# Solidity

Amnézic

# Contents

# 1 Introduction

- hautement typé

# 2 Type

(Commence à la page 53 du cours)

## 2.1 Types de variables

### 2.1.1 Booléen :

Les deux seules valeurs possibles pour une variables booléenne sont: true ou false. On peut bien évidemment leur appliquer les opérations booléennes usuelles:

- négation : !
- et : &&
- ou : ||
- égalité : ==
- inégalité : !=

### 2.1.2 Entiers :

Signés et non-signés : Solidity inclut les uint (entiers non-signés) et int (entiers signés) dans les nombres entiers et permet d'appliquer les mêmes opérations aux uint qu'aux int, sans pour autant pouvoir les mélanger entre eux (langage hautement typé).

Rappel :

> La différence entre un entier signé et non-signé est la plage de nombre qu'il représente : les non-signés ne permettent pas de représenter les nombres négatifs. Soit un entier m codé sur n bits, alors :
>
> - si m est non-signé, alors m$\in$[0;$2^n$-1]
> - si m est signé, alors m$\in$[-$2^{n-1}$-1;$2^{n-1}$-1]

Astuce :

> Pour obtenir la valeur minimale ou maximale d'un type (numérique), il suffit d'utiliser la méthode type(<type>).min/max.

Les entiers peuvent être codés sur 8 à 256 bits (par pas de 8 : int8, int16, int32, ..., int248, int256). Si le nombre de bits n'est pas précisé, le nombre sera codé sur 256 bits par défaut.

Les opérations : On peut appliquer aux entiers les opérations suivantes :

- opérateurs arithmétiques : +, -, *, /, %, ** (x**n = $x^n$)
- comparateurs : ==, !=, <, <=, >=, >
- opérateurs de décalage :
  - à gauche : x $\ll$ n $\rightarrow$ rajoute n 0 à la droite de la représentation binaire de x
  - à droite : x $\gg$ n $\rightarrow$ on "supprime" les n bits de poids faible et on rajoute n fois 0 ou 1 (en fonction de la positivité de x) à la gauche du x obtenu
- opérateurs sur les bits : &(and), |(or), ^ (xor), $\sim$ (not) $\rightarrow$ on applique l'opérateur sur chaque pair de bit des nombres

## 2.2 Nombres à virgule :

**Nombres à virgule fixe :** Les nombres à virgule fixe sont un entre-deux entre les nombres entiers et flottants : les nombres entiers n'ont aucune partie décimale tandis que les nombres flottants ont un nombre théoriquement infini de chiffres après la virgule. Les nombres à virgule fixe est donc un type de données qui est constitué d'un nombre entier m de chiffres avant la virgule et d'un nombre entier n de chiffres après la virgule. Comme pour les entiers, le préfixe $u$ précise si le nombre est signé ou non. Par exemple :

- fixed128x18 sera un nombre (possiblement négatif) qui aura 128 chiffres avant la virgule et 18 chiffres après
- ufixed16x2 sera un nombre (obligatoirement positif car non-signé) qui aura 16 chiffres avant la virgule et 2 chiffres après

Important :

Avec ce type de données, m$\in$[8,16,32,...248,256] et n$\in$[0;80]

**Fixed point numbers are not fully supported by Solidity yet. They can be declared, but cannot be assigned to or from**