TD2 : Conversion binaire décimal

Ce TD sera noté ! Il y a une partie à faire **sur papier et à rendre en main propre**, et une partie sur machine à rendre sur le GoogleDrive, dans le répertoire **BTS SIO/Math/Algo**. Créez un répertoire personnel en utilisant votre nom, pour que le chemin ressemble à ceci : **BTS SIO/Math/Algo/DEBIZE**

1 Le problème

Nous avons déjà vu en cours comment convertir un nombre écrit en base 2 en son équivalent en base 10. Rappel : pour convertir $N = (11011)_2$, il faut faire le calcul suivant :

$$\begin{aligned} N &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 16 + 8 + 0 + 2 + 1 \\ &= 27 \end{aligned}$$

Le but de ce TD est de reproduire ce calcul. Le programme demandera à l'utilisateur de rentrer un nombre écrit en binaire. Il faudra enregistrer ce nombre dans une variable de type *chaine*.
Pour convertir ce nombre, il faudra ensuite parcourir la chaine de la droite vers la gauche, et convertir chaque caractère en le multipliant avec sa puissance de 2 correspondante. On pourra stocker dans une variable la valeur de la puissance de 2 correspondante et la mettre à jour lors du parcours de la chaine.

Pour le parcours de la chaine, vous pourrez avoir besoin de la fonction `longueur()`. En C++, on l'utilise comme suit, `chaine` étant notre chaine :

```
chaine.length()
```

Il ne faudra pas oublier d'inclure la bibliothèque `string.h` pour utiliser les chaines et cette fonction.

Quand vous aurez besoin de convertir un caractère, vous risquez de remarquer que ça ne vous donne pas du tout le bon résultat. Par exemple, le nombre binaire 10 va être converti en 146 ! C'est parce qu'il faut d'abord convertir le caractère en entier avant de faire des calculs dessus. Je m'explique :
Le premier caractère que vous voulez convertir est '0', cependant, il ne vaut pas 0 ! Comment ça se fait ? Parce que '0' est un caractère ASCII, et pas le nombre 0, et ce caractère est le 48^e de la table ASCII, donc si vous utilisez '0' dans des calculs, la machine va comprendre 48 et pas 0. De même '1' ne sera pas compris comme 1, mais comme 49, et donc le résultat de la conversion de 10 sera :
$49 \times 2 + 48 = 146$ au lieu de $1 \times 2 + 0 = 2$ ... embêtant !

Par contre, les caractères '0' à '9' sont consécutifs et dans l'ordre croissant dans la table ASCII, une chance ! Une technique est donc de soustraire '0' au caractère que nous lisons pour avoir réellement l'entier correspondant. Exemple, si notre caractère est `c = '1'` :
`c - '0'` renverra bien 1 (la machine fera en réalité 49-48).

2 Questions :

1. Ecrire en pseudo-code (sur papier et en langage algorithmique) une fonction `bin2dec` qui prend en paramètre une chaine de caractère contenant le nombre en binaire, et qui renvoie le nombre converti en base 10. La signature de la fonction sera donc :

   ```
   bin2dec(binaire : chaine) : entier
   ```

2. Ecrire l'algorithme en pseudo-code (sur papier) du programme principal qui demandera à l'utilisateur le nombre à convertir, utilisera cette fonction, et affichera à l'écran le résultat. L'interface devra être de ce type :

   ```
   Entrez un nombre en binaire à convertir en décimal
   110
   Votre nombre en décimal : 6
   ```

3. Implémenter en C++ la fonction `bin2dec` ainsi que la programme principal et testez le avec quelques nombres simples pour vérifier le bon comportement.