.php

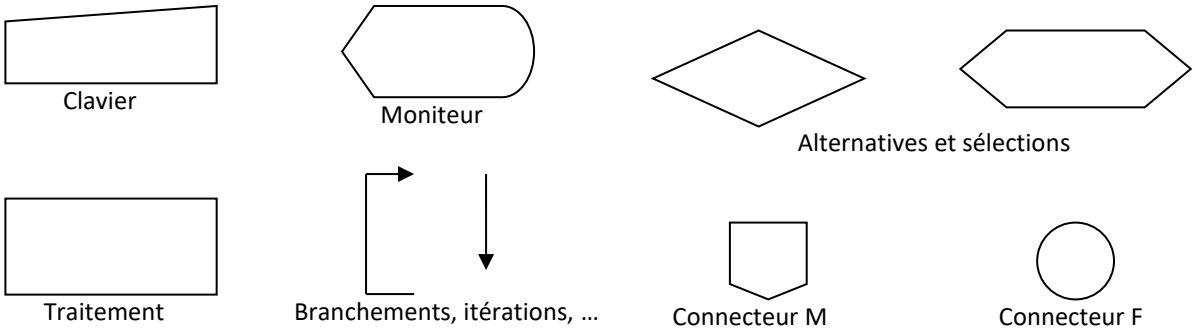
7. Quelques méthodes de représentation d'algorithmes

7.1. Organigramme de programmation

« Un organigramme de programmation (parfois appelé algorigramme, logigramme ou plus rarement ordinogramme) est une représentation graphique normalisée de l'enchaînement des opérations et des décisions effectuées par un programme d'ordinateur. » [2]

7.1.1 Les structures de programmations

Diverses techniques sont utilisées pour schématiser les programmes et les algorithmes qui les composent. Elles présentent toutes l'avantage d'être indépendantes des langages de programmation. Le flow chart, ou ordinogramme, est assez bien connue du profane puisque largement utilisée dans la presse pour l'explication de divers mécanismes (droits, fiscalités, allocations, …). Voici la signification des figures utilisées dans ce support :



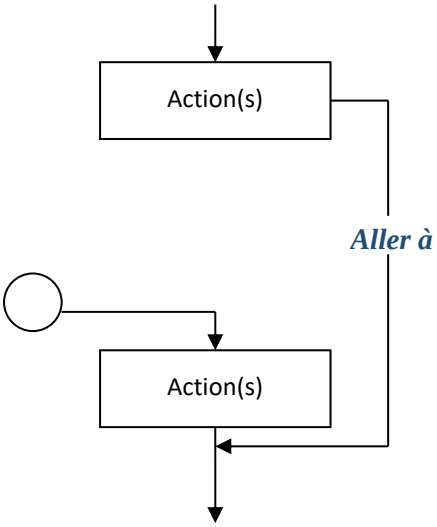
7.1.2 La séquence

Toutes ces instructions s'exécutent l'une après l'autre, successivement, séquentiellement, chaque fois que le programme ou le sous-programme (procédure ou fonction) est lancé.

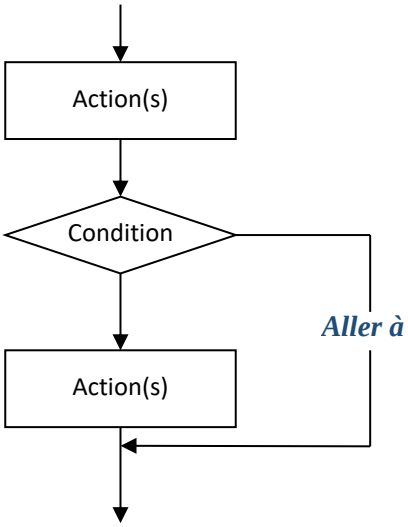
action1;
action2;
action3;
… … …

[2] ©Wikipedia, https://fr.wikipedia.org/wiki/Organigramme_de_programmation

7.1.3 La rupture de séquence (branchement)



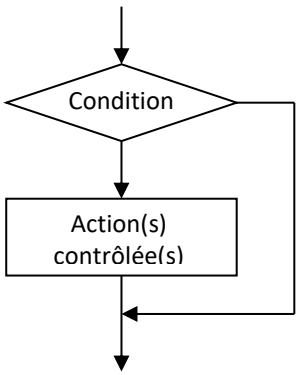
La rupture de séquence inconditionnelle.

