# Divers codages d'information en informatique.

## Le bit, l'octet (le byte), le mot, l'hexadécimal.

Dans un ordinateur, l'information est stockée sous forme de tensions électriques. Soit la tension est inférieure à un seuil, soit elle est supérieure à un seuil. Dans une mémoire flash, elle est stockée sous la forme de présence ou d'absence d'électrons dans un condensateur. Sur un disque dur ou une bande magnétique, l'information est stockée dans des minuscules régions sous forme de magnétisation, soit dans un sens, soit dans l'autre sens. Sur un CD-ROM ou un DVD, elle est stockée sous forme de creux et de bosses, qui réfléchissent différemment la lumière. Autrefois, on stockait l'information sur des bandes perforées, soit il y avait un trou, soit il n'y en avait pas.

Dans tous les cas, l'information élémentaire n'a que deux possibilités, que l'on note généralement **0** et **1**.
Cette information élémentaire est appelée un **bit**, abréviation de **BI**nary uni**T**.
On peut donc représenter l'information stockée dans un ordinateur, un disque dur, etc. par une suite de bits, donc une suite de 0 et de 1.

Pour plus de commodité, on a regroupé ces bits en **octets** ou **bytes**. Un **octet** est une suite de 8 bits. En écrivant ces 8 bits sous forme de 8 chiffres valant soit 0 soit 1, cela correspond à écrire un nombre en **notation binaire**. On dit pour cela que toute l'information est stockée sous forme de nombres dans un ordinateur, ce qui n'est qu'une manière pratique de se représenter la mémoire d'un ordinateur.

Un **bit** n'a que deux valeurs possibles, qui sont zéro ou un.
Un **octet** a $2^8 = 256$ valeurs possibles, qui vont de 0000 0000 à 1111 1111 (0 à 255).

On regroupe souvent les octets par pairs pour former des **mots de 16 bits**.
Un **mot de 16 bits** a $2^{16} = 65'536$ valeurs possibles.
On regroupe souvent les octets par quadruplets pour former des **mots de 32 bits**.
Un **mot de 32 bits** a $2^{32} = 4'294'967'296$ valeurs possibles, soit environ 4 Giga valeurs.
La limitation de mémoire de 4 Giga octets des systèmes d'exploitations de 32 bits vient de là.
On regroupe encore les octets par série de huit, pour former des **mots de 64 bits**.
Un **mot de 64 bits** a $2^{64}$ valeurs possibles, soit environ $16 \cdot 10^{18}$ valeurs.
Les systèmes d'exploitations 64 bits ne sont pas limités par la taille des mots de 64 bits.

Pour noter un octet ou un mot de 16 ou 32 bits, la notation binaire est désagréable. Pour cela on utilise la notation **hexadécimale**, qui regroupe des bits par série de 4, pour former des nombres allant de 0 à 15.

```
0000 = 0    0001 = 1    0010 = 2    0011 = 3    0100 = 4    0101 = 5    0110 = 6    0111 = 7
1000 = 8    1001 = 9    1010 = A    1011 = B    1100 = C    1101 = D    1110 = E    1111 = F
```

En notation hexadécimale, six "chiffres" sont ajoutés, que l'on note A, B, C, D, E et F.
Ainsi, un octet est noté à l'aide de deux nombres hexadécimaux.
0000 0000 = #00 Le # est une manière d'indiquer la numérotation hexadécimale.
1111 1111 = #FF

*Exercice 1*
Complétez les notations binaires et hexadécimales :

```
01011101  =                    10000111  =

          = #46                          = #7A

          = #CC                          = #B0
```

*Exercice 2*
Complétez...

Un mot de 16 bits est représenté par ____ chiffres hexadécimaux.

Un mot de 32 bits est représenté par ____ chiffres hexadécimaux.

#FA18 représente un mot de ____ bits.

Le plus grand mot de 32 bits est noté en hexadécimal par _____

## Codage des couleurs

Puisque l'œil humain est composé de 3 sortes de capteurs de couleurs, sensibles soit aux couleurs voisines du rouge, soit du vert, soit du bleu, on peut coder presque toutes les couleurs en indiquant la proportion de rouge, de vert et de bleue.
Une image sur un écran d'ordinateur ou de télévision est constitué de **pixels** (PIXture ELements).
Chaque pixel a une couleur, déterminée par la quantité de rouge, de vert et de bleu. C'est le système **RVB** ou **RGB** (Red Green Blue en anglais).
Chacune des trois quantités est déterminée par un octet, donc la valeur comprise entre 0 et 255.
Donc un pixel est représenté par un nombre hexadécimal de 6 chiffres, les deux premiers indiquant la quantité de rouge, les deux suivants indiques la quantité de vert, les deux derniers celle de bleue.

*Exercice 3*
Faites la correspondance entre le nombre hexadécimal et la couleur.

#FF0000 ↔ _____ #00FF00 ↔ _____

#0000FF ↔ _____ #888888 ↔ _____

_____ ↔ noir _____ ↔ blanc

_____ ↔ gris foncé _____ ↔ gris clair

_____ ↔ jaune _____ ↔ magenta

_____ ↔ cyan _____ ↔ brun

# Codage de caractères, ASCII, iso-8859-1, iso-885-15, Windows-1252, utf-8

Chaque caractère est aussi codé par une séquence de bits. A l'origine, c'était facile, car les américains n'avaient besoin de coder que les 26 caractères alphabétiques minuscules, les 26 majuscules, les 10 chiffres et quelques caractères de ponctuation tels que : , . ; : ' ? ! + - = " ( ) * % & / [ ] { } etc. 96 (=128 – 32) symboles suffisaient amplement.
Donc pour ces caractères, un octet est largement suffisant.
Les 32 premiers nombres, codent des caractères de contrôle, tel que le tabulateur, le retour à la ligne, le saut de ligne, le saut de page, la fin de fichier.
Les nombres allant de 32 à 127 codent les caractères cités précédemment.
Le 8ème bit servait de bit de contrôle, pour tester si une erreur s'était introduite, par exemple lors d'une transmission. Ce bit de contrôle n'est plus utilisé de nos jours.
Ce codage sur 7 bits est le **code ASCII**. Il est **standard** et utilisé par tous les systèmes d'exploitation.

