Travailler avec différentes bases numériques

(Note: Ces exemples utilisent le symbole ^ pour signifier 'élevé à la puissance de' - ceci est seulement une convention dans ce document, PureBasic n'a pas actuellement d'opérateur d'élévation à la puissance ! Utilisez plutôt la commande PureBasic Pow() de la bibliothèque "Math".)

Introduction

Une base numérique est une manière de représenter les nombres, en utilisant une certaine quantité de symboles possibles par chiffre. La plus courante que vous devez connaître est la base 10 (c-a-d décimale), car il y a 10 chiffres utilisés (0 à 9), et c'est celle que la plupart des humains peuvent manipuler le plus facilement.

Le but de cette page est d'expliquer différentes bases numériques et comment vous pouvez travailler avec elles. En effet les ordinateurs travaillent en binaire (base 2) et il y a certains cas où il est avantageux de le comprendre (par exemple, lors de l'utilisation d'opérateurs logiques, masques de bits, etc).

Vue d'ensemble des bases numériques

Le système Décimal

Pensez à un nombre décimal, et réfléchissez à sa représentation en colonnes. Prenons par exemple le nombre 1234. Séparé en colonnes, nous avons :

```
  1   2   3   4
```

Réfléchissez à ce que sont les en-têtes de chaque colonne. Nous savons qu'il y a les unités, les dizaines, les centaines et les milliers, présentés ainsi :

```
1000  100   10    1
   1    2    3    4
```

Nous pouvons voir que le nombre 1234 est constitué de

```
   1*1000=1000
+ 2* 100= 200
+ 3*  10=  30
+ 4*   1=   4
Total   =1234
```

Si nous regardons également les en-têtes de colonne, vous verrez qu'à chaque fois que vous bougez d'une colonne vers la gauche nous multiplions par 10, qui justement s'avère être la base numérique. Chaque fois que vous bougez d'une colonne vers la droite, vous divisez par 10. Les en-têtes de colonnes peuvent être appelées les poids, puisque pour obtenir tout le nombre nous devons multiplier les chiffres de chaque colonne par le poids. Nous pouvons exprimer les poids en utilisant des index. Par exemple 10^2 signifie '10 élevé à la puissance de 2' ou 1*10*10 (=100). De même, 10^4 signifie 1*10*10*10*10 (=10000). Remarquez dans ces exemples, que peu importe la valeur de l'index, il correspond au nombre de fois que nous multiplions le nombre qui doit être élevé. 10^0 signifie 1 (puisque nous multiplions par 10 aucune fois). Utiliser des nombres négatifs montre que nous devons diviser, par exemple 10^-2 signifie 1/10/10 (=0.01). Les valeurs de l'index prennent plus de sens quand nous attribuons un numéro à chaque colonne - vous verrez souvent des choses comme 'bit 0' qui signifie en fait 'chiffre binaire en colonne 0'.

```
Dans cet exemple, ^ signifie élevé à la puissance de, donc 10^2 signifie 10 élevé à la puissance de 2.
Numéro de colonne          3     2     1     0
Poids (index)           10^3  10^2  10^1  10^0
Poids (valeur réelle)   1000   100    10     1
Nombre exemple (1234)      1     2     3     4
```

Précédemment nous avons vu comment convertir le nombre 1234 dans son équivalent décimal. Conversion peu justifiée, car il était déjà en décimal mais nous pouvons en tirer la méthode générale - voici donc comment convertir n'importe quel nombre en sa valeur décimale :

```
B = Valeur de la base numérique

1) Séparer le nombre, peu importe la base, en colonnes. Par
   exemple, si nous avions la valeur 'abcde' dans notre base numérique fictive 'B',
   les colonnes seraient : a   b   c   d   e

2) Multiplier chaque symbole par le poids de chaque colonne (le poids étant
   calculé par 'B' élevé à la puissance du numéro de la colonne):
     a * B^4 = a * B * B * B * B
     b * B^3 = b * B * B * B
     c * B^2 = c * B * B
     d * B^1 = d * B
     e * B^0 = e

3) Calculer la somme de toutes ces valeurs. En écrivant toutes ces valeurs dans leur
   équivalent décimal pendant les calculs, il devient beaucoup plus facile de voir le
   résultat and de faire le calcul (si nous convertissons en décimal).
```