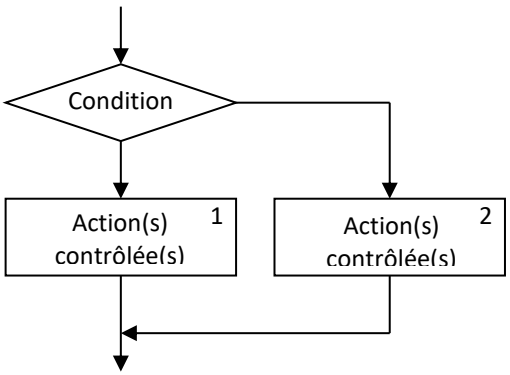


La rupture de séquence conditionnelle.

Bien que présente sous l'une ou l'autre forme dans tous les langages, l'instruction « Aller à » sera toujours évitée par l'usage d'alternatives bien construites ou d'itérations judicieusement organisées.

7.1.4 L'alternative





Le contrôle d'une alternative se réalise toujours par une expression logique. Plusieurs alternatives peuvent être assemblées en cascades ou imbriquées pour réaliser des choix complexes.

Modèles en Php (**if … else**) :

```
if ( condition_réalisée ) action ;

if ( condition_réalisée ) action1 ; else action2 ;
```

Meilleure écriture :

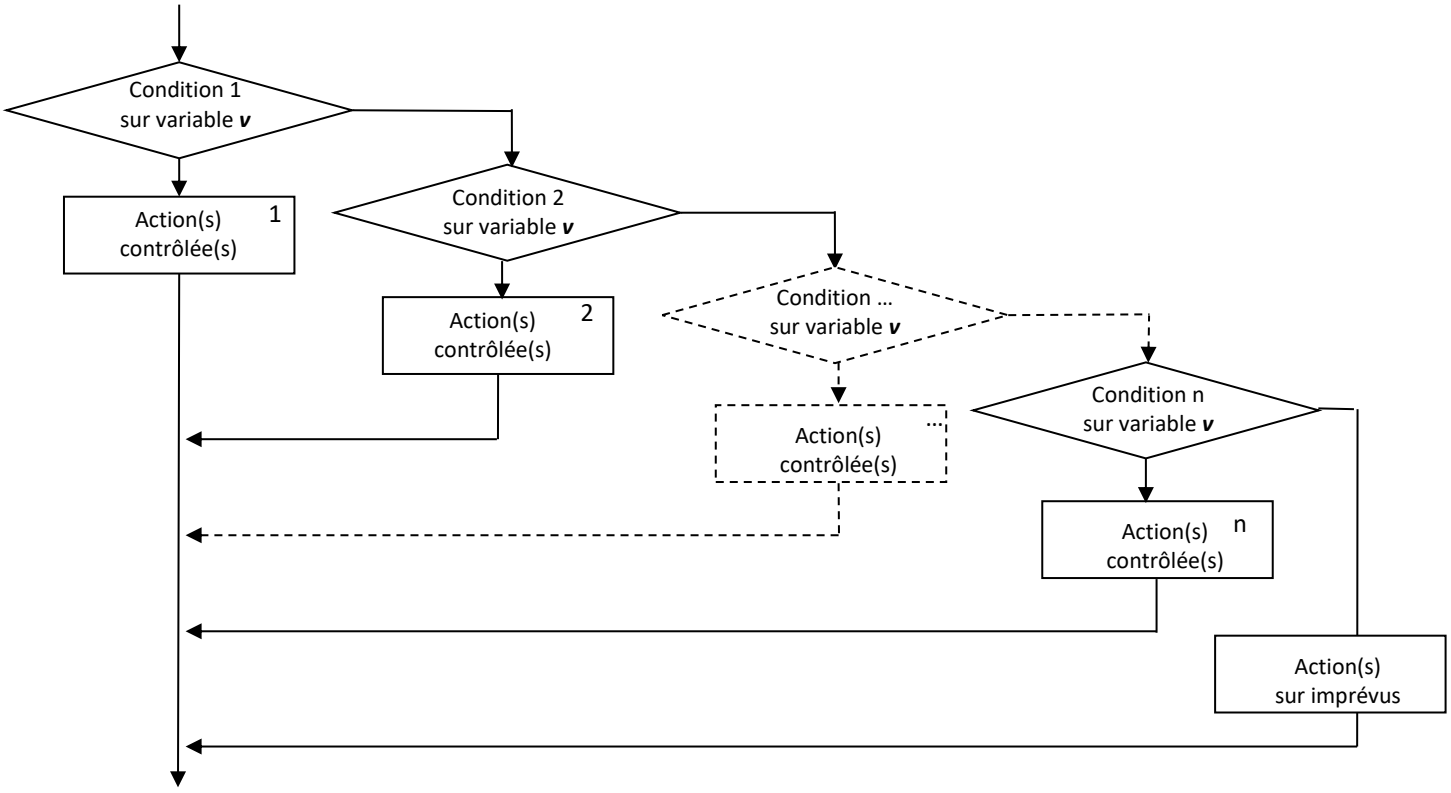
```
if ( condition_réalisée )
  action ;
if ( condition_réalisée )
  action1 ;
else
  action2 ;
```

```
if ( condition_réalisée )
 {
  premier_groupe_d'actions ;
 }
else
 {
 deuxième_groupe_d'actions ;
 }
```

