



Problème I - Comptage

Ecrivez un programme qui compte le nombre d'occurrences d'une lettre dans un texte.

Exemples

Dans le texte "J'aime lire des livres français", la lettre 'e' figure 4 fois.

Dans le texte "Parfois je lis aussi un livre en anglais", la lettre 'y' figure 0 fois.

Entrées et sorties du programme

⇒ Le programme lit du clavier le texte, puis la lettre à rechercher.

⇐ Le programme affiche le nombre d'occurrences de la lettre à l'écran.

Restrictions

Le texte ne se compose que des lettres (minuscules et/ou majuscules) de l'alphabet français, des lettres accentuées de la langue française ainsi que de l'espace, du 'ç' et de l'apostrophe.

La longueur du texte est limitée à 100 caractères.

Remarques

Le programme doit faire la différence entre lettres minuscules et majuscules.

Prévoyez un message d'erreur si le texte contient des caractères non valides, p.ex. signes de ponctuation ou si la longueur du texte est supérieure à 100.

Prévoyez de même un message d'erreur si la lettre à rechercher n'est pas une lettre, mais p.ex. un chiffre.



Remettez le programme sous le nom COMPTAGE.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable COMPTAGE.EXE correspondant au programme.

Problème II - Suites

Ecrivez un programme qui génère des suites arithmétiques et géométriques.

Définitions

Une suite est une séquence (ici finie) de termes/facteurs entiers relatifs:

$$x_1; x_2; x_3; x_4; \dots; x_{n-1}; x_n \text{ avec } x_i \in \mathbb{Z} \text{ et } n \in \mathbb{N}^* (n \geq 2)$$

Une suite arithmétique est une suite qui vérifie $x_i = x_{i-1} + \text{constante}$ avec $i \in \mathbb{N}^*$, $2 \leq i \leq n$, $\text{constante} \in \mathbb{Z}$

Une suite géométrique est une suite qui vérifie $x_i = x_{i-1} \cdot \text{constante}$ avec $i \in \mathbb{N}^*$, $2 \leq i \leq n$, $\text{constante} \in \mathbb{Z}$

Exemples

0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14 est une suite arithmétique (constante = 1)

-17; -13; -9; -5; -1; 3; 7; 11; 15; 19; 23 est une suite arithmétique (constante = 4)

1; 2; 4; 8; 16; 32; 64; 128; 256; 512; 1024; 2048; 4096 est une suite géométrique (constante = 2)

1; -3; 9; -27; 81; -243; 729; -2187 est une suite géométrique (constante = -3)

Entrées et sorties du programme


- ⇒ Le programme lit du clavier le premier terme de la suite, la constante, le nombre de termes ainsi qu'une lettre indiquant s'il s'agit d'une suite arithmétique ou géométrique (prendre 'A' pour une suite arithmétique et 'G' pour une suite géométrique).
- ⇐ Le programme affiche les termes de la suite à l'écran de la gauche vers la droite. Les termes sont séparés par un espace.

Restrictions

Le premier terme de la suite, la constante ainsi que le nombre de termes sont des nombres entiers relatifs dont la valeur absolue est au plus 100.

Remarques

Prévoyez un message d'erreur si les valeurs saisies par l'utilisateur du programme sont fausses, p.ex. premier terme de la suite ou constante non entiers, nombre de termes négatif, erreur sur la lettre indiquant le type de la suite, etc. resp si les restrictions énoncées ci-dessus ne sont pas respectées.

-  Remettez le programme sous le nom SUITES.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable SUITES.EXE correspondant au programme.

Problème III - Détermination de Pi

Ecrivez un programme qui détermine une approximation de π par la formule de Bernouilli, qui converge vers la valeur exacte de π .

Formule de Bernouilli:
$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots$$

On obtient la valeur exacte de π si le nombre de termes dans la formule est infini. Si le nombre de termes dans la formule est fini, on n'aboutit qu'à une approximation de la valeur exacte de π .

Exemples

Avec 10 termes on obtient: = 3,049361636...

Avec 50 termes on obtient 3,122626523...

Avec 100 termes on obtient 3,132076532...

Avec 1.000 termes on obtient 3,140638056...

Avec 10.000 termes on obtient 3,141497164...

Avec 100.000 termes on obtient 3,141583104...

Entrées et sorties du programme

- ⇒ Le programme lit du clavier le nombre de termes.
- ⇐ Le programme affiche une approximation de la valeur de π à l'écran.


Restrictions

Le nombre de termes est inférieur ou égal à 1.000.000.

Nombre de places décimales derrière la virgule: au minimum 9.

Remarque

Prévoyez un message d'erreur si l'utilisateur désire plus que 1.000.000 de termes.

-  Remettez le programme sous le nom APPROXPI.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable APPROXPI.EXE correspondant au programme.

Problème IVa (au choix) - Conversion

Ecrivez un programme qui effectue une conversion du système de numérotation binaire vers le système hexadécimal et inversement.

Codage

Un nombre binaire (en base 2) est composé des 2 chiffres binaires (appelés *bits*) 0 et 1.

