TP 3 & 4 : Structures Itératives

Consignes

— Le TP est à faire de manière individuelle.
— Il est demandé de rendre une archive `nom_prenom.zip` qui contient les fichier `.c` et `.py`.
— L'archive doit être rendue sur Moodle à la fin des 4h de TP.
— Il est conseillé d'utiliser un éditeur directement sur le PC (e.g. Visual Studio Code).

Exercice 1 : Caractères ASCII

Nous souhaitons afficher pour chaque valeur de la table ASCII (**entier** allant de 0 à 255) le **caractère** (signe ou lettre) correspondant. Il faudra afficher pour chaque nombre le caractère ASCII associé.

1. Implémenter le programme en langage C. Le fichier s'appellera `caracteres_ASCII_v1.c`.
2. Implémenter le programme en langage Python. Le fichier s'appellera `caracteres_ASCII_v1.py`.
   Aide : Vous pourrez vous aider de la fonction Python `chr()`.

```
Le caractere numero 33 est !.
Le caractere numero 34 est ".
Le caractere numero 35 est #.
Le caractere numero 36 est $.
Le caractere numero 37 est %.
```

```
Le caractere numero 64 est @.
Le caractere numero 65 est A.
Le caractere numero 66 est B.
Le caractere numero 67 est C.
Le caractere numero 68 est D.
```

Nous souhaitons maintenant saisir seulement un "nombre" ASCII et vérifier qu'il soit compris entre 0 et 255. De plus, la saisie de ce "nombre" ne sera terminée que si sa valeur est comprise entre 0 et 255.

3. Implémenter le programme en langage C. Ce fichier s'appellera désormais `caracteres_ASCII_v2.c`.
4. Implémenter le programme en langage Python. Le fichier s'appellera `caracteres_ASCII_v2.py`.

Exercice 2 : Chute libre

Nous voulons modéliser les effets de la gravité sur un objet qui tombe depuis une tour de hauteur donnée en mètres. Nous rappelons que la distance d parcourue par un objet en fonction du temps de parcours t est donnée par la formule :

$\boxed{\boldsymbol{d = \frac{1}{2} g t^2}}$ où g l'accélération de la pesanteur est égale à 9,80665 m/s^2.

De plus, il faudra :
- Afficher la hauteur à laquelle se trouve l'objet toutes les x secondes pendant sa chute.
- Écrire un message "boum" quand l'objet heurtera le sol.

1. Implémenter une solution en langage C. Le fichier s'appellera `chute.c`.
2. Implémenter une solution en langage Python. Le fichier s'appellera `chute.py`.

Exercice 3 : Nombre Parfait

Un entier naturel est **parfait** s'il est égal à la moitié de la somme de ses diviseurs.

Par exemple, $\mathbf{6 = \frac{1+2+3+6}{2}}$.

Nous souhaitons trouver tous les nombres parfaits plus petits que 10000.

1. Implémenter la solution en C dans `nombre_parfait.c`.
2. Implémenter la solution en Python dans `nombre_parfait.py`.

Exercice 4 : Notation binaire

Nous souhaitons trouver les puissances de deux qui composent un nombre entier positif. La fonction prend en paramètre ce nombre entier et affiche la somme les puissances.

Par exemple, soit le nombre **625**, le programme doit afficher :

$$\mathbf{625 = 512 + 64 + 32 + 16 + 1}$$

En effet, $\mathbf{512 = 2^9}$, $\mathbf{64 = 2^6}$, $\mathbf{32 = 2^5}$, $\mathbf{16 = 2^4}$ et $\mathbf{1 = 2^0}$.

1. Implémenter la fonction en langage C. Le fichier s'appellera `binaire.c`.
2. Implémenter la fonction en langage Python. Le fichier s'appellera `binaire.py`.

Exercice 5 : Figures géométriques

Nous souhaitons dessiner à l'écran certaines figures composées d'étoiles : sablier, losange, et papillon. Le choix de la figure (sablier, losange, papillon) et sa hauteur seront les deux paramètres de la fonction. Par exemple pour une hauteur de 9, on obtient :

Le type de figure est décidé de la manière suivante : 1- Sablier, 2- Losange et 3- Papillon.

1. Traduire cet algorithme en langage C. Ce fichier s'appellera désormais `figures.c`.
2. Traduire également cet algorithme en langage Python. Ce fichier s'appellera désormais `figures.py`.