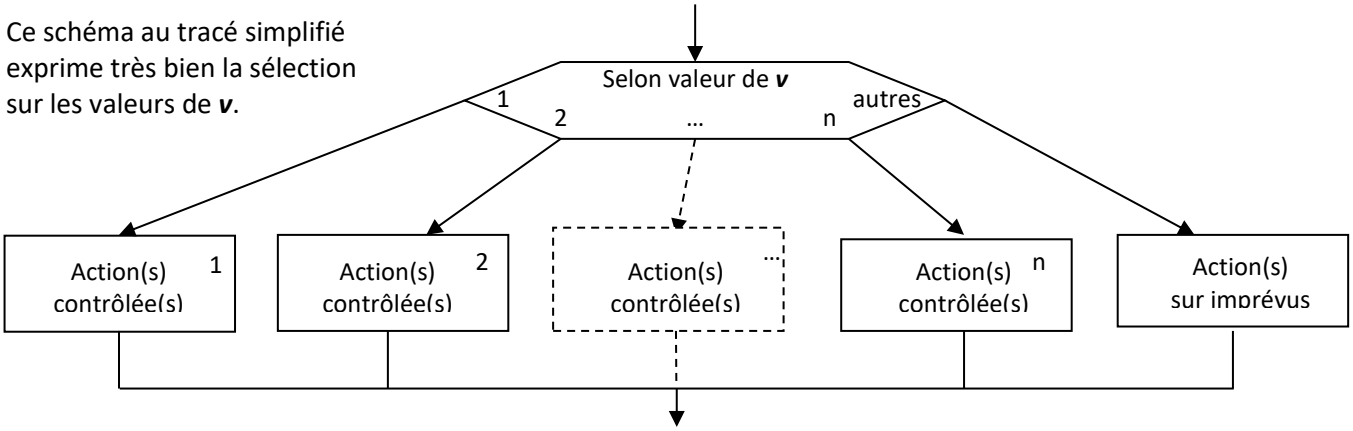
En Php, les accolades sont nécessaires pour encadrer un groupe d'instructions soumises à un même contrôle.
Elles sont superflues lorsqu'une seule instruction est soumise au contrôle.
Dans tous les cas, elles peuvent faciliter la lecture du code par le développeur.