Avec l'apparition du DOS et l'utilisation d'ordinateurs en Europe, divers codages sont apparus pour coder les caractères accentués. Cela c'est fait de manière complètement anarchique, et de nombreux systèmes de codages ont étendus le code ASCII. Microsoft et Apple ont chacun développé leur système de codage.
En 1986, la norme **iso-8859-1** a définie le codage des caractères accentués, sur un octet.
Il définit le codage de 191 caractères.
Il manquait certains caractères, tel que le : œ.
La norme **iso-8859-15** a comblé en 1998 certains manques, en étendant le codage iso-8859-1.
Ce codage se nomme également **Windows-1252**.
L'avantage de ces systèmes de codage est qu'un caractère est codé sur un octet.

Pour les Américains, du nord et du sud et les Européens, cela suffit. Mais pas pour les arabes, les Japonais, le chinois et une majorité du monde ! Pour cela d'autres systèmes de codages ont été définis.
Au début des années 1990, le codage **UTF-8** est apparu. Il est utilisé actuellement dans plus de trois quarts des pages Web. Il a l'*avantage* de pouvoir coder plus d'un million de caractères, dont les caractères arables, japonais, chinois et bien d'autres. Il a le *désavantage* d'utiliser de 1 à 4 octets pour coder un caractère. Il est *compatible avec le codage ASCII*. Donc les caractères de l'ASCII sont codés sur un octet en utf-8, de la même manière qu'en ASCII.
Par contre, les caractères accentués sont codés sur deux octets.
C'est ce système de codage qui est utilisé sur UNIX, Linux et MAC OSX.
D'autres systèmes de codage existent, tels que l'utf-16 et l'utf-32. **Unicode** est une définition encore plus générale.

*Exercice 4*
Complétez le tableau suivant : (# signifie codage en hexadécimal) c.f. http://www.utf8-chartable.de/

| Caractère | code ASCII | code iso-8859-15 | code utf-8 |
|---|---|---|---|
| A | | | |
| | #61 = 97 | | |
| | | | #20 = 32 |
| é | | | |
| ñ | | | |
| | | | #c3a6 |
| | | | #e0a9a9 |

## Codage des nombres entiers

Les nombres peuvent se classer dans trois catégories. Les nombres entiers positifs, les nombres entiers relatifs et les nombres à virgules. L'informatique étant limitée, aucune distinction n'est faite entre les nombres irrationnels et les nombres rationnels. Parmi tous les nombres, très peu peuvent être codés de manière exacte.

*Exercice 5*
Si on se limite aux **entiers positifs** :
un **octet** peut coder les nombres de 0 à 255.
un **mot de 16 bits** peut coder les nombres de 0 à _____
un **mot de 32 bits** peut coder les nombres de 0 à _____
un **mot de 64 bits** peut coder les nombres de 0 à _____

Le codage des **entiers relatifs**, suit une logique mathématique.
Remarquons que : 255 + 1 = 256 = 1 0000 0000 en binaire.
Si on élimine le 9ème bit, cela donne 255 + 1 = 0, donc 255 représente le nombre entier négatif –1.
Toujours en éliminant le 9ème bit, 254 + 2 = 0, donc 254 représente le nombre entier négatif –2.

Sur un octet, on code les nombres entiers relatifs comme suit :
0 1 2 ... 127 sont les entiers positifs
255 254 ... 129 sont les entiers négatifs, de –1, –2, ... à –127.
128 est une autre manière de coder 0, il correspond à " –0 ".

Sur un **mot de 16 bits**, rappelons que $2^{16}$ = 65'536.
65'535 + 1 = 0, si on élimine le 17ème bit. Donc 65'535 = –1.
32'769 + 32'767 = 0, donc 32'769 = –32'767.

Sur un **mot de 32 bits**, rappelons que $2^{32}$ = 4'294'967'296. La moitié vaut : 2'147'483'648.

Le codage des nombres négatifs dépend du nombre d'octets choisi pour coder les nombres entiers.

*Exercice 6*
Sur un *mot de 32 bits*, à quel entier positif correspond le nombre –1 ?

En se limitant aux *octets*, que donne l'opération suivante, en nombre positif et en nombre négatif.
En nombre négatif, on parlera d'**overflow**, qui signifie que l'on dépasse les capacités de ce codage.
69 + 61

Sur un *mot de 32 bits*, dans le codage des entiers relatifs, quel est le plus grand nombre positif ?

Suggérez la taille d'un mot à considérer pour coder des entiers relatifs plus grand que mille milliards.
Dans ce système, quel sera le plus grand entier ? À quel nombre entier correspondra –1 ?

Dans un système de *mots de 16 bits*, donnez un exemple d'addition provocant un overflow.

Dans un système de *mots de 8 bits*, donnez un sens à l'égalité suivante.
2 + 250 = –4

# Codage des nombres à virgule

Le système de codage des nombres à virgules dans un ordinateur est assez complexe. On ne considérera que deux type de codage, en simple précision, sur 32 bits et en double précision, sur 64 bits. Ce dernier type de codage est le plus couramment utilisé de nos jours. Ces codages imitent l'écriture scientifique des nombres.

Codage **simple précision** = à **virgule flottante en 32 bites** : nombre = $s \cdot 2^{e-127} \cdot 1,$mantisse
s = 1 ou –1 suivant la valeur du premier bit.
e = le nombre entier positif représenté par les 8 bits allant du 2ème au 9ème.
La mantisse donne les 23 chiffres binaires après la virgule.

Codage **double précision** = à **virgule flottante en 64 bites** : nombre = $s \cdot 2^{e-1023} \cdot 1,$mantisse
s = 1 ou –1 suivant la valeur du premier bit.
e = le nombre entier positif représenté par les 11 bits allant du 2ème au 12ème.
La mantisse donne les 52 chiffres binaires après la virgule.

Il y a une exception si e = 0, alors le nombre = $s \cdot 2^{-127} \cdot 0,$mantisse en simple précision.
Il y a aussi une exception si e = maximum, le nombre représente + ou – l'infini.

*Exemples en simple précision :*

```
 1      = 0 01111111 00000000000000000000000
-2      = 1 10000000 00000000000000000000000
13      = 0 10000010 10100000000000000000000
```

Le signe est positif.§
le facteur multiplicateur vaut $2^{130-127} = 2^3 = 8$
1,mantisse vaut $1 + 1 \cdot 1/2 + 0 \cdot 1/4 + 1 \cdot 1/8 + 0 = 13/8$
Donc le nombre vaut bien $8 \cdot 13/8 = 13$.

```
 0,625  = 0 01111110 01000000000000000000000
```

le facteur multiplicateur vaut $2^{126-127} = 2^{-1} = 1/2 = 0,5$
1,mantisse vaut $1 + 0 \cdot 1/2 + 1 \cdot 1/4 + 0 \cdot 1/8 + 0 = 1 + 1/4 = 1,25$
Donc le nombre vaut bien $0,5 \cdot 1,25 = 0,625$

```
 0,1    ≈ 0 01111011 10011001100110011001100
```

le facteur multiplicateur vaut $2^{123-127} = 2^{-4} = 1/16 = 0,0625$
1,mantisse vaut $1 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-20} + 2^{-21}$
Le produit des deux nombres ci-dessus vaut presque 0,1.
Il n'est pas possible d'écrire 0,1 de manière exacte dans ce système de codage des nombres !

