Université de Limoges
Faculté des Sciences
Master Procédés et Matériaux

Année 2007-2008

Travaux Pratiques - Langage C

1 Recommandations

Cette séance de travaux pratiques est d'une durée de quatre heures. Au début de chaque programme, vous noterez en commentaire :

- 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…

2 Description de l'environnement

Vous utilisez l'environnement de bureau KDE. La barre située en bas de l'écran s'appelle le tableau de bord. Il contient, de gauche à droite :

- un menu K qui vous permet de lancer les applications
- des raccourcis vers les applications les plus souvent utilisées
- des bureaux virtuels. Tout se passe comme si vous aviez plusieurs écrans. Pour changer de bureau virtuel, il suffit de cliquer sur le numéro correspondant.
- une barre des tâches qui contient les icônes des fenêtres actuellement ouvertes
- de petites applications à droite (boîte à miniatures, horloge…)

3 Préparation de l'environnement de travail

Placez-vous sur le bureau n°1. Vous allez lancer un émulateur de terminal, qui va vous permettre d'interagir avec la machine en mode texte. Pour cela, appuyez simultanément sur Alt et F2. Une boîte de dialogue vous demande le nom de l'application que vous souhaitez exécuter. Saisissez **konsole** et validez par entrée ou cliquez sur Exécuter. C'est depuis ce terminal que vous compilerez vos programmes.

Passez maintenant sur le bureau n°2. Appuyez simultanément sur Alt et F2 et lancez l'éditeur de texte **kate**. Notez que nous aurions pu lancer konsole et kate depuis le menu K, ou encore depuis leur raccourci dans le tableau de bord s'il existe.

4 Manipulations élémentaires

Revenez dans konsole. À l'aide de la commande `pwd`, vérifiez dans quel répertoire vous vous trouvez actuellement. Vérifiez ce que contient ce répertoire avec la commande `ls`. Notez qu'un code de couleur vous permet d'identifier rapidement le type de fichier : blanc pour un fichier simple, bleu pour les répertoires, vert pour les fichiers exécutables, violet pour les images, et bleu clair pour les liens symboliques.

Créez un répertoire `tp1` à l'aide de la commande `mkdir`. Vérifiez avec `ls` que ce répertoire a bien été créé. Saisissez `ls -l` pour obtenir davantage d'informations sur les fichiers et les répertoires. Les tailles des fichiers sont données en octet, ce qui n'est pas très lisible pour les gros fichiers. Vous pouvez afficher les tailles en Ko, Mo et Go avec la commande `ls -lh` (h pour human readable). Allez dans le répertoire `tp1` à l'aide de la commande `cd`.

Retournez maintenant dans kate (bureau n°2) et saisissez le texte suivant dans l'éditeur :

```
La vie : "La vie c'est quelque chose de très fort et de très beau....
La vie appartient à tous les vivants. It's both a dream and a feeling.
C'est être ce que nous ne sommes pas sans le rester. La vie c'est mourir
aussi... Et mourir c'est vraiment strong... c'est rester en vie au delà
de la mort... Tous ceux qui sont morts n'ignorent pas de le savoir."
        -+- Jean-Claude VanDamme -+-
```

Enregistrez ce fichier sous le nom `vandamme.txt` dans le répertoire `tp1` que vous venez de créer. Pour cela, utilisez l'entrée de menu Fichier→Enregistrer. Vous pouvez également cliquer sur l'icône enregistrer de la barre d'outils, ou actionner le raccourci-clavier Ctrl+S.

Retournez maintenant dans le terminal et vérifiez avec `ls -lh` que le fichier que vous venez d'enregistrer est bien présent. Vérifiez également le nom de son propriétaire ainsi que sa date de création. Visualisez son contenu avec la commande `cat`. Renommez ce fichier en jeanclaude.txt et vérifiez avec `ls`. Pour terminer, supprimez ce fichier avec la commande `rm`.

5 Premier programme

Retournez dans kate et fermez le document en cours avec l'entrée de menu Fichier→Fermer. Saisissez alors le programme suivant dans l'éditeur :

```
#include <stdio.h>
main(){
printf("boujour tout le monde\n");
}
```

et enregistrez-le sous le nom `bonjour.c` dans le répertoire `tp1`. Remarquez que le fait d'enregistrer le fichier avec l'extension `.c` active automatiquement la coloration syntaxique associée au langage C.

Il est temps maintenant de compiler votre programme. Retournez dans