7.1.5 La sélection

La sélection désigne plusieurs alternatives en cascade, chacune vérifiant sa condition sur une seule et même variable commune. La plupart des langages offrent une instruction de sélection.



Ce schéma au tracé simplifié exprime très bien la sélection sur les valeurs de ***v***.



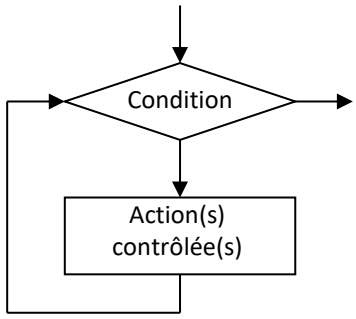
Modèles en Php (**switch … case**) :

```
switch( $variable )
{
case valeur1 :
        {
        premier_groupe_d'actions ;
        break ;
        }
case valeur2 :
        {
        deuxième_groupe_d'actions ;
        break ;
        }
default :
        {
        éventuelles_actions_pour_les_cas_non_prévus ;
        }
}
```

Les accolades d'ouverture et de fermeture d'un bloc case sont facultatives.

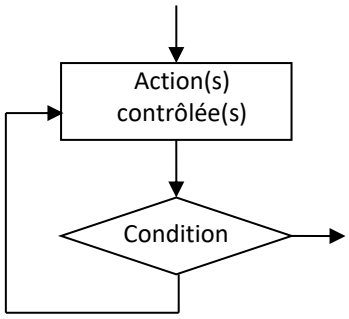
7.1.6 Les itérations

Itération « manuelle » avec contrôle avant le code concerné



Le code contrôlé ne sera peut-être jamais exécuté.

Itération « manuelle » avec contrôle après le code concerné



Le code contrôlé sera exécuté au moins une fois.

Il faut, dans un cas comme dans l'autre, que le code contrôlé contienne une instruction modifiant une ou plusieurs variables participant à la condition de sorte à permettre la sortie de la boucle après un certain nombre d'itérations.

Modèles en Php (**while … do … while**) - Tant que:

```
while ( condition_est_réalisée )
{
        une_ou_plusieurs_actions ;
}
```

```
do
{
        une_ou_plusieurs_actions ;
}
while ( condition_est_réalisée ) ;
```

Les boucles *jusqu'à* n'existent pas en langage Php, mais elles peuvent être réalisées par les boucle *tant que* dont on inverse les conditions.

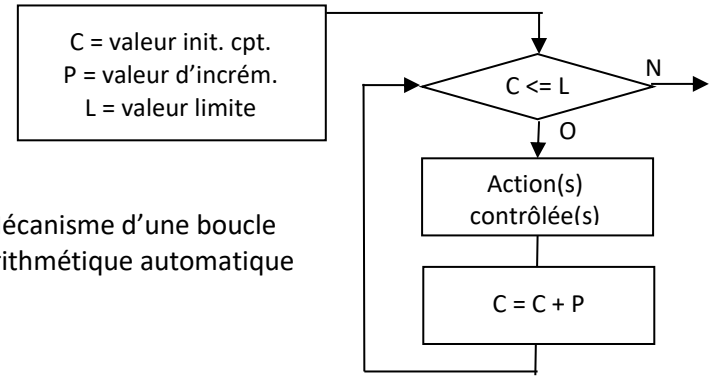
Modèles en Php (**do … while**) - Jusqu'à :

```
while !( condition_est_réalisée )
{
        une_ou_plusieurs_actions ;
}
```

```
do
{
        une_ou_plusieurs_actions ;
}
while !( condition_est_réalisée );
```

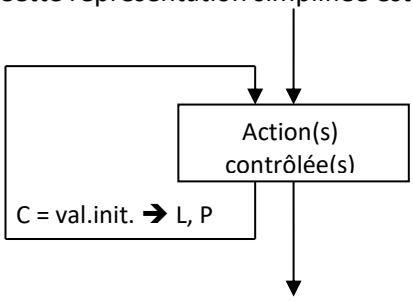
La boucle arithmétique

La boucle arithmétique « automatique » est basée sur un compteur dont la valeur initiale, la valeur limite et le pas d'incrémentation sont prédéfinis. Le code contrôlé est exécuté un nombre de fois prédéterminé.



