Les constantes

Reprenez le programme permettant de faire clignoter la LED 13.
Vous remarquerez que certains mots étaient colorés en bleu : HIGH, LOW OUTPUT… Ce sont des constantes.
Il existe donc des constantes déjà définies mais nous pouvons également créer nos propres constantes.
Voici le programme qui transforme le nombre 13 en une constante que j'ai appelée : CONNEXION.

```
1
2  // on définit la constante en donnant un nom et un nombre.
3  const int CONNEXION=13;
4
5  void setup()
6  {
7      // on l'utilise en écrivant juste son nom
8      pinMode(CONNEXION,OUTPUT);
9  }
10
11 void loop()
12 {
13     // là encore.
14     digitalWrite(CONNEXION,HIGH);
15     delay(1000);
16     // et là encore.
17     digitalWrite(CONNEXION,LOW);
18     delay(1000);
19 }
```

- On écrit le nom des constantes en majuscules (ça fait partie des conventions de programmation) ça permet de savoir que ce sont justement des constantes.
- Le nom de notre constante (CONNEXION) ne s'écrit pas en bleu car elle n'est pas une constante réservée à l'IDE mais bien une à nous
- La déclaration d'une constante (c'est-à-dire sa construction) commence par le mot clé **const** suivi du type de constante (dans notre cas la constante CONNEXION est de type "**int**", qui signifie nombre entier
  - La déclaration d'une constante est toujours de la forme :
    const type NOMDELACONSTANTE=valeur;

Remarquez que le signe égal en informatique n'a pas le même sens qu'en mathématiques, dans notre cas il a un rôle d'affectation, ce qui signifie qu'il dit à l'ordinateur de stocker le nombre valeur dans une case mémoire et de l'associer au mot « NOMDELACONSTANTE »
Par conséquent, vous ne pourrez pas modifier une constante en cours de programme.

Les variables

Les variables, comme leur nom l'indique, **peuvent varier pendant l'exécution**. On peut expliquer une variable comme une sorte de boîte dans laquelle on place des éléments (entier, caractères, décimaux…)
La déclaration de variables se fait en deux temps :

- *Premier temps* : je réserve une place en mémoire pour une variable et je lui donne un nom (cette opération est faite une seule fois dans le programme).
- *Autres temps :* j'affecte une valeur à la variable (cette opération peut être faite à tout moment dans le programme).

Exemple :
int nombreEleveICN ;
nombreEleveICN =34 ;
nombreEleveICN =35 ;

Au début on a tendance à vouloir mettre des noms de variables courts (genre une lettre) par ce qu'on a la flemme d'écrire. Mais très vite on s'aperçoit que des noms de variables clairs (presque des phrases) facilitent la lecture du programme et sa compréhension. Habituez-vous très vite à cette façon de faire, vous gagnerez par la suite beaucoup de temps. D'autant que la taille du nom d'une variable n'a pas d'influence sur le programme.
Pas d'accent, pas d'espace dans les noms de variable. Évitez en général les caractères spéciaux.
Interdit aussi de commencer un nom de variable par un chiffre.

Le **typage** est obligatoire lorsqu'on définit une variable ou une constante, ce n'est pas le cas dans tous les langages de programmation (ex python).

Type	Taille en octet (8 bits)	Valeurs stockées
boolean	1	true ou false
byte	1	un entier entre 0 et 255
int	2	un entier entre -32768 et 32767
unsigned int	2	un entier entre 0 et 65535
float	4	un décimal, précis à 7 chiffres après la virgule

Exercice 1 *: Dans le programme précédent, créer une variable qui contiendra le temps de pause entre deux clignotements (valeur contenue dans delay), puis modifier le programme de façon à utiliser cette variable.*

Calculs sur les variables

Une variable peut être modifiée par calcul, ainsi on peut écrire :

```
int uneVariable;  //uneVariable contient on ne sait trop quoi
uneVariable=10;  //uneVariable contient 10
uneVariable=10+22;  //uneVariable contient 32 (résultat de 10+22)
```

