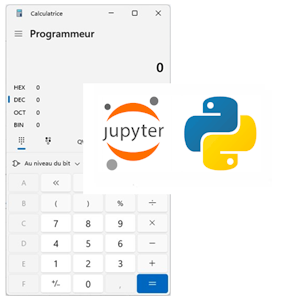
|  |  |  |
| --- | --- | --- |
| **Logo_Lycée** | **Représentation des nombres en machine** | logo%20ac%20orl%E9ans%20toursDescription : Description : Description : pemDescription : Description : Description : pem |
| **TP1** |



**Nom :** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Classe :** \_\_\_\_\_\_\_\_\_\_\_\_

**Prénom** :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Groupe** : \_\_\_\_\_\_\_\_\_\_\_\_

**Date** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Ressource****: le mode programmeur de la calculatrice Windows https://urlr.me/JMR3q*

***CAPACITÉS***

Passer de la représentation d’une base à une autre (10, 2, 16). Effectuer des opérations sur des nombres codés en complément à 2. Choisir le type d’une donnée, d’une variable.

# 1. Introduction

La représentation des nombres en machine fait référence à la manière dont les **ordinateurs stockent et manipulent les nombres**.

En informatique, il existe plusieurs systèmes de représentation des nombres, tels que la représentation **binaire**, **octale**, **décimale** et **hexadécimale**.

La représentation **binaire** est la plus **fondamentale**, car les ordinateurs utilisent des circuits électroniques qui manipulent des signaux électriques (0V ou 5V) pour effectuer des opérations. Chaque chiffre binaire (**0** ou **1**) est appelé un "**bit**", et un groupe de **8 bits** est appelé un "**octet**".

En utilisant la représentation binaire, **on peut représenter des nombres entiers et une approximation des réels, ainsi que des caractères et des instructions** pour les **processeurs**.

# 2. Utilisation de la calculatrice Windows en mode "Programmeur"

## Mode programmeur

En mode **Programmeur** la calculatrice est essentiellement utilisée pour effectuer des opérations en **binaire (BIN)**, en octal (OCT), en **décimal (DEC)** ou en **hexadécimal (HEX)**, ou encore des conversions de bases de numération.

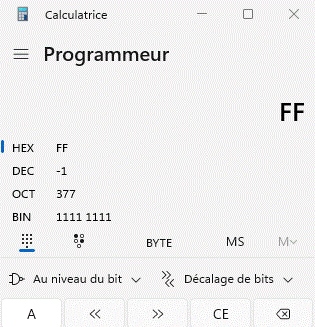
**Commandes**

**- CE** (**C**lear **E**ntry) efface la dernière entrée que vous avez saisie, que ce soit un nombre ou une opération.

- **C** (**C**lear) réinitialise complètement la calculatrice.

Entrez « **Calculatrice** » dans **la zone de recherches de la barre des tâches** et passez en mode programmeur.

## Identification des fonctionnalités de la calculatrice



**Cliquez** sur QWORD jusqu’à faire apparaître **BYTE**. Sélectionnez **HEX** et entrez **FF** au clavier.

La calculatrice doit **correspondre** à la copie d’écran ci-contre. Cliquez sur le bouton **BYTE** pour accéder aux autres **types de données\*** et complétez le tableau ci-dessous.

\* "**Type de donnée**" se réfère au **format d'information** qu'une variable peut prendre dans un programme informatique.

|  |  |  |
| --- | --- | --- |
| Le **type de donnée** |  | |
| **Byte** code un nombre binaire sur | 8 | bits |
| **WORD " "** |  |
| **DWORD " "** |  |
| **QWORD " "** |  |

## Complément à 2

1. **Objectif**: identifier l’effet du type d’une donnée binaire sur sa représentation en décimal.

* Sélectionnez le type de donnée puis entrez la valeur dans le champ HEX pour convertir 7F16, FF16 et FFFF16 en binaire et en décimal. Complétez le tableau ci-dessous.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Type de donnée** | **Hexadécimal** | **Binaire** | **Décimal** |
| 1 | **BYTE** | 7F16 |  |  |
| 2 | **WORD** | FF16 |  |  |
| 3 | **DWORD** | FFFF16 |  |  |

Ne vous contentez pas de recopier le résultat donné par la calculatrice, écrivez dans la colonne Binaire le nombre de bits conforme au type de donnée