Mécanisme d'une boucle arithmétique automatique

Cette représentation simplifiée est tout aussi significative.

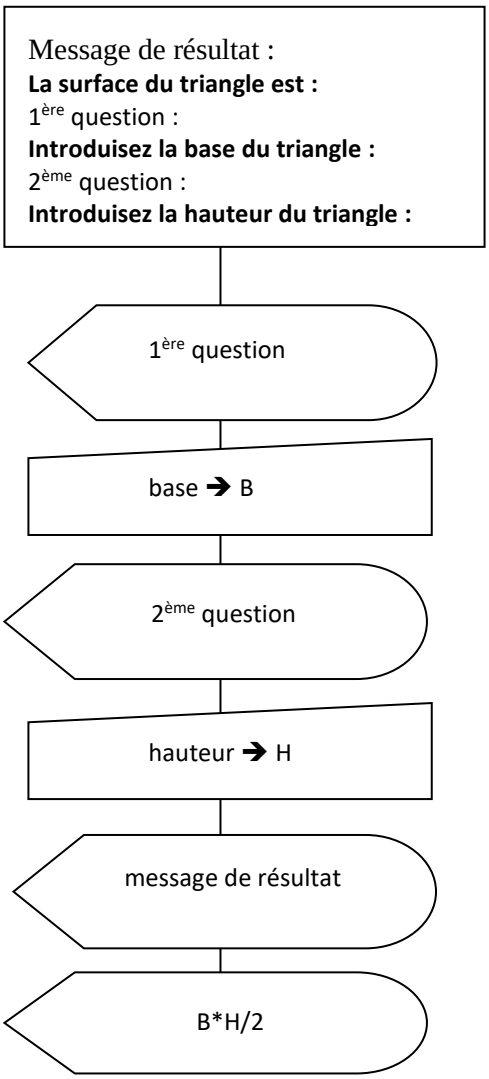


Modèles en Php (**for …**) :

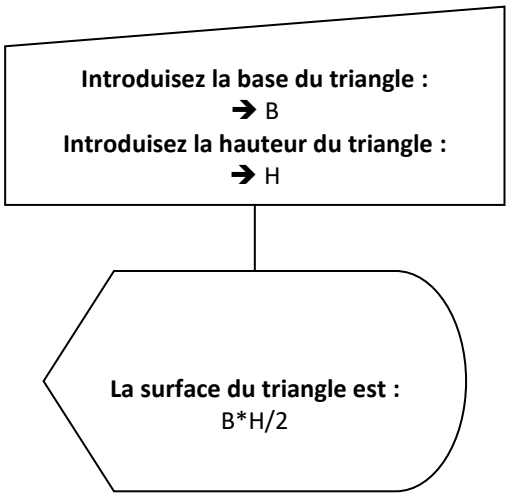
```
for (i = 1 ; i < 11 ; i++ )
{
  action(s)_répétée(s)_10_fois ;
}
```

L'action est recommencée pour chaque valeur de i qui varie au pas d'une unité de 1 à 10.

Exemple d'organigramme complet : programme qui calcul la surface du triangle :