*Exemples en double précision :*

```
  1      = 0 01111111111 0000000000000000000000000000000000000000000000000000
-13      = 1 10000000010 1010000000000000000000000000000000000000000000000000
 -0,1    ≈ 1 01111111011 1001100110011001100110011001100110011001100110011001
  0,75   = 0 01111111110 1000000000000000000000000000000000000000000000000000
```

le facteur multiplicateur vaut $2^{126-127} = 2^{-1} = 1/2 = 0,5$
1,mantisse vaut $1 + 1/2 = 1,5$
Donc le nombre vaut bien $0,5 \cdot 1,5 = 0,75$

Pour qu'un nombre puisse s'écrire de manière exacte, il faut qu'il puisse s'écrire sous la forme d'un nombre entier divisé par 2 à la puissance un nombre entier.
Donc la plupart des nombres à virgules sont approximés en informatique.

**Utilisez des feuilles supplémentaires pour résoudre les exercices qui suivent.**

*Exercice 7 :*
En simple précision, quels sont les nombres représentés par :

```
A = 0 10000011 00010000000000000000000
B = 0 01111101 01000000000000000000000
C = 1 10000011 00000100000000000000000
D = 0 01111111 10011001100110011001100
E = 1 01111101 00110011001100110011001
F = 0 10000000 10010010000111111011010
G = 0 10000000 10010001111010111000010
H = 0 10000000 01011011111100001010100
I = 0 01111111 01101010000010011110011
J = 0 01111111 10011110001101110111100
```

*Exercice 8 :*
En double précision, quels sont les nombres représentés par :

```
A = 0 10000000000 10010010000111111011010100010001000010110100011000
B = 0 10000000000 01011011111100001010100010110001010001010111011010001
C = 0 01111111111 01101010000010011110011001100111111100111011111001101
D = 0 01111111111 10011110001101110111100110111001011111110100101010000
```

*Exercice 9 :*
En vous basant sur l'exercice 7, pouvez-vous écrire la représentation binaire en double précision, du nombre 1,6 ? Et du nombre –0,3 ?

*Exercice 10 :*
Représentez en binaire, simple précision, les nombres suivants :
( Si vous êtes particulièrement curieux, vous pouvez les représenter aussi en double précision )

```
 55
−28
Votre âge
 0,6
 -pi/4
 0,7
Le plus grand nombre possible
Le plus petit nombre positif
```

*Exercice 11 :*
Écrivez un programme en javascript, ou dans un autre langage, qui fait toutes les conversions de nombres...
C'est un projet en soi !

# Initiation au C++

Ce qui suit est une initiation au C++, sous le système d'exploitation GNU/Linux, XUbuntu 14.4, donc avec l'environnement Xfce.

Nous utiliserons l'IDE (Integrated Development Environment) Code::Blocks.
Nous nous baserons sur le cours de l'excellent site Web "http://fr.openclassrooms.com/"
Sous : Cours > Informatique C++ on arrive à http://fr.openclassrooms.com/informatique/c-1/cours
Prenez la recommandation de l'équipe, "Programmez avec le langage C++"
http://fr.openclassrooms.com/informatique/cours/programmez-avec-le-langage-c

Je ne résumerai qu'une petite partie de ce cours, que je vous conseil de suivre.

Sous "2. Les logiciels nécessaires pour programmer"
Vérifiez si le logiciel suivant existe sous "Menu démarrer" > "Développement" > "**Code::Blocks IDE**"
Si ce n'est pas le cas, vous devez l'installer, ainsi qu'un compilateur, comme suit :
1) ouvrez un Terminal, soit dans "Accessoires" > "Émulateur de Terminal"
soit en tapant "super T" où la touche "super" est celle de "Windows", entre "Ctrl" et "Alt".
2) dans ce Terminal, tapez : sudo apt-get install build-essential codeblocks
Le mot de passe standard vous est demandé, puis l'installation se fait.
"build-essential" est le compilateur C++, codeblocks est l'IDE.

Sous "3. Votre premier programme"
Lancez "Code::Blocks IDE" sous "Développement"
File > New > Project...
Choisir : "Console application" Next> "C++" Next>
Écrivez un titre, je suggère : "ex001_hello_world" tout en minuscule
"Folder to create project in:" je suggère : "/home/bg/cpp" ( *bg* étant votre identifiant )
Next>
Cochez la case "Create Release configuration", l'autre case, je la décoche.
Finish.
Cliquez sur la flèche, à gauche de "Sources".
Double cliquez sur main.cpp
Vous voyez votre premier programme ! Pour le compiler et l'exécuter, il y a 3 possibilités :
1) "Build" > "Build and run"
2) Pressez sur l'icône suit la flèche verte, celle qui a une flèche verte et un roue dentée jaune.
3) Pressez sur la touche F9.

Un fenêtre s'ouvre avec le texte : "Hello world!"
Pressez la touche Enter pour fermer cette fenêtre.

Essayez d'écrire un autre texte à la place de "Hello world", puis de relancer.
Vous pouvez aussi dupliquer cette ligne et changer le texte. Amusez-vous un peu...
Même un texte avec des accents fonctionne, ce qui n'était pas évident autrefois.
La ligne : "#include <iostream>" inclue la bibliothèque servant au "Input" et au "Output".

"int main()" est la fonction principale, celle qui s'exécute au début du lancement de votre programme.
Elle retourne un "int", c'est-à-dire un nombre entier.
"cout << "Hello world!" << endl;" dirige vers la sortie standard le texte, avec un retour à la ligne.
"return 0;" est la valeur retournée par cette fonction. "0" indique qu'il n'y a pas eu d'erreurs.

Sauvez le projet, fermez-le : "File" > "Close project"
Créez un nouveau projet : "File" > "New" > "Project" > "Console application"
Appelez-le "ex002_sommes"

Il est aussi possible de prendre le premier projet, de l'enregistrer comme un "Template"
"File" > "Save project as template..."
Puis "File" > "New" > "Project" > "User templates" puis prendre celui qu'on vient de sauver en "Template"

Ainsi je vous conseil d'avoir plusieurs projets pour sauvegarder divers **ex**emples.

