Conversions

I Conversions Manuelles

Exercice 1. *Binaire*

1. Écrire 178 en binaire sur 8 bits.
2. Convertir `0110 0111` en décimal.

Exercice 2. *Hexadécimal*

1. Donner 456 en hexadécimal.
2. Convertir `1212` écrit en base 16, en décimal.

Exercice 3. *Binaire Complément à Deux*

1. Écrire -79 en binaire complément à deux sur 8 bits.
2. Convertir `1001 0000 1010 0001` écrit en binaire complément à 2 sur 16 bits, en décimal.

II Conversion automatiques

Pour toutes les fonctions de cette section on essayera de produire une version impérative et une version récursive.

Exercice 4. *Conversion dans un sens …*

Implémentez une fonction `string_vers_entier(s : str) -> int` qui prend en entrée une chaîne de caractère `s` composée uniquement de caractères parmi `"0123456789"` et renvoie l'entier correspondant.

On pourra utiliser **`ord`** qui donne le code ASCII d'un caractère et le fait que dans la table ASCII, les chiffres sont consécutifs et dans l'ordre numérique.

Il est évidemment interdit d'utiliser la fonction **`int`**.

Exercice 5. *… puis dans l'autre …*

Implémentez une fonction `entier_vers_chiffres(n : int) -> list[int]` qui prend en entrée un entier `n` et renvoie la liste de ses chiffres. Le chiffre des unités sera en position 0, celui des dizaines en 1, etc.

Exercice 6. *… mais en chaîne …*

Implémentez ensuite une fonction `entier_vers_string(n : int) -> str` qui prend en entrée un entier `n` et renvoie la chaîne de caractère qui représente l'entier. Vous devrez utiliser la fonction précédente.

On pourra utiliser **`chr`** qui donne le caractère associé à un code ASCII et le fait que dans la table ASCII, les chiffres sont consécutifs et dans l'ordre numérique.

Il est évidemment interdit d'utiliser la fonction **`str`**.

Exercice 7. *… puis encore dans un sens…*

Implémentez une fonction `binaire_vers_decimal(b : list[int]) -> int` qui prend en entrée la liste `b` des chiffres de l'écriture binaire d'un nombre et renvoie l'entier correspondant. Les chiffres seront donnée du bit de poids faible au bit de poids fort. Par exemple : `[0,1,1,0,1]` représente l'entier 22.

Exercice 8. *… puis enfin dans l'autre.*

Implémentez une fonction `decimal_vers_binaire(n : int) -> list[int]` qui prend en entrée un entier `n` et renvoie la liste `b` des chiffres de l'écriture binaire de n. Les chiffres seront donnée du bit de poids faible au bit de poids fort. Par exemple : `22` sera écrit `[0,1,1,0,1]`.

III Pour aller plus loin

Exercice 9. *Encore une conversion dans un sens …*

Implémentez une fonction `base_vers_decimal(b : list[int], base : int) -> int` qui prend en entrée la liste `b` des chiffres de l'écriture en base `b` d'un nombre et renvoie l'entier correspondant. Les chiffres seront donnée du chiffre de poids faible au chiffre de poids fort.

Exercice 10. *… et finalement dans l'autre …*

Implémentez une fonction `decimal_vers_base(n : int, base : int) -> list[int]` qui prend en entrée un entier `n` et renvoie la liste des chiffres de l'écriture en base `b` de n. Les chiffres seront donnée du chiffre de poids faible au chiffre de poids fort.

Exercice 11. *Addition, soustraction, multiplication*

Implémentez les fonctions d'addition, de soustraction et de multiplication entre deux nombres données en entrée sous la forme de la liste de leur chiffre en base `b`.