Examen - ING1 GM
Programmation C

8 janvier 2019

Modalités

- Durée : 3 heures
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document papier n'est autorisé.
- Aucune sortie n'est autorisée avant une durée incompressible de 1 heure.
- Aucun déplacement n'est autorisé.
- **Aucune question au professeur n'est autorisée**. Si le sujet ne vous semble pas clair, à vous d'expliquer dans des commentaires pourquoi est ce que ça ne vous semble pas clair, et quelles modifications vous effectuez pour réaliser l'exercice.
- Aucun échange, de quelle que nature que ce soit, n'est possible.
- Les différentes fonctions/procédures demandées seront toutes à tester dans un **main**.
- Vous traiterez chaque exercice dans un dossier différent.
- Le barème est indicatif, il intègre pour chaque question un test dans un main.
- La présence d'un Makefile et des commentaires doxygens sont un plus, et seront appréciés.
- La lisibilité du code et les commentaires seront pris en compte dans la notation, de même que la séparation des différentes procédures/fonctions dans des fichiers d'en-tête (.c .h).
- **Tout code qui ne compile pas ne sera pas corrigé et entraînera une pénalité de 50%. Si une partie de votre code ne compile pas, commentez-le. Ceci implique évidemment que vous compiliez au fur et à mesure!**
- Chaque **warning** de compilation entraînera une **pénalité de 15%**.
- Tout ce qui sera dans le dossier rendu correspondra à votre rendu, et sera récupéré à la fin de l'examen. Je vous conseille vivement de travailler directement dans ce dossier.

Exercice 1 : Résolution équation du 3e degré (9 pts)

On considère une équation du troisième degré à coefficients réels $ax^3 + bx^2 + cx + d = 0$.
Afin de trouver les racines de l'équation, nous allons d'abord trouver une racine "évidente". Si l'on connaît déjà une solution d'une équation de degré 3, cela permet, pour trouver les autres, de se ramener à une équation de degré 2. En effet, si $p(x) = ax^3 + bx^2 + cx + d$ admet une solution x_0 alors il peut se factoriser sous la forme $p(x) = (x - x_0).q(x)$ avec $q(x)$ polynôme du second degré.
Les racines obtenues sont réelles et parfois complexes.

Trouver une racine "évidente" se fait en deux étapes :

1. Définir un intervalle dans lequel se situe la racine.
2. Par dichotomie, déterminer le zéro du polynôme.

Afin de déterminer le zéro du polynôme par dichotomie, le principe est le suivant. On considère deux nombres réels a et b et une fonction réelle f continue sur l'intervalle $[a, b]$ telle que $f(a)$ et $f(b)$ soient de signes opposés. D'après le théorème des valeurs intermédiaires, f a au moins un zéro dans l'intervalle $[a, b]$. La méthode de dichotomie consiste à diviser l'intervalle en deux en calculant $m = (a + b)/2$. Il y a maintenant deux possibilités : ou $f(a)$ et $f(m)$ sont de signes contraires, ou $f(m)$ et $f(b)$ sont de signes contraires. On rappelle le même principe sur l'intervalle restreint $[a, m]$ ou $[m, b]$ jusqu'à trouver le zéro de la fonction. Numériquement, il est fort possible qu'on ne trouve pas exactement 0, donc on acceptera une solution à 10^{-6} près.

Pour déterminer l'intervalle $[a, b]$ dans lequel se situe une racine, on commencera avec $a = -1$ et $b = 1$. Tant que $p(a)$ et $p(b)$ sont du même signe, on diminue a de 1 et on augmente b de 1.

1. Définir les structures suivantes :
    a. La structure `sComplexe` qui définit un nombre complexe par sa partie réelle et sa partie imaginaire.
    b. La structure `sSolution` qui regroupe l'ensemble des solutions possibles pour une équation du 3e degré.
    c. La structure `sEquation3` qui regroupe les coefficienrts a, b, c et d de l'équation du 3e degré.
2. Écrire la fonction `calculerPx(sEquation3 equation, double x)` qui renvoie la valeur de $p(x)$ de l'équation `equation` donnée au `x` donné.
3. Écrire la procédure `definirIntervalle(sEquation3 equation, double* a, double* b)` qui permet de trouver l'intervalle $[a, b]$ dans lequel se situe une racine de l'équation.
4. Écrire la fonction `zeroFonction(sEquation3 equation, double a, double b)` qui retourne une racine par la méthode de dichotomie présenté précédemment. On acceptera une solution à 10^{-6} près.

5. Écrire la fonction `polynomeQ(sEquation3 equation, double racine)` qui renvoie la nouvelle équation du 2nd degré $q(x)$.
6. Écrire la fonction `calculerSolution(sEquation3 equation)` qui renvoit les solutions de l'équation donnée.
7. Dans la fonction `main`, faire saisir les coefficients afin d'avoir une équation du 3e degré, calculer les racines de l'équation et les afficher.

Exercice 2 : Le tri fusion (11 pts)

Nous voulons réaliser un programme qui permet de trier un tableau d'entier par la méthode du tri fusion. Les étapes de la méthode sont les suivantes :

a. Si le tableau n'a qu'un élément, il est déjà trié.
b. Sinon, séparer le tableau en deux parties à peu près égales.
c. Trier récursivement les deux parties avec l'algorithme du tri fusion.
d. Fusionner les deux tableaux triés en un seul tableau trié.

Nous souhaitons récupérer les données depuis un fichier. Le fichier stocke l'ensemble des entiers à trier. Par exemple, le fichier "nombres.txt" contient :

```
3 1 6 42 65 9 32 47 10 35 24 11 7
```

Pour se faire, nous avons découpé le problème en plusieurs fonctions qui correspondront aux différentes questions.

1. Écrire la fonction `compterNombres` qui permet de compter le nombre de nombres présent dans le fichier dont le nom est donné en paramètre.
2. Écrire la fonction `allouerTableau` qui permet d'allouer un tableau d'entier d'une taille donnée.
3. Écrire la fonction `int* recupererTableau(char* nomFichier, int* tailleTableau)` qui permet de lire depuis un fichier l'ensemble des nombres et les stocker dans un tableau. La fonction renvoie le tableau initialisée, et met à jour la taille du tableau.
4. Écrire la procédure `afficherTableau` qui permet d'afficher un tableau.
5. Écrire la procédure `copierSousTableau` qui permet de copier dans un tableau *dest* les valeurs du tableau *src* allant de l'indice *debut* à l'indice *fin*.
6. Écrire la procédure `fusion` qui permet de fusionner deux tableaux triés de façon croissante *tab1* et tab2 dans un tableau résultat *tab* qui sera lui aussi trié de façon croissante.
7. Écrire la procédure `triFusion` qui trie un tableau de façon croissante. Cette procédure suivra les étapes du tri fusion décrites précédemment.
8. Écrire la fonction `main` qui permet :
    a. d'initialiser un tableau à partir d'un fichier dont le nom est donné en argument;
    b. d'afficher le tableau d'entiers non trié ;
    c. de trier le tableau d'entiers ;
    d. d'afficher le tableau d'entiers trié.