Sous "4. Utiliser la mémoire"
Le C++ est un langage typé. Chaque variable a un type bien défini. Regardez les type défini dans cette section.
Lisez la suite si vous voulez, je la saute.

Sous "5. Une vraie calculatrice" regardez ce que vous voulez.
Pour la suite, je vous laisse voir.

Voici un petit programme pour apprendre, avec des commentaires.
C'est celui que je mets dans "ex002_sommes"

```
#include <iostream>  // pour les entrées - sorties (In & Out)
#include <cmath>     // pour les fonctions mathématiques,  sqrt = racine carrée.
using namespace std;

int main()
{
int nCount = 0;  // indice qui va parcourir les entiers de 1 à nMax
int nMax = 0;    // valeur maximum de l'indice de la somme
double vSum = 0; // résultat de la somme

// Demande à l'utilisateur la valeur de nMax
cout << "nombre de termes à sommer : ";
cin >> nMax;  // attend la saisie d'un nombre entier positif.

// Boucle effectuant la somme
for (nCount = 1; nCount <= nMax; nCount++) {
   // tenez compte de l'indentation !
   vSum = vSum + 1.0 / (1.0*nCount * nCount);
   // sans le "1.0*nCount", le programme est très limité, car
   // on dépasse rapidement la capacité de codage des nombres entiers.
   }

// Ce premier résultat ne permet pas de deviner ce que cette somme vaut.
cout << "La somme des inverses des carrés de 1 à " << nMax
     << "vaut : " << vSum << endl;

// Autre affichage, qui donne plus d'espoir de deviner la valeur de la somme
// lorsque  nMax  tend vers l'infini.
cout << "Racine carré de 6 fois la somme ci-dessus : " << sqrt(6*vSum) << endl;

return 0;
}
```

`On peut ajouter : cout << setprecision(15);  // pour définit une précision.`
À ajouter dans l'en-tête : #include <iomanip> // pour le formattage, tel que setprecision, setw, setfill('-'), setbase

Sous "6. Les structures de contrôle"
Apprenez à utiliser les structures de contrôles :

4 structures de contrôle importantes :
if (...) { ... } else if (...) { ... } else { ... }
for ( i=1; i<10; i++) { ... }
while (...) { ... }
do { ... } while (...);

*Exercice 1*
Écrivez un programme, qui calcule la plus grande valeur de nMax, telle que la somme des carrés de nombres de 1 à nMax soit le plus grand nombre inférieur à 1'000'000 !

*Exercice 2*
Écrivez le programme suivant :
( Il tient sur 45 lignes, avec espaces et des commentaires ! )
L'ordinateur tire un nombre au hasard entre 1 et 100.
Utilisez :

```
#include <time.h>
srand (time(NULL));  // pour initialiser le générateur de nombres aléatoires
nNbrCache = rand() % 100;  //  pour tirer un nombre entier au hasard entre 0 et 99.
```

Le joueur écrit un nombre.
S'il est correct, il a gagné. On indique le nombres d'essais.
S'il est trop petit, ou trop grand, l'ordinateur l'indique et le joueur essaye à nouveau.
Cela initie à la méthode *dichotomique*.

*Cette page est laissée blanche, pour prendre des notes manuscrites...*

# Initiation au PHP.

## Introduction

Deux notions sont essentiels pour comprendre la suite :
1) **Le client** est la personne qui surf sur le Web avec un navigateur et qui reçoit la page Web. Souvent on parle du client pour décrire l'ordinateur de la personne qui surf.
2) **Le server** est l'ordinateur qui contient les données transmises par Internet au client. En général, il n'y a pas de personne du côté du serveur, juste un ordinateur puissant qui envoie les données au client qui surf sur le Web.

Le *javascript s'exécute sur l'ordinateur du client*. Ce langage ne permet donc pas de sauver des données sur le serveur, dans une base de données par exemple. Il permet de sauver des informations sur l'ordinateur du client, mais le client peut les effacer.

Le *PHP s'exécute sur le serveur*. Ce langage permet donc de sauver des données sur le serveur et ne sont pas gérable par le client, qui utilise un service d'un site Web.

*L'interpréteur de javascript* est contenu dans les divers navigateur, donc aucune adjonction n'a besoin d'être faite pour l'utiliser sur son ordinateur.

Si un programme utilisant du PHP a été écrit et est chargé dans un navigateur, il ne se passe rien et la partie "PHP" ne s'exécute pas. C'est normal, car le PHP ne s'exécute que sur un serveur !
Pour développer du PHP, il existe deux solutions :
1) Écrire son programme PHP, le transférer (par FTP) sur le serveur, puis le charger et tester dans un navigateur en accédant à la page Web correspondante sur le serveur.
2) Installer un serveur sur son propre ordinateur, qui jouera en même temps le rôle du serveur et de l'ordinateur du client.
   Cette deuxième option est plus compliquée au départ, mais simplifie la vie du développeur sur le long terme.

## Un premier exemple

Testons la première solution avec un petit exemple PHP.
Recopiez les lignes de la page suivantes dans un éditeur de texte.
Nommez le fichier : **php01_compteur.php**
L'extension .php est essentielle !
( Il est aussi téléchargeable sous : http://www.juggling.ch/zgisin/a2015_oc4/index.html )

Vous devez également créer un fichier nommé : **compteur001.txt** qui contient uniquement le nombre 0 dans sa première ligne et rien d'autre.

Depuis un gestionnaire de fichier, vous devez modifier les permissions d'accès ( = Attributes ).
Tout le monde doit pouvoir accéder à ce fichier en Lecture & Écriture.

```
<head>
<meta charset="utf-8">
<title>
Test de compteur en PHP
</title>
</head>
<body>
Test de compteur en PHP<br>
<br>
<?php
$couleurtexte="#F00000";

   // nom du fichier contenant le numéro du compteur
   $fichier = "./compteur001.txt";

   // Ouvre le fichier  $fichier  en lecture
   $fp = @fopen($fichier, "r");

   // Test si le fichier  $fichier  existe et peut être lu
   if (!$fp) {
      echo "Impossible d'ouvrir $fichier en lecture";
      exit();
      }

   // Lit le contenu du fichier, qui est le numéro du compteur
   $visites = fgets($fp, 8);
   $visites = $visites + 1;  // incrément du numéro du compteur

   // affiche dans la couleur désirée le numéro du compteur
   echo "<span style='color:$couleurtexte;'>";
   echo $visites;
   echo "</span>"; // on affiche $visites, et on increment $visites.
   fclose($fp);  // Ferme le fichier

   // Ouvre le fichier  $fichier  en écriture
   $fp = @fopen($fichier, "w"); // le fichier est ouvert en ecriture, remis a zero

   // Test si le fichier  $fichier  existe et peut être modifié
   if (!$fp) {
      echo "<br>";
      echo "Impossible d'ouvrir $fichier en ecriture";
      exit();
      }

   // Le contenu du fichier est remplacé par la nouvelle valeur du compteur
   fputs($fp, $visites);
   fclose($fp); // Ferme le fichier.
?>
</body>
</html>
```

Transférez les deux fichiers : **php01_compteur.php** et **compteur001.txt** sur votre site Web.

