Université de Limoges
Faculté des Sciences
Master Procédés et Matériaux

Année 2005-2006

Travaux Pratiques - Langage C - Semestre 2
Série 1

1 Recommandations

Cette séance de travaux pratiques est d'une durée de trois heures. À la fin de la séances, vous rendrez un compte-rendu à l'enseignant.

Au début de chaque programme, vous noterez en commentaires :

- votre nom
- la date
- le nom de l'exécutable
- le but du programme

Veillez à commenter correctement votre programme. Rappelez-vous que les commentaires sont aussi importants que le code en lui-même…

Créez le répertoire `tp1` à la racine de votre compte utilisateur.

2 Manipulations de vecteurs avec la GSL

2.1 Adresses et types de données

Commencez tout d'abord par saisir le code ci-dessous dans le fichier `vecteur.c`

```
1  #include <stdio.h>
2  #include <gsl/gsl_vector.h>
3
4  int main(void)
5  {
6    int i;
7    gsl_vector *v;
8
9    /* initialisation du vecteur */
10   v = gsl_vector_calloc(10);
11   for(i=0;i<10;i++){
12     gsl_vector_set(v,i,i/10.0);}
13
14   /* libération de la mémoire */
15   gsl_vector_free(v);
```

```
16  }
```

Compilez le programme. Le binaire s'appellera `vecteur`. Exécutez-le et vérifiez son bon fonctionnement.

Modifiez le code du programme pour qu'il affiche d'une part l'adresse en mémoire du vecteur `v`, mais aussi, pour chaque élément du vecteur, l'adresse en mémoire et la valeur stockée, sous forme de colonnes. À l'exécution, votre programme devra ressembler à ceci :

```
Adresse du vecteur : B80004A0
Adresse    Valeur
-------    ------
804A030    v[0] = 0.00
804A038    v[1] = 0.10
804A040    v[2] = 0.20
804A048    v[3] = 0.30
804A050    v[4] = 0.40
804A058    v[5] = 0.50
804A060    v[6] = 0.60
804A068    v[7] = 0.70
804A070    v[8] = 0.80
804A078    v[9] = 0.90
```

Le spécificateur de conversion `0X` permet d'afficher les adresses en mémoire en base hexadécimale.

1. relancez plusieurs fois le programme. Les adresses réservées aux variables changent-elles ?

2. Ajoutez dans votre programme la déclaration d'un autre vecteur `u` avant `v` et affichez également son adresse en mémoire. Exécutez le programme et notez les adresses utilisées. L'adresse de la première variable déclarée dans le programme change-t-elle ?

3. Quel est le nombre d'octets qui sépare les adresses des éléments du vecteur ? Quel est le type de variable stocké dans le vecteur ? Est-ce cohérent ?

4. Le type de variable stocké dans le vecteur peut-être modifié. Remplacez partout dans votre programme `gsl_vector_` par `gsl_vector_float_`. Recompilez et exécutez le programme. Notez à nouveau le nombre d'octets séparant deux éléments du vecteur. Expliquez le résultat.

5. Remplacez maintenant vos `gsl_vector_float_` par des `gsl_vector_long_double_`. Compilez et exécutez le programme. Quel est maintenant le nombre d'octets séparant deux valeurs ? Est-ce cohérent ? Corrigez éventuellement votre code source pour que les valeurs de `v[i]` s'affichent correctement.

2.2 Normalisation d'une série de données

L'objectif du programme est de lire deux vecteurs contenus dans deux fichiers, d'additionner ces vecteurs, de déterminer les valeurs minimales et maximales (ainsi que leur rang) et enfin d'enregistrer la somme de ces deux vecteurs dans un fichier. Les valeurs écrites dans les fichiers auront deux chiffres après la virgule.

1. Commencez d'abord par regarder le contenu des deux fichiers `vecteur1.dat` et `vecteur2.dat` qui doivent normalement se trouver à la racine de votre compte. Déplacez-les dans le répertoire `tp2`Vérifiez avec la commande `ls -l` que vous avez le droit de lire ces fichiers, mais pas de les modifier. Vérifiez ensuite leur contenu avec la commande `cat`.

2. Dans votre programme, déclarez deux vecteurs `v1` et `v2`, contenant chacun 5 élements initialisés à zéro.

3. placez le contenu du fichier `vecteur1.dat` dans le vecteur `v1`

4. de même, placez le contenu du fichier `vecteur2.dat` dans le vecteur `v2`

5. Quelle fonction allez-vous utiliser pour additionner les deux vecteurs ? Où se trouve alors le vecteur résultant de l'addition ?

6. additionnez les deux vecteurs

7. enregistrez le vecteur contenant la somme dans le fichier `somme_non_normalisee.dat`

8. Depuis la console, vérifiez avec la commande `cat` le contenu du fichier `somme_non_normalisee.dat`

9. Affichez à l'écran l'indice et la valeur du minimum et du maximum du vecteur contenant la somme. Vérifiez si le résultat est cohérent.

10. Normalisez ce vecteur et enregistrez-le dans le fichier `somme_normalisee.dat`. Regardez le contenu de ce fichier depuis la console. Quelles sont les valeurs maximale et minimale ? Est-ce cohérent ?

2.3 Génération de la table du sinus

En vous aidant des programmes précédents, écrivez un programme qui génère un vecteur contenant les valeurs de $\sin(x)/x$ pour x allant de -10π à 10π. Le vecteur contiendra 1025 valeurs. Vous enregistrerez ce vecteur dans le fichier `sin.dat` au format ASCII.

3 Transformées de Fourier

Reprenez le code du programme correspondant au TD 4. Vérifiez le bon fonctionnement du programme avec le fichier pulse.dat qui vous est fourni.