* Nous conservons les nombres n°2 et n°3 mais **modifions leur type de donnée**. Sélectionnez le type de donnée puis entrez la valeur dans le champ HEX pour convertir FF16 et FFFF16 en **binaire** et endécimal. Complétez le tableau ci-dessous.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Type de donnée** | **Hexadécimal** | **Binaire** | **Décimal** |
| 2 | **BYTE** | FF16 |  |  |
| 3 | **WORD** | FFFF16 |  |  |

Ne vous contentez pas de recopier le résultat donné par la calculatrice, écrivez dans la colonne Binaire le nombre de bits conforme au type de donnée

* Comparez les résultats des nombres n°2 et n°3 dans les deux tableaux. Que remarquez-vous ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Journal avec un remplissage uni | Ce que vous venez de constater est dû au **codage des nombres** dans les machines numériques **appelé complément à 2.**  **Complément à 2n**  Le complément à 2n ou complément à 2, est une méthode utilisée en informatique pour représenter les **nombres négatifs**.  Cette méthode est couramment utilisée pour effectuer des **opérations arithmétiques** sur les **nombres binaires**, en particulier lorsqu'il s'agit de représenter des **nombres négatifs dans un format binaire**. En complément à 2, un nombre N10 se situe dans l’intervalle suivant :  **- 2n-1≤ N10 ≤ 2n-1 – 1**  n = nombre de bits de la représentation binaire de N.  Dans cette représentation si le **MSB** est égal à **1** alors le nombre est **négatif**. Si le **MSB** est égal à **0** alors le nombre est **positif**. |

1. **Objectif :** déterminer le nombre de bits nécessaires au codage d’un nombre décimal en binaire complément à 2.

* Convertissez 1210 et -410 en binaire complément à 2 et en hexadécimal sur **un** **octet (BYTE)**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de donnée** | **Décimal** | **Hexadécimal** | **Binaire** |
| **BYTE** | **1210** |  |  |
| **BYTE** | **-410** |  |  |

Ne vous contentez pas de recopier le résultat donné par la calculatrice, écrivez un résultat conforme au type de données dans les colonnes Binaire et Hexadécimal.

* Donnez le nombre minimum de bits qu’il faut pour représenter 1210 et -410 en binaire complément à 2. Donnez l’intervalle dans lequel se situe le nombre N10, codé en binaire complément à 2, si n = 5.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Avec n = 5, \_\_\_\_\_\_\_\_≤\_\_N10\_\_≤\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Complétez** le tableau ci-dessous

|  |  |  |
| --- | --- | --- |
| **Type de donnée** | **Intervalle en puissance de 2** | **Valeurs entières dans l’intervalle** |
| **BYTE** | **-27 ≤ N10 ≤27-1** | **-128 ≤ N10 ≤ 127** |
| **WORD** | ≤ N10 ≤ | ≤ N10 ≤ |
| **DWORD** | ≤ N10 ≤ | ≤ N10 ≤ |
| **QWORD** | ≤ N10 ≤ | ≤ N10 ≤ |

1. **Sélectionnez** le type de donnée minimum pour les opérations ci-dessous et donnez le résultat en décimal et en hexadécimal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de donnée** | **Opération** | **Décimal** | **Hexadécimal** |
|  | 1 568 21010 - 471010 = |  |  |
|  | 471010 - 1 568 21010 = |  |  |

Ne vous contentez pas de recopier le résultat donné par la calculatrice, écrivez un résultat conforme au type de données dans la colonne Hexadécimal

|  |  |
| --- | --- |
| Avertissement | **On remarque comme précédemment que la calculatrice n’étend pas le bit de signe lorsque celui-ci est égal à 0.**  Exemple : On sélectionne **BYTE** et on entre 1510 dans DEC. La calculatrice affiche 11112 au lieu de 000011112 |

1. Les valeurs ci-dessous sont codées en **complément à 2.** Sachant que ces deux opérations ont pour résultat la valeur **610**, quel type de donnée minimum faut-il choisir. Effectuez les opérations et donnez le résultat en binaire et en hexadécimal.

**Calcul d’adresses mémoires**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de donnée** | **Opération** | **Binaire** | **Hexadécimal** |
|  | 1C16 + EA16 = |  |  |
|  | 001C16 +FFEA16 = |  |  |

On choisit le type de données **DWORD.**  Effectuez les opérations et donnez le résultat en binaire et en hexadécimal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de donnée** | **Opération** | **Binaire** | **Hexadécimal** |
| DWORD | 1C16 + EA16 = |  |  |
| DWORD | 001C16 +FFEA16 = |  |  |