Au premier essai, il est probable que vous ayez des problèmes d'accès en écriture au fichier compteur001.txt. Il faudra modifier ses droits d'accès depuis FileZilla :
Clique droit sur le fichier ; droits d'accès au fichier, cocher toutes les cases "Lire" et "Écrire".

Rechargez la page Web : "php01_compteur.php" dans votre navigateur pour voir le compteur changer.
Si vous allez dans la page Web d'un autre élève, son compteur augmentera aussi.

Nous étudierons plus tard la signification du code.
Lisez le code source de votre page Web, depuis le navigateur, il ne contient plus d'instruction PHP !

## Installation d'un serveur sur votre ordinateur

C.f. le cours de : https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql
Un autre cours : http://www.toutjavascript.com/savoir/savoir23.php3
Un manuel de référence PHP en français : http://php.net/manual/fr/
LA référence du w3schools.com du PHP, en anglais : http://www.w3schools.com/php/default.asp

On se rend compte rapidement qu'il est désagréable de devoir transférer sur le serveur le fichier .php pour le tester. De plus, lorsque plusieurs personnes accèdent au même domaine, des conflits et blocages interviennent !
C'est la raison pour laquelle, il est agréable d'installer un serveur sur son propre ordinateur.
On peut le faire manuellement, mais la méthode automatique suivante simplifie les étapes.

**Suivez les instructions suivantes pour installer un serveur sur votre ordinateur :**

1) Allez sur le site : https://www.apachefriends.org/index.html
2) Cliquez sur "Download Click here for other versions"
3) Allez sous XAMPP for Linux ...
   Télécharger la version PHP 5.5.28 "Download (32 bit)".
   Il est conseillé de télécharger dans un dossier simple, tel que : "Bureau".
   La version peut changer, mais sur votre système 32 bits, il faut charger celle correspondante.
   Le téléchargement prendra quelques minutes...
4) Ouvrez un Terminal.
   Dans ce Terminal, tapez "cd *nom du dossier de téléchargement*", suivit de la touche Enter.
   Par exemple : "**`cd Bureau`**", suivit de la touche Enter.
5) Tapez **`ls`** pour vérifier le contenu du dossier. Cela n'est pas indispensable.
6) Deux choix possibles :

6a) Dans le Terminal, tapez : **`chmod 755 xampp-linux-*-installer.run`**

6b) Depuis un gestionnaire de fichiers, indiquez que le fichier que vous venez de télécharger soit exécutable par tout le monde, en modifiant ses propriétés.

7) Tapez **sudo ./xampp** suivit de la touche TAB, ce qui complétera le nom.
8) Pressez sur la touche **Enter**.
   Le mot de passe standard vous est demandé, puisque vous installez un logiciel en mode "**root**".
   **1234** est ce mot de passe sur votre clé USB.
9) L'installation des logiciels nécessaires pour que votre ordinateur fasse office de serveur démarre.
   Il faut attendre un peu avant qu'une fenêtre de "setup" s'ouvre. Suivez les instructions.
   Laissez les options par défaut. Acceptez les conditions, continuez...
   L'installation prend plusieurs minutes.
   À la fin, cliquez sur Terminez. Fermez la fenêtre qui s'ouvre.
10) Lancez un navigateur et tapez l'URL : **`http://localhost`**
    Une page Web devrait s'ouvrir, indiquant que l'installation c'est bien passé.
11) Pour ne pas changer notre habitude d'utiliser le répertoire "website" pour le Web, depuis un Terminal, tapez : **`sudo ln -s $HOME/website /opt/lampp/htdocs/web`**
    De cette manière, en tapant l'URL : **http://localhost/web**, vous arrivez dans votre website.
12) Copiez les fichiers : **php01_compteur.php** et **compteur001.txt** dans le dossier **website**.
13) Dans un navigateur, tapez dans la barre d'adresse :
    **`http://localhost/web/php01_compteur.php`**
14) Mettez dans votre fichier **index.html** des **liens** sur les fichiers .php que vous créez.

On peut maintenant copier des fichiers dans le dossier **tests** et utiliser le serveur de l'ordinateur.
Pour lancer le serveur, tapez depuis un Terminal : **`sudo /opt/lampp/lampp start`**
Pour arrêter le serveur, tapez depuis un Terminal : **`sudo /opt/lampp/lampp stop`**

Si tout c'est bien passé, vous pouvez tester votre premier programme PHP sur votre ordinateur, sans nécessiter une connexion Internet !

Anciennement, entre parenthèses, si on n'a pas fait les points 11 à 14 ci-dessus.

11) Dans le Terminal, tapez : **`cd /opt/lampp/htdocs`**
12) Tapez : **`sudo mkdir tests`** ce qui créera un nouveau dossier, dans lequel nous sauverons les pages Web PHP à tester.
13) Tapez : **`sudo chown bg tests`** ce qui changera le "owner" (possèsseur) du dossier tests.
14) Tapez : **`sudo chgrp bg tests`** ce qui changera le groupe du dossier tests.

## Étude du premier exemple de programme PHP

Un programme PHP est une simple page HTML, dans laquelle une nouvelle balise est utilisable :
**<?php .... ?>**
Tout ce qui se trouve à la place des **...** est du code PHP qui générera du code HTML.
La page reçue dans le navigateur ne voit que le résultat final et non le code PHP.

Comme dans tous les langages de programmation, il y a : ( Voir plus loin, le mini résumé PHP )

1) Les **variables**, qui commencent par un $
2) Les opérations sur les variables. + - * /
3) Les tests **if (condition) { liste d'instructions }**
4) Les boucles **for**
5) Les boucles **while**
6) Les **commentaires**

etc.