Mais également, et c'est là que cela devient intéressant :

```
int uneVariable;
uneVariable=10;
uneVariable=uneVariable+22;
```

 ... ben oui, car n'oubliez pas qu'une variable est une sorte de boîte, ainsi si dans uneVariable j'ai mis 10, je peux ajouter dans la boite uneVariable : 22, c'est donc toujours la même boîte mais avec 22 éléments en plus….

```
//Déclarations
int a;
int b;
int c;

//Initialisations
a=3; // a vaut 3
b=5; // b vaut 5
c=0; // c vaut 0

//Calculs
c=b+a; // c vaut ……..
a=a+c; // a vaut ……..
b=a+b; // b vaut ……..
c=a+b; // c vaut ……..
```

Si vous avez compris….

Exercice 2 *: Complétez les commentaires dans le programme ci-dessous en donnant les valeurs des variables a, b et c.*

Le moniteur série

Comme vous l'avez sûrement remarqué, l'Arduino de base n'a pas d'écran.
C'est là qu'entre en scène le **moniteur série**, qui est une fenêtre que l'on peut ouvrir en cliquant sur **la loupe** du logiciel Arduino. Cette fenêtre reçoit et envoie des infos aux ports concernés.
Donc grâce au moniteur série, l'Arduino peut (à condition d'être connecté à un PC) envoyer des infos à l'ordinateur qui va pouvoir les afficher en temps réel.
Pour ce faire on fait appel à une bibliothèque spéciale, déjà incluse dans l'IDE : **la bibliothèque Serial.**

```
void setup()
{
  Serial.begin(9600); // initialisation de la communication
  Serial.println("Communication initialisée"); // envoi d'un message
}
void loop()
{
  Serial.println("Je suis dans la boucle !"); //envoi d'un autre message
}
```

Téléversez ce programme, puis cliquez sur le bouton d'affichage du moniteur (la loupe) et observez ce qu'il se passe.

Pour comprendre le principe d'utilisation d'une bibliothèque :

- Les codes liés à une bibliothèque commencent tous par son nom (ici Serial) suivi d'un point et de la commande.
- La commande est spécifique à la bibliothèque utilisée.
- La liste des commandes liées à une bibliothèque est ce qu'on nomme la documentation. Elle est souvent liée à la bibliothèque.

Exercice 3 *: Modifier ce programme pour que le message dans la boucle ne soit envoyé que toutes les deux secondes.*

Exercice 4 *: Ecrivez le programme suivant : à l'ouverture du moniteur, un message qui affiche sur trois lignes "Hello Arduino World !" avec un mot par ligne (le point d'exclamation avec le dernier mot). Le message ne doit pas se répéter à l'infini.*

A retenir :
Serial.println() : affiche avec retour à la ligne
Serial.print() : affiche sans retour à la ligne

```
int compteur; //déclaration d'une variable compteur

void setup()
{
  Serial.begin(9600); //initialisation communication
  compteur=1; // initialisation de compteur
}

void loop()
{
  Serial.println(compteur); //affiche la valeur de compteur
  compteur=compteur+1; //on ajoute 1 à compteur
}
```

Ecrivez et téléversez ce programme. Décrivez ce qu'il se passe.

Le fait d'ajouter 1 à une variable s'appelle **incrémenter**, et le fait d'enlever 1 c'est **décrémenter**.
Ce sont des mots de vocabulaire à connaître.