* Que faut-il faire pour retrouver la bonne valeur ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Journal avec un remplissage uni | **Méthode de calcul du complément à 2n oucomplément à 2**   * Pour représenter un **nombre positif m**, on utilise sa représentation en binaire, précédée d’autant de zéro que nécessaire pour avoir n bits au total.   Exemple : 510 = 01012. Sa représentation sur un octet est : 0000 01012   * Pour représenter un **nombre négatif** **-m** :  1. Coder en binaire le nombre m comme précédemment avec un nombre de bits n tel que : **- 2n-1≤ m10 ≤ 2n-1 – 1** 2. Calculer le complément à 1 de m (c.-à-d. inverser tous ses bits) 3. Additionner 1 au complément à 1   Exemple : représentation en binaire complément à 2 de -6810  1. 6810 = 10000102 est codé sur 7bits. Avec n = 7 bits on peut représenter des nombres compris entre -64 et 63  Il faut donc ajouter un 0 à gauche pour coder -6810. Donc 6810 = 010000102 est maintenant codé sur 8bits  2. Le complément à 1 de 010000102 est : 101111012  3. Enfin, on effectue l’opération : 1  101111012  + 1  ----------------  = 101111102 = -6810    Remarque : **en cas de débordement** (nombre de bits du résultat > n), **le bit de poids fort est ignoré**. |

1. **Calcul (à la main) en complément à 2**

Convertissez 1C16 et EA16 en binaire. Lequel de ces deux nombres est négatif ? **Écrivez** l’opération 1C16 + EA16 en binaire.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 3. Les types de données dans les langages de programmation

**3.1 Introduction**

Un **langage de programmation "typé"** est un langage dans lequel **les variables** doivent être déclarées avec un type spécifique (comme **booléens**, **entier**, **flottant**, **chaîne de caractères**, etc.) et **ne peuvent pas changer de type après la déclaration**. Cela permet au compilateur ou à l'interpréteur de détecter certaines erreurs de type avant l'exécution du programme.

Un **langage de programmation** "non typé" (ou "**dynamiquement typé**") permet aux variables de changer de type au cours de l'exécution du programme. Cela signifie que **vous n'avez pas besoin de déclarer explicitement** le type d'une variable, et **celui-ci peut être déterminé lors de l'exécution**.



Exemples de langages de programmation **typés** :

**C**, C++

Java, C#

Exemples de langages de programmation **dynamiquement typés** :

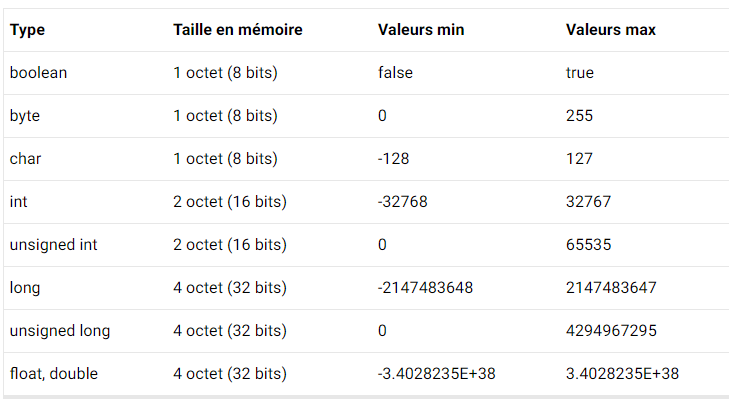


**Python**

JavaScript

**3.2 Types numériques**

* **Arduino (C, C++)**





* **Python**



|  |  |
| --- | --- |
| Avertissement avec un remplissage uni | Tous les nombres en Python sont des objets immuables c.-à-d. qu’une opération sur un nombre crée un autre nombre. Les **littéraux numériques n’incluent pas de signe** : un + ou un – devant un nombre est un opérateur distinct. |

|  |  |  |  |
| --- | --- | --- | --- |
| *Types natifs* | *Taille en mémoire* | *Valeurs*  *(min, max)* | *Représentation* |
| Entier (int) | Non limité, dépend de la machine | - | Décimaux : 1, 2, 300  Binaires : 0b010001, 0b01, 0b1100010  Octaux : 0o1, 0o27, 0o777  Hexadécimaux : 0x01, 0x17, 0xff, 0xdae15 |
| Virgule flottante (float) | Type double en C (53bits) | 2.2250738585072014e-308  1.7976931348623157e+308  PC sous Windows 64bits | 1.23, -12.0, -1,2E-28 |

Mis en œuvre dans le notebook Jupyter "Nombres"

Autre type natif : complexe

Pour aller plus loin : [Vidéo sur les nombres](https://www.youtube.com/watch?v=vvjRuVJQ7Kg) en Python