En plus :

a) Il est simple d'accéder à des fichiers, de les lire et de les modifier.
b) L'instruction **echo** écrit le code HTML dans la page Web.

Les indications ci-dessus devraient vous permettre de comprendre votre premier programme PHP.

## Un deuxième exemple de programme PHP, plus simple

Dans **website** nommez : **php02_simple.php** le fichier contenant les instructions suivantes :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>
Test simple en PHP
</title>
</head>
<body>
Test simple en PHP<br>
<br>
<?php
   $nbr = 87654;
   // affiche dans la couleur désirée le nombre $nbr
   echo "<span style='color:#008000'>";
   echo "Le \"nombre\" est : $nbr";
   echo "</span>";
?>
<br>
<hr style="margin-left:0; width:95%">
<p style="margin-top:0; font-size:80%">
Plan du Site : <a href="index.html">Home</a>  
<img src="images/arrow_right.gif" alt="arrow" width="13" height="9">   php02_simple.php
</p>
<hr style="margin-left:0; margin-bottom:0; width:95%">
<p style="margin-top:0; font-size:80%">
Page mise à jour le 20 septembre 2015 par Bernard Gisin<br>
Hébergement par : <a href="http://www.infomaniak.ch">www.infomaniak.ch</a>
</p>
<br><br><br><br>
</body>
</html>
```

Dans un navigateur, tapez dans la barre d'adresse :
**`http://localhost/web/php02_simple.php`**

N'oubliez pas d'avoir activez le serveur avec : **`sudo /opt/lampp/lampp start`**
si vous l'aviez arrêté entre temps.

Regardez le code source.
Modifiez ce programme, jouez avec...
Remarquez que \" permet d'introduire des guillemets dans un texte.

**Les commentaires** en PHP sont importants, comme dans tout langage, pour comprendre sur le long terme ce que le programme fait. Comme en javascript, un commentaire commence par **`//`**
On peut mettre plusieurs lignes en commentaire si elles se trouvent entre **`/*`** et **`*/`**

Remarquez que **chaque instruction doit se terminer par un point-virgule `;`**

## Un troisième exemple de programme PHP : les menus

Il arrive que le même code HTML se retrouve dans plusieurs pages HTML. Lors d'une modification de ce code, il faut répéter cette modification pour chaque page, cela peut être long.
L'exemple typique est le menu que l'on retrouverait dans chaque page.
En PHP, il est possible d'**inclure un fichier .php** dans votre code PHP.

Référez-vous au site : http://www.juggling.ch/zgisin/a2015_oc4/index.html
pour ce troisième exemple. L'extension **.txt** devra être remplacé par **.php**
**php03_menu_include.php** , contiendra le menu
**php03_menu_1.php** , est un exemple qui inclus le menu
**php03_menu_2.php** , est un autre exemple qui inclus le menu
Vous devez écrire vous-même les exemples : php03_menu_2.php et php03_menu_3.php à partir de php03_menu_1.php

La seule nouveauté du point de vue PHP est l'instruction :
**`<?php include("php03_menu_include.php"); ?>`**

Du point de vue CSS, le menu est assez sophistiqué.

Il est instructif de regarder le code source de la page **php03_menu_1.php** depuis un navigateur.

Un bout de code javascript a été introduit dans **php03_menu_include.php**, pour griser et désactiver le lien sur la page à elle-même. Par exemple, la page "**php03_menu_1.php**" ne doit pas avoir de lien qui redirige sur la page "**php03_menu_1.php**".

## Les variables en PHP

Le chapitre 5 du livre disponible sur le Web à l'adresse ci-dessous explique bien la notion de variable.
https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-variables-44

Une variable en PHP commence toujours par le symbole $
**Elle est composée de lettre, de chiffres et du caractère _**
D'aucun autre caractère, pas d'espace, pas d'accent, pas de symbole.

Les types des bases des variables sont :

1) **String** ( = chaîne de caractères )
   $mon_nom = "Gisin";
   $prenom = 'Bernard'; // on peut utiliser " et ' pour délimiter un string.
   $special = "un guillemet \" dans un string"; // remarquez comment insérer un guillemet.
2) **int** ( = nombre entier signé )
   $mon_age = 18;
3) **float** ( = nombre à virgule )
   $pi = 3.14; // approximativement :-)
4) **bool** ( = booléen, soit **true** soit **false** )
   C'est généralement le résultat d'un test
   $cours_interessant = true;
   $j_en_ai_mare = false;
   $test = $pi == 3.141592653; // le test retournera false si on prend la définition précédente de $pi
5) **NULL** ( = la variable qui ne contient rien )
   $pas_de_valeur = NULL; // la variable ne contient rien, mais cela pourra changer

Pour afficher le contenu d'une variable et qu'elle fasse partie du code HTML généré par l'interprétation du PHP, l'instruction **`echo`** est essentielle.

```
<?php
echo "mon nom est : $mon_nom<br>";
echo "mon âge est : $mon_age<br>";  // affichera : mon âge est : 18
// remarquez que la variable est aussi dans les guillemets !
// Les variables se trouvant dans des guillemets double sont remplacées par leur contenu.
echo 'âge : $mon_age<br>';  // affichera : âge : $mon_age<br>
// Les variables se trouvant dans des guillemets simple ne sont pas interprétées.
?>
```

Mettre un point **.** entre deux strings permet de la **concaténer**.

```
echo 'âge :' . $mon_age . '<br>';  // donne : âge : 18<br>
```

La page "**php04_variables.php**" donne des exemples.

Indiquons encore les **opérations** possibles : `+ – * /`
Pour le reste de la division d'un nombre entier par un autre, l'opération s'appelle "modulo" et se note `%`

```
echo 18 % 7; // donnera  4, car  4 = 18 - 2*7
```

## Autres chapitres sur le PHP

Les chapitres suivants sont utiles et importants. Nous les verrons lorsque cela nous sera utile.

6) **Les conditions, if ... else, les conditions multiples, switch :**
   https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-conditions-41
7) **Les boucles, while, for :**
   https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-boucles-34
8) **Les fonctions :**
   https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-fonctions-32
9) **Les tableaux :**
   https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-tableaux-43

# Transmettre des données à une page Web

## 10) Transmettre des données avec l'URL

https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/transmettre-des-donnees-avec-l-url

Les pages écrites en PHP ne seraient pas très utiles, si elles ne pouvaient pas recevoir des données d'un utilisateur et les transmettre à une page Web.

Une première manière de transmettre des données à une page écrite en PHP est à travers son URL.

La page "**php08_donnees_dans_url.php**" donne un exemple.
Regardez le code de cette page.

La page "**php09_donnees_dans_url.php**" donne un exemple plus complexe, avec sauvegarde de données dans un fichier, sur le serveur !
*Quelques explications sont nécessaires :*

```
if (isset($_GET['nom'])) $mon_nom = $_GET['nom'];
```
Test qu'un paramètre nommé `nom` existe.
Si c'est le cas, assigne à la variable `$mon_nom` la valeur du paramètre `nom`.