Convertir dans la direction opposée (de décimal vers la base numérique 'B') s'effectue en utilisant la division au lieu de la multiplication :

```
1) Commencer par le nombre décimal que vous voulez convertir (e.g. 1234).

2) Diviser par la base numérique ciblée ('B') et prendre note du quotient et du
   reste.

3) Diviser le quotient de (2) par la base numérique ciblée ('B') et prendre note du
   quotient et du reste.

4) Continuer à diviser comme ça jusqu'à ce que vous obteniez un quotient de 0.

5) Votre nombre dans la base numérique ciblée est constitué des restes écrits dans l'ordre
   du dernier calculé au premier calculé. Par exemple, votre nombre serait constitué
   des restes des étapes dans cet ordre 432.
```

Des exemples plus particuliers seront donnés dans les paragraphes concernant les bases numériques particulières.

Système binaire

Tout dans un ordinateur est stocké en binaire (base 2, avec les symboles définis '0' ou '1') mais le travail avec des nombres binaires suit les mêmes règles qu'en décimal. Chaque symbole dans un nombre binaire est appelé bit, abréviation de BInary digiT (chiffre binaire).

Généralement, vous travaillerez avec des bytes (octets, 8-bit), words (mots, 16-bit) ou longwords (mots longs, 32-bit) car ce sont les tailles par défaut des types intégrés à PureBasic. Les poids pour un byte (octet) sont les suivants :

```
(^ signifie 'élevé à la puissance de', la base numérique est 2 pour le binaire)
Bit/numéro de colonne      7     6     5     4     3     2     1     0
Poids (index)            2^7   2^6   2^5   2^4   2^3   2^2   2^1   2^0
Poids (valeur réelle)    128    64    32    16     8     4     2     1
```

Donc, par exemple, si nous avions le nombre 00110011 (Base 2), nous pourrions établir sa valeur comme ceci :

```
  0 * 128
+ 0 *  64
+ 1 *  32
+ 1 *  16
+ 0 *   8
+ 0 *   4
+ 1 *   2
+ 1 *   1
=      51
```

Un exemple de conversion inversée serait d'écrire la valeur 69 en binaire. Nous ferions comme ceci :

```
69 / 2 = 34 r 1     ^
34 / 2 = 17 r 0    /|\
17 / 2 =  8 r 1     |
 8 / 2 =  4 r 0     |    Lire les restes dans cette direction
 4 / 2 =  2 r 0     |
 2 / 2 =  1 r 0     |
 1 / 2 =  0 r 1     |
(Arrêt ici car le quotient de la dernière division était 0)

Lire les restes de bas en haut pour obtenir la valeur en binaire = 1000101
```

Une autre chose à considérer lorsque l'on travaille avec des nombres binaires est la représentation des nombres négatifs. Comme nous le faisons quotidiennement, nous pourrions simplement mettre un symbole négatif devant le nombre décimal. Nous ne pouvons pas faire cela en binaire, mais il y a un moyen (après tout, PureBasic travaille principalement avec des nombres signés, et doit donc être capable de gérer les nombres négatifs). Cette méthode est appelée 'complément à deux' et indépendamment de toutes les bonnes fonctionnalités que cette méthode possède (et qui ne seront pas expliquées ici, pour épargner une certaine confusion) la manière la plus simple de la comprendre, est de se dire que le poids du bit le plus significatif (Most Significant bit / le MSb est le numéro du bit avec la plus grande valeur. Dans le cas d'un octet (byte), ce serait le bit 7) est réellement une valeur négative. Donc pour un système de complément à deux, les poids des bits sont modifés en :

```
(^ signifie 'élevé à la puissance de', la base numérique est 2 pour le binaire)
Bit/numéro de colonne      7     6     5     4     3     2     1     0
Poids (index)           -2^7   2^6   2^5   2^4   2^3   2^2   2^1   2^0
Poids (valeur réelle)   -128    64    32    16     8     4     2     1
```

et vous pourrez faire la conversion de binaire vers décimal exactement de la même manière que ci-dessus, mais en utilisant le nouveau jeu de poids. Par exemple, le nombre 10000000 (Base 2) est -128, et 10101010 (Base 2) est -86.

Pour convertir un nombre binaire positif en négatif et vice versa, vous inversez tous les bits et ensuite ajoutez 1. Par exemple, 00100010 passerait en négatif en inversant -> 11011101 et additionant 1 -> 11011110.