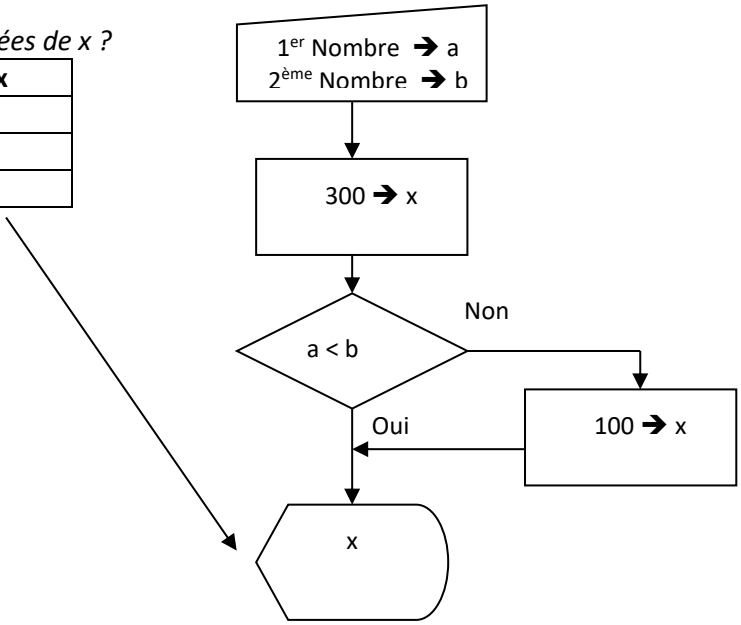
L'usage de raccourcis n'est pas interdit tant que l'information transmise par le flow chart reste évidente. Pour le programme de calcul de la surface d'un triangle, nous pouvons simplifier le schéma comme ceci :



Exercice 6 ordinogrammes / flow chart

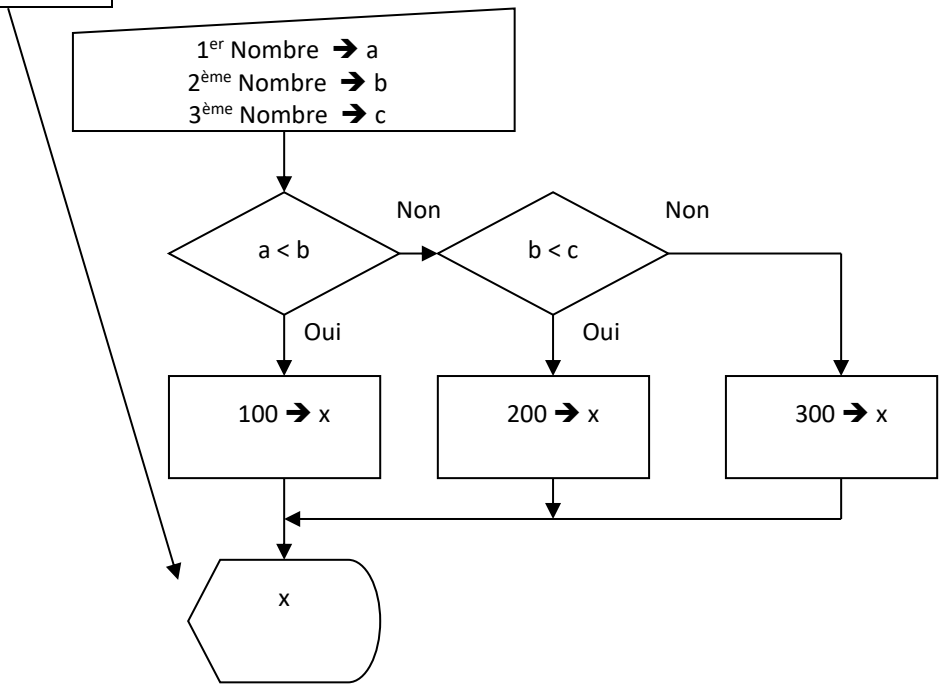
1.Que seront les valeurs affichées de x ?

a	b	x
5	10	
10	5	
7	7	



2.Que seront les valeurs affichées de x ?

a	b	c	x
5	10	15	
15	5	10	
15	10	5	



3. Réaliser l'organigramme de l'exercice 4.2 (Ecrire un programme qui calcule et affiche la somme des 100 premiers entiers positifs.)