```
$fichier = "./noms_liste.txt";      $fp = @fopen($fichier, "r");
```
Ouvre en lecture le fichier : nom_liste.txt, qui se trouve sur le serveur !

```
$aLignes = array();
```
Définit un tableau (une liste), qui contiendra les lignes du fichier.

```
$nCompte = 0;
while ($ligne = fgets($fp, 40)) {  // Lit toutes les lignes du fichier
      $nCompte = $nCompte + 1;
      $aLignes[$nCompte] = $ligne;
      }
$ligne = fgets($fp, 40)
```
Cette instruction, à l'intérieur du `if`, lit une ligne du fichier, la sauvegarde dans la variable `$ligne`.
De plus, si c'est la dernière ligne du fichier, retourne `false`, donc la boucle `while` se termine.
Si ce n'est pas la dernière ligne, l'instruction retourne `true`, donc la boucle `while` continue.

```
$aLignes[0] = $mon_nom . "\n"; //
```
Ajoute le dernier nom tapé, avec un retour à la ligne
Donc ici, le tableau $aLignes contient toutes les lignes du fichier, plus la donnée qui vient d'être tapée.

```
if ((strlen($mon_nom) >= 1) && ($mon_nom != "---")) {
```
Si une nouvelle donnée vient d'être tapée, pas juste un champ vide, alors sauve toutes les lignes dans le fichier.

On voit que la structure d'une **boucle for** est très similaire à celle en javascript.

```
echo "<b>$aLignes[$nn]</b><br>\n";
```
Écrit des lignes de code HTML dans le fichier qui sera reçu par le navigateur.

Rappelons que toutes l'interprétation du code PHP se fait du côté du serveur !

## Autres exemples de code PHP et des exercices.

Il est aussi possible d'inclure une page Web dans une autre et ainsi d'inclure un bout de page Web PHP dans une page Web HTML. Cela est utile par exemple pour inclure un compteur dans une page Web HTML standard. c.f. mon exemple php10... et le suivant.

-> Introduction au MySql ... Peut-être une autre année :-)

## Documentation.

Une excellente **documentation** en français est disponible sous :
http://php.net/download-docs.php
Je vous conseille de télécharger la documentation en français au format .chm (le premier ou le deuxième choix).
Avec la "Logitéque Ubuntu", installer "xCHM" (ou KchmViewer).
Ceci vous permettra de visualiser la documentation très complète, avec exemples sur le PHP.

**Exercices.**

1) Écrivez une page PHP qui écrit la liste de tous les nombres de 1 à 10 et leur carré.
1b) Idem avec un joli affichage utilisant un tableau.

2) Écrivez une page PHP qui écrit une table de multiplication de 2 x 2 jusqu'à 10 x 10
Il est recommandé d'utiliser un tableau pour un plus joli affichage.

3) Écrivez une page PHP qui crée un tableau de boutons, sur 4 lignes et 5 colonnes, chaque bouton ayant un 'id' différent égale à son numéro de colonne + 10 fois son numéro de ligne.

4) Écrivez une page PHP un mini chat en ligne. Cela se fait en quelques dizaines de lignes !
On peut le complexifier ensuite pour arriver à des milliers de lignes de code ou plus, comme dans Facebook.
À partir d'une certaine complexité, une base de données est nécessaire.

5) Écrivez une page PHP qui code de manière secrète les données envoyées au serveur dans l'URL et qui permet de n'accéder à une page Web que si on donne le bon mot de passe.

# Mini résumé d'instructions PHP

## Introduction

Les **instructions php** commencent par **<?php** et terminent par **?>**

Chaque instruction doit se terminer par un **point-virgule** **;**

Manuel de référence PHP : http://php.net/download-docs.php

## Les commentaires

Comme dans tout langage, ils sont importants en PHP pour comprendre sur le long terme ce que le programme fait. Comme en javascript, un commentaire commence par **//**

On peut mettre plusieurs lignes en commentaire si elles se trouvent entre **/*** et ***/**

## Les variables

Une variable en PHP commence toujours par le caractère **$**

Elle est composée de lettres **minuscules** et/ou **Majuscule**, de **chiffres** et du caractère **_**

Elles sont sensibles à la casse (minuscule - Majuscule)

°) **String** ( = chaîne de caractères )
$mon_nom = "Gisin";
$prenom = 'Bernard'; // on peut utiliser " et ' pour délimiter un string.
$special = "un guillemet \" dans un string"; // remarquez que \" permet d'introduire des guillemets dans un texte.

°) **int** ( = nombre entier signé )
$mon_age = 18;

°) **float** ( = nombre à virgule )
$pi = 3.14; // approximativement :-)

°) **bool** ( = booléen, soit **true** soit **false** )
C'est généralement le résultat d'un test
$cours_interessant = true;
$j_en_ai_mare = false;
$test = $pi == 3.141592653; // le test retournera false si on prend la définition précédente de $pi

°) **NULL** ( = la variable qui ne contient rien )
$pas_de_valeur = NULL; // la variable ne contient rien, mais cela pourra changer

## Les opérations sur les variables.

Opérateurs usuels sur les nombres : + - * / % (pour le modulo) ** (pour la mise à la puissance, dès la version 5.6)

```
<?php $reste = 24 % 7;  echo $reste; // donne le reste de la division de 24 par 7 ?>
```

Opérateur pour concaténer deux strings : + ou .

```
<?php  $mon_nom = "Gisin";  $n_p = $mon_nom."Bernard"; echo $n_p; ?>
```

Opérateur logique, généralement utilisé pour comparer des variables :

```
==   !=   <   >   <=   >=   || or (ou)   && and (et)
```

Il y en a encore plusieurs autres, c.f. "Les opérateurs" dans le manuel de référence PHP.

## L'instruction **echo**

Pour afficher le contenu d'une variable et du code HTML, pour qu'ils fassent partie du code HTML généré par l'interprétation du PHP, l'instruction **echo** est essentielle.