Un nombre hexadécimal (en base 16) est composé des 16 chiffres hexadécimaux 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

Exemples

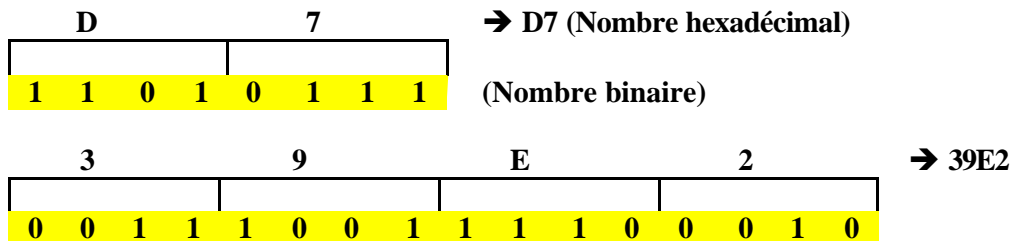
| | |
|-------------------------|---------------------------|
| Nombres binaires: 10100 | Nombres hexadécimaux: A34 |
| 11110000 | 4956 |
| 1010101000110011 | BC2F33 |
| 11 | FF |

Méthode de conversion

La conversion du système binaire vers le système hexadécimal et inversement peut se faire par la *méthode de conversion rapide*, qui consiste à faire correspondre des groupes de 4 bits à un chiffre hexadécimal. De cette façon on aboutit à la table de conversion suivante:

| | | | |
|----------|----------|----------|----------|
| 0000 ⇔ 0 | 0100 ⇔ 4 | 1000 ⇔ 8 | 1100 ⇔ C |
| 0001 ⇔ 1 | 0101 ⇔ 5 | 1001 ⇔ 9 | 1101 ⇔ D |
| 0010 ⇔ 2 | 0110 ⇔ 6 | 1010 ⇔ A | 1110 ⇔ E |
| 0011 ⇔ 3 | 0111 ⇔ 7 | 1011 ⇔ B | 1111 ⇔ F |

Exemples



Entrées et sorties du programme

- ⇒ Le programme lit du clavier un nombre binaire ou hexadécimal ainsi que le sens de la conversion (prendre 'H' pour une conversion du système binaire en système hexadécimal et 'B' pour l'inverse).
- ⇐ Le programme affiche le nombre converti à l'écran


Restrictions

On suppose que le nombre binaire à convertir se limite à 32 bits et que le nombre hexadécimal ne se compose qu'au maximum de 8 chiffres.

Remarques

Prévoyez un message d'erreur si les limites énoncées ci-dessus ne sont pas respectées respectivement si le nombre tapé par l'utilisateur est fautif, p.ex. A19G pour un nombre hexadécimal.

Prévoyez aussi un message d'erreur si la lettre indiquant le sens de la conversion est ni un 'B' ni un 'H'.

 Remettez le programme sous le nom CONVERT.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable CONVERT.EXE correspondant au programme.

Problème IVb (au choix) – Codage binaire

Ecrivez un programme qui effectue un codage d'un nombre décimal en système binaire.

Codage

Un nombre binaire (encore appelé un nombre en base 2) est composé des 2 chiffres binaires (appelés *bits*) 0 et 1. En général, les nombres binaires sont groupés par paquets de 8, 16, 32 ou 64 bits.

Exemples de nombres binaires

| | |
|----------------------------------|---|
| 10100100 | (8 bits, encore appelé un <i>byte</i>) |
| 11110000 | (8 bits) |
| 1010101000110011 | (16 bits) |
| 10000001100000001111111110101010 | (32 bits) |

Principe de la conversion du système décimal vers le système binaire – exemple

La principe consiste à diviser le nombre exprimé en système décimal par des puissances successives de 2.

$$(209)_{10} = (?)_2$$

La plus grande puissance de 2 qui est comprise dans $(209)_{10}$ est $128 = 2^7$

$$\begin{array}{l}
 (209)_{10} \rightarrow 209 : 2^7 = \mathbf{1} \text{ reste } 81 \\
 81 : 2^6 = \mathbf{1} \text{ reste } 17 \\
 17 : 2^5 = \mathbf{0} \text{ reste } 17 \\
 17 : 2^4 = \mathbf{1} \text{ reste } 1 \\
 1 : 2^3 = \mathbf{0} \text{ reste } 1 \\
 1 : 2^2 = \mathbf{0} \text{ reste } 1 \\
 1 : 2^1 = \mathbf{0} \text{ reste } 1 \\
 1 : 2^0 = \mathbf{1} \text{ reste } 0
 \end{array}
 \rightarrow (11010001)_2 \text{ ce qui correspond à un nombre binaire sur un byte.}$$

$$(26)_{10} = (?)_2$$

La plus grande puissance de 2 qui est comprise dans $(26)_{10}$ est $16 = 2^4$

$$\begin{array}{l}
 (26)_{10} \rightarrow 26 : 2^4 = \mathbf{1} \text{ reste } 10 \\
 10 : 2^3 = \mathbf{1} \text{ reste } 2 \\
 2 : 2^2 = \mathbf{0} \text{ reste } 2 \\
 2 : 2^1 = \mathbf{1} \text{ reste } 0 \\
 0 : 2^0 = \mathbf{0} \text{ reste } 0
 \end{array}
 \rightarrow (00011010)_2 \text{ ce qui correspond à un nombre binaire sur un byte.}$$

Entrées et sorties du programme


- ⇒ Le programme lit du clavier le nombre décimal à convertir.
- ⇐ Le programme affiche le nombre binaire correspondant à l'écran

Restrictions

Le nombre décimal à convertir se limite à $2^{32}-1=4.294.967.295$.

Remarques

Prévoyez un message d'erreur si la limite énoncée ci-dessus n'est pas respectée.

-  Remettez le programme sous le nom DEC2BIN.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable DEC2BIN.EXE correspondant au programme.