Les conditions

La condition est un test que va réaliser le programme en se posant une question. S'il peut y répondre « vrai » alors il exécute un bloc d'action (qui sera mis entre accolades), sinon il ne l'exécute pas.

```
int affichageFait; // déclaration de la variable

void setup()
{
  Serial.begin(9600);
  affichageFait=0; //initialisation de la variable
}

void loop()
{
  if (affichageFait==0) // pas de point-virgule à ce niveau
  {
    //ce code n'est exécuté que si la condition est vérifiée
    Serial.println("Hello");
    Serial.println("Arduino");
    Serial.println("World !");
    affichageFait=1;//on passe la variable à 1 pour ne plus exécuter le code
  }
}
```

Une condition s'écrit donc avec le **mot clé if**, puis la condition à tester est mise entre parenthèses, puis le code à exécuter entre accolades. **Attention pas de point-virgule après l'écriture du if.**

if (condition) { code à exécuter }

Vous remarquerez que pour tester une valeur dans une condition on met un double égal : ==

C'est la façon de teste l'égalité dans une condition.

Voici le tableau des signes de condition :

Code	Condition testée
==	égal
>=	supérieur ou égal
<=	inférieur ou égal
>	supérieur
<	inférieur
!=	différent (non égal)

Exercice 5 *: Reprenez le programme qui permet de compter de 1 en 1 (cf. page précédente) et modifiez-le pour que le programme compte jusqu'à 20 puis s'arrête.*

Exercice 6 *: Reprenez le programme qui permet de faire clignoter la LED 13, modifiez-le pour que :*

L'Arduino affiche sur le moniteur série le nombre 1 et allume une fois la LED 13.
Puis après un temps d'arrêt d'une seconde, il doit afficher le nombre 2 et faire clignoter 2 fois la LED 13.
Puis encore un temps d'arrêt d'une seconde, puis affichage du nombre 3 et 3 clignotements de la LED 13.

Et si je voulais que l'opération se déroule jusqu'à 20….
Vous voyez que dans l'état actuel de vos connaissances, cette programmation devient rébarbative.
Il existe cependant une solution : **les boucles** que nous verrons dans la séance 4.

Dans le programme en haut de la page 3, vous aviez utilisé une variable **affichageFait** qui prenait la valeur 0 pour permettre l'affichage du message puis passait à 1 et donc empêchait la réécriture du message.
Sans le savoir vous avez utilisé une variable d'un type un peu spécial : **le type booléen**
Ce sont des variables qui ne peuvent prendre que deux valeurs : soit true (ou 1), c'est-à-dire vrai, soit false (ou 0), c'est-à-dire faux.

Les booléens sont des variables pratiques pour faire des tests comme celui dont nous avons besoin pour vérifier si le compteur de notre programme "Arduino compte seul" a fini de compter jusqu'à 20.
On les déclare de la manière suivante :

boolean nomdelavariable ;

Exemple :

```
boolean etat; //déclaration de la variable etat de type boolean
etat=true; //initialisation de etat à VRAI
if (etat==true) // test si etat est VRAI
{
   //bloc de code exécuté si etat est VRAI
}
```

Rq. On peut écrire if (etat==true) ou if (etat) qui signifie la même chose.

Exercice 7 *: Reprenez le programme en haut de la page précédente, et modifiez-le avec une variable booléenne.*

A ce niveau, vous devez avoir compris l'intégralité de ce que vous avez écrit et programmé. Si ce n'est pas le cas, relisez le cours, refaites les exercices.

Exercice 8

```
**********************
Table de multiplication
La table de : 7

0 x 7 = 0
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
11 x 7 = 77
12 x 7 = 84
13 x 7 = 91
14 x 7 = 98

**********************
```

Afficher la table de multiplication par 7 des nombres de 0 à 14.

- ***Le programme devra afficher la table de multiplication une seule fois, il faudra donc utiliser une condition pour vérifier si elle a été affichée.***
- ***Pour rendre le programme facile à modifier, vous stockerez le nombre 7 dans une variable de type int.***
- ***Le coefficient multiplicateur :0, 1, 2, 3, 4 etc. devra également être placé dans une variable, celle-ci sera donc incrémentée à chaque tour de boucle.***
- ***Le moniteur devra afficher une table qui ressemble à l'image ci-contre***