```
<?php
$mon_nom = "Gisin";
$mon_age = 18;
echo "mon nom est : $mon_nom<br>";
echo "mon âge est : $mon_age<br>";  // affichera : mon âge est : 18
// remarquez que la variable est aussi dans les guillemets !
// Les variables se trouvant dans des guillemets double sont remplacées par leur contenu.
echo 'âge : $mon_age<br>';  // affichera : âge : $mon_age<br>
// Les variables se trouvant dans des guillemets simple ne sont pas interprétées.
echo "<span style='color:rgb(255,0,0);'>âge : $mon_age</span><br>";
// affichera : âge : 18 de couleur rouge.
echo "<javascript> alert('non prévu'); </javascript>";
// générera le code javascript qui déclenchera une alerte.
?>
```

Les tableaux **array**
Définition d'un tableau (d'une liste), de variables
Très utilisé avec les boucles for et foreach

```
<?php
$aPrenoms = array();
$aPrenoms[0] = "Tristant";
$aPrenoms[1] = "Sebastien";
$aPrenoms[2] = "Lucas";
//...
$aPrenoms[13] = "Adrien";
?>
```

## Les structures de contrôle

### Le test **if (condition) { liste d'instructions }**

```
<?php
if ($a > $b) {
  echo "a=$a est plus grand que b=$b";
  }
elseif ($a < $b) {
  echo "a=$a est plus petit que b=$b";
  }
else {
  echo '$a est égal à $b';
  }
?>
```

Syntaxe alternative :

```
<?php if ($a == 5): ?>
A égal 5
<?php endif; ?>
```

### Les boucles **for**

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i."  ".$aPrenoms[$i]."<br>";
}
?>
```

### La boucle **foreach**

foreach (array_expression as $value) { //commandes }

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    echo $value."<br>";
}
?>
```

Passe en revue le tableau $arr et affiche les valeurs du tableau.

// Autre exemple, permettant de modifier les valeurs du tableau parcouru

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {  // Remarquez le  &  devant  $value
    $value = $value * 2;
}
// $arr vaut maintenant array(2, 4, 6, 8)
?>
```

Cette manière de faire, avec le &$value permet de modifier les valeurs du tableau
Plus d'explication est donné dans la partie sur les fonctions.

Les boucles **while (condition) { ... }**

Tant qu'une condition est satisfaite, continue d'exécuter une suite d'instructions.

```
<?php
$nCompte = 0;
while ($ligne = fgets($fp, 40)) {  // Lit toutes les lignes du fichier
      $nCompte = $nCompte + 1;
      $aLignes[$nCompte] = $ligne;
      }
$ligne = fgets($fp, 40)
?>
```

Les boucles **do { ... } while**

Les boucles do { ... } while (codition) ressemblent beaucoup aux boucles while, mais assure que le contenu de la boucle soit exécuté au moins une fois.

```
<?php
$sum = 0;
$k = 0;
do
  $k = $k + 1;
  $sum = $sum + $k;
  }
while ($sum <= 1000);
?>
```

Additionne les entiers successifs, jusqu'à ce que la somme dépasse 1000.

Quelques instructions qui permettent de modifier le déroulement normal d'une boucle :

```
break;   continue;   return;   goto;
<?php
for($i=0,$j=50; $i<100; $i++) {
  while($j--) {
    if($j==17) goto end;
  }
}
echo "i = $i";
end:
echo 'j hit 17';
?>
```

L'inclusion d'un fichier PHP dans un autre : **include** '*nom de fichier*';

Premier fichier : **vars.php**

```
<?php
$couleur = 'verte';
$fruit = 'pomme';
?>
```

Deuxième fichier : **test.php**

```
<?php
echo "Une $fruit $couleur"; // Une
include 'vars.php';
echo "Une $fruit $couleur"; // Une pomme verte
?>
```

## Les fonctions

```
<?php
function Max($n1, $n2) {
//======================
// Retourne le maximum entre $n1 et $n2
// Manière bizarre de faire, pour illustrer le passage de paramètre par valeur
if ($n1 < $n2) {
   $n1 = $n2;
   }
return $n1;
} // Max
?>
```

Dans cet exemple, les paramètres sont passés **par valeur**, cela signifie que c'est la valeur des paramètres qui sont transmis à la fonction.
Un appel du genre : $a1= 3; $a2=7; Max($a1, $a2); retournera 7 et ne modifiera pas les valeurs de $a1 ni de $a2. Le fait de modifier $n1 dans la fonction Max laisse inchangée les valeurs de $a1 et $a2.

```
<?php
function Swap(&$n1, &$n2) {
//=========================
// Échange les valeurs des variables transmises dans $n1 et dans $n2.
// Remarquez le  &  devant les variables, qui sont ainsi transmises par référence.
{
$temp = $n1;
$n1 = $n2;
$n2 = $temp;
} // Swap
?>
```

Dans cet exemple, les paramètres sont passés **par référence**, cela signifie que c'est la référence aux variables passées dans les paramètres qui sont transmis à la fonction.
Un appel du genre : $a1= 3; $a2=7; Swap($a1, $a2); aura pour conséquence que $a1==7 et que $a2==2. Les contenus des deux variables ont été modifiées. Cela est possible car c'est l'adresse des variables qui ont été transmises, et non la valeur des variables.

## Quelques **fonctions utiles**

**echo** déjà vu précédemment.
**fprintf** permet de définir plus précisément la sortie, ce qui peut être utile pour des nombres à virgule.
**exit();** // quite l'interpréteur PHP, généralement suite à une erreur ou anomalie.
**$Nom = $_GET**['nom']; // récupère la valeur du paramètre 'nom' de l'URL.
**isset**(**$Nom**); // Test si la variable $Nom contient une valeur (nombre, string, boolean, ...)

```
$fichier = "./noms_liste.txt";
$fp = @fopen($fichier, "r"); // Ouvre en lecture le fichier : nom_liste.txt, qui se trouve sur le serveur !
$fp = @fopen($fichier, "w"); // Ouvre en écriture le fichier : nom_liste.txt, qui se trouve sur le serveur !
$ligne = fgets($fp, 40); // Lit une ligne du fichier, la sauvegarde dans la variable $ligne.
fputs($fp, $ligne); // Écrit le contenu de $ligne sur une ligne à la fin du fichier.
fwrite($fp, $ligne); // comme fputs
fclose($fp); // Ferme le fichier précédemment ouvert
fseek($fp, $position_en_octets); // définit la position dans le fichier.
$long = strlen($mon_nom); // retourne la longueur d'une chaîne de caractère (string)
```