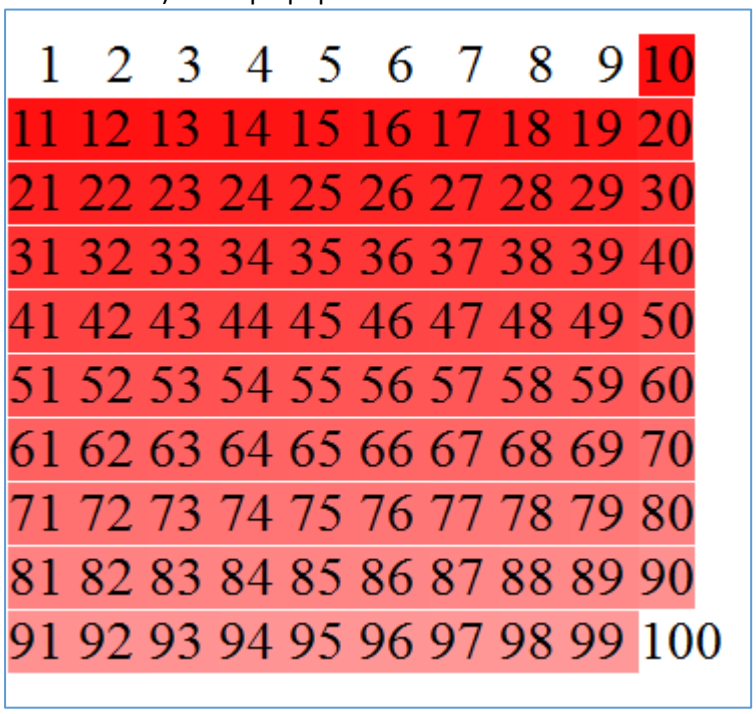
4. Réaliser l'organigramme de l'exercice 4.3 (Ecrire un programme qui calcule et affiche le produit des 100 premiers entiers positifs.)

EXERCICES RECAPITULATIFS

- 1. Affichez les nombres suivant en HTML à l'aide d'une répétitive. – recap1.php

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

- 2. Reprenez l'exercice précédent et affectez une couleur de fond **différente** pour chaque chiffre (utilisez un <SPAN> en HTML) – recap2.php



- 3. Écrire le programme permettant d'afficher la table de multiplication suivante – recap3.php

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

- 4. Afficher les 50 premiers multiples de 5 – recap4.php

5 10 15 20 25 30 35.....250

- 5. Écrire un programme qui affiche les nombres de 20 à 1 de trois en trois. – recap5.php

Exemple :
20 19 18
17 16 15
14 13 12
...

(faire deux fois l'exercice : avec "for" et avec "while") –recap6.php

- 7. Affichez tous les nombres de 3 chiffres qui sont égaux à la somme des cubes de leurs chiffres (difficile !) – recap7.php
- 8. Affichez tous les nombres premiers plus petits que 1000(difficile !) – recap8.php
- 9. Affichez les nombres de 11 à 121 par pas de 11 – recap9.php
- 10. Affichez un « carré » de « O » de 10 sur 10 – recap10.php
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
  O O O O O O O O O O
- 11. Affichez un triangle de « O » - recap11.php
  O
  OO
  OOO
  OOOO
  OOOOO
  OOOOOO
  OOOOOOO
  OOOOOOOO
  OOOOOOOOO
  OOOOOOOOOO
- 12. Affichez un triangle de « O » avec **aléatoirement**[3] des « X » - recap12.php
  O
  O O
  O O O
  O O X O
  X X O O O
  O O O O O O
  O X O O O X O
  O O O O X O X O
  O O O O O O O O O
  O O O O O X O O X O

- 13. Ecrivez un algorithme qui permet d'afficher, à travers une boucle, le compte à rebours: 10,9,8,7,6,5,4,3, 2,1 Bonne année ! - recap13.php

- 14. Vous devez gérer la disponibilité des locaux de l'IFOSUP. - recap14.php
  Créez un tableau qui reprend les données suivantes:
  106: occupé
  108: occupé
  110: disponible
  111: disponible
  Ecrivez un algorithme qui affiche les locaux disponibles[4]

- 15. Même exercice, mais n'affichez que le premier local rencontré disponible. - recap15.php

[3] Utilisez la fonction rand() pour obtenir un nombre aléatoire
[4] Utiliser l'instruction foreach. Documentation disponible ici https://www.php.net/manual/fr/control-structures.foreach.php