

Exemple d'épreuve écrite d'Programmation Avancée

Documents interdits à l'exception d'une feuille d'aide-mémoire en format A4 manuscrite.

Téléphones et appareils électroniques interdits.

Le sujet comporte 2 exercices et 2 pages.

Durée : 1h30

1 Approximation de la racine carrée

L'objectif de cet exercice est de rechercher une valeur approchée de la racine carrée d'un nombre réel positif x ($x \geq 1$) à ε près à l'aide d'un algorithme dichotomique.

Pour rappel : la dichotomie ("couper en deux" en grec) est, en algorithmique, un processus itératif de recherche où, à chaque étape, on coupe en deux parties un espace de recherche qui devient restreint à l'une de ces deux parties. On suppose bien sûr qu'il existe un test relativement simple permettant à chaque étape de déterminer l'une des deux parties dans laquelle se trouve une solution.

1. Définir l'espace de recherche pour le problème de la recherche d'une racine carrée.
2. Quelle condition booléenne permet de savoir si il doit y avoir une nouvelle itération ?
3. Quel test va vous permettre de savoir dans laquelle des deux parties se trouve la solution ?
4. Proposez l'implémentation de la fonction suivante qui effectue une recherche dichotomique:

```
double racine_carree (double x, double epsilon);
```

2 Manipulation des grands nombres

Dans cet exercice, on s'intéresse aux calculs de grands nombres entiers (au-delà de la limitation de type int, long int ...). Pour cela, on utilise la structure de liste chaînée pour stocker les grands nombres et les manipuler. Un exemple est donné ci-après pour l'entier $n = 123456789123456789$, la liste chaînée est $[9, 8, 7, 6, 5, 4, 3, 2, 1, 9, 8, 7, 6, 5, 4, 3, 2, 1]$. Attention : l'entier est stocké en sens inverse dans la liste chaînée.



1. Proposez la structure de données de liste chaînée en C (avec **struct** et **typedef**) pour stocker les grands nombres.

Pour la suite de l'exercice, nous allons utiliser la structure définie dans la question 1 pour la manipulation des grands nombres. Pour simplifier, on suppose que les grands nombres traités sont des entiers positives.

2. Écrivez la fonction permettant d'ajouter un chiffre (donnée en paramètre) au début de la liste.
3. Écrivez la fonction permettant de compter la longueur d'une liste (en termes de nombre de chiffres). Par exemple, `longueur([1, 2, 3, 4])` retourne 4.
4. Écrivez la fonction permettant d'afficher le grand nombre stocké dans une liste. Par exemple : le nombre 4321 est stocké dans la liste sous forme `[1, 2, 3, 4]`, donc l'affichage est 4321 (mais pas 1234).
5. Écrivez la fonction permettant d'additionner deux listes de grands nombres et retourner la liste contenant la somme. Par exemple, `somme([3, 2, 8], [9, 8, 7, 9])` retourne `[2, 1, 6, 0, 1]` (ceci correspond à la somme de $823+9789=10612$).

On dispose d'une fonction `int toNumber(char c)` permettant de convertir un caractère `c` en valeur entière correspondante, -1 si le caractère n'est pas un chiffre. Par exemple, `toNumber('3')` retourne 3, `toNumber('a')` retourne -1.

6. Écrivez le programme qui :

- (a) Demande à l'utilisateur deux chaînes de caractères et puis convertit les chaînes en liste chaînée si la chaîne de caractères contient bien une séquence de chiffres, redemande la saisie jusqu'à ce que les entrées soient valides.
- (b) Appelle la fonction somme de la question 5 pour faire la somme de deux listes saisies précédemment, puis affiche le résultat.
- (c) On dispose maintenant d'un fichier, nommée `numbers.txt`, qui contient deux lignes donc chacune compose une séquence de chiffres correspondant à un grand nombre. Un exemple du fichier est donnée ci-après :

fichier <code>numbers.txt</code>
123456789012334445678990
9837403353939303

Ajoutez dans le programme la lecture du fichier et la conversion des deux grands nombres en liste chaînée. Puis, faites la somme des nombres en utilisant la fonction somme de la question 5 et écrivez le résultat à la fin du fichier `numbers.txt`. Pour l'exemple précédent, ça donne

fichier <code>numbers.txt</code>
123456789012334445678990
9837403353939303
123456798849737799618293

Fin de l'énoncé