Université de Xidian – Polytech'Sophia 2017–2018

Langage C *(V. Granet)*

Durée : 2h00 Aucun document autorisé - Téléphones interdits

Nom chinois : **Nom Pinyin :**

Numéro étudiant : **Classe :**

▶ 1. On dispose de la fonction **long** `factorielle(int n)` qui renvoie $n!$. Sachant que l'entier le plus grand sur le type **long** est donné par la constante `LONG_MAX`, écrivez un programme qui détermine le plus grand entier `n` dont on peut calculer $n!$ sur le type **long**.

▶ 2. Le code de César est une des méthodes de cryptographie les plus anciennes. Il consiste en une substitution mono-alphabétique, où la substitution est définie par un décalage circulaire de lettres. Par exemple avec un décalage circulaire de 3 lettres, on remplace A par D, on remplace B par E, C par F, D par G, ... X par A, Y par B, Z par C.

Écrivez un programme C qui lit sur l'entrée standard un décalage (un entier, compris entre 1 et 25 - à vérifier) et une lettre (un caractère), et qui écrit la lettre chiffrée selon la méthode de César sur la sortie standard. Vous appliquerez cette transformation uniquement sur les lettres minuscules et majuscules, tout autre caractère restera inchangé.

Soit d'une fonction f continue sur un intervalle réel $[a, b]$, on veut trouver les x, appelés zéros, tels que $f(x) = 0$. Pour cela, nous utiliserons la méthode dichotomique suivante qui part du principe que si $f(a)f(b) < 0$ alors il existe au moins un zéro :

1. choisir $[a, b]$ qui contient a priori un zéro ;
2. choisir le milieu m de l'intervalle $[a, b]$;
3. si $f(m) = 0$ alors on a trouvé un zéro
4. si $f(a)f(m) < 0$ alors définir le nouvel intervalle $[a, m]$;
5. si $f(b)f(m) < 0$ alors définir le nouvel intervalle $[m, b]$;
6. recommencer à partir de 2.

▶ 3. Écrivez la fonction `zero` qui renvoie un zéro d'une fonction f selon l'algorithme itératif précédent. Un zéro sera trouvé à un epsilon près, à passer en paramètre. Enfin, pour garantir l'achèvement de l'énoncé itératif, le nombre maximal d'itérations sera également passé en paramètre.

On représente un entier non signé (**unsigned int**) sous forme binaire à l'aide un tableau d'entiers courts (**short**) contenant des 0 et des 1. Le bit de poids faible, 2^0, est à l'indice 0, et le bit de poids fort, $2^{\texttt{NB\_BITS}-1}$, est à l'indice `NB_BITS-1`. Les questions qui suivent sont liées.

▶ 4. À l'aide d'un `define`, définissez la constante `NB_BITS` égale au nombre d'éléments du tableau nécessaires à la représentation d'un entier non signé.

▶ 5. Écrivez la fonction `valeurDecimale` qui renvoie la valeur décimale d'un entier non signé donné à partir de sa représentation binaire contenue dans un tableau. Vous ne ferez aucune évaluation à la puissance. L'en-tête de cette fonction est le suivant :

```
unsigned int valeurDecimale(const short bits[])
```

▶ 6. Écrivez la méthode `decalageDroit` qui produit un décalage de n éléments vers la droite dans le tableau qui représente un entier non signé. Le décalage est ouvert, c'est-à-dire que des 0 sont réinjectés en début de tableau – i.e. à partir de l'indice 0. L'en-tête de cette fonction est le suivant :

```
void decalageDroit(short bits[], const int n)
```