Ceci facilite la conversion de décimal à binaire, car vous pouvez convertir les nombres décimaux négatifs comme leurs équivalents positifs (en utilisant la méthode ci-dessus) et ensuite rendre le nombre binaire négatif à la fin.

Les nombre binaires dans PureBasic sont précédés du symbol 'pourcentage', et évidement tous les bits du nombre doivent être un '0' ou un '1'. Par exemple, vous pouvez utiliser la valeur %110011 dans PureBasic pour signifier 51. Notez que vous n'avez pas besoin de mettre les '0' de tête (ce nombre serait vraiment %00110011) mais cela peut faciliter la lisibilité de votre source si vous écrivez l'ensemble des bits.

Le système Hexadécimal

L'hexadecimal (pour base 16, symboles '0'-'9' suivis de 'A'-'F') est une base numérique qui est plus généralement employée en informatique, car il s'agit probablement de la base numérique non décimale la plus facile à comprendre pour les humains, et vous n'avez pas à écrire de longues chaines de symboles pour vos nombres (comme c'est le cas avec le binaire).

Les mathématiques hexadécimales suivent les mêmes règles qu'en décimal, bien que vous ayez maintenant 16 symboles par colonne jusqu'à ce que vous ayez besoin d'une retenue. La conversion entre l'hexadécimal et le décimal suit les mêmes principes qu'entre le binaire et le décimal, sauf que les poids sont des multiples de 16 et les divisions sont faites par 16 au lieu de 2 :

```
Numéro de colonne          3      2      1      0
Poids (index)           16^3   16^2   16^1   16^0
Poids (valeur réelle)   4096    256     16      1
```

Convertir la valeur hexadécimale BEEF (Base 16) en décimal serait fait comme ceci :

```
  B * 4096 = 11 * 4096
+ E *  256 = 14 *  256
+ E *   16 = 14 *   16
+ F *    1 = 15 *    1
=               48879
```

Et convertir la valeur 666 en hexadécimal serait fait comme ceci :

```
666 / 16 = 41 r 10    ^
 41 / 16 =  2 r  9   /|\    Lire les chiffres dans cette direction, se souvenir de convertir
  2 / 16 =  0 r  2    |     en chiffres hexadecimaux où nécessaire
(Arrêt ici car le quotient de la dernière division était 0)
La valeur hexadécimale de 666 est 29A
```

La chose vraiment intéressante avec l'hexadécimal c'est qu'il permet également de convertir vers le binaire très facilement. Chaque chiffre hexadécimal représente 4 bits, pour convertir entre hexadécimal et binaire, vous devez juste convertir chaque chiffre hexadécimal en 4 bits ou chaque groupe de 4 bits vers un chiffre hexadécimal (en se rappelant que 4 est un diviseur de toutes les tailles communes de nombres binaires dans un CPU). Par exemple :

```
Nombre hexadécimal     5        9        D        F          4E
Valeur binaire      0101     1001     1101     1111       01001110
```

Lorsque vous utilisez des valeurs hexadécimales dans PureBasic, vous mettez un symbole 'dollar' devant le nombre, par exemple $BEEF.

Les conversions Hexadécimal <-> Binaire

Il est aisé de convertir un nombre hexadécimal en un nombre binaire et inversement, en utilisant les 'nibles'. Un nible est un groupe de 4 bits, appelé demi octet ou quartet. Un quartet contenant 4 bits, il peut donc prendre seize (2^4) valeurs différentes et correspond donc à un chiffre en hexadécimal. Deux chiffres hexadécimaux formant un octet, ce dernier est souvent représenté par deux nibbles.

Par exemple:

```
  Numéro de colonne          3      2      1      0
  Poids (index)            2^3    2^2    2^1    2^0
  Poids (valeur réelle)      8      4      2      1
                           ------------------------
                  $7         0      1      1      1  : 0*8 + 1*4 + 1*2 + 1*1 = 7
                  $C         1      1      0      0  : 1*8 + 1*4 = 12 = $C
```

Comment s'écrit $F2 en binaire ?

```
  Poids (valeur réelle)      8  4  2  1 |  8  4  2  1
                           ------------|-------------
                  $F2       1  1  1  1    0  0  1  0 : 1*8 + 1*4 + 1*2 + 1*1 = 15 = $F
                                                      et 0*8 + 0*4 + 1*2 + 0*1 =  2 = $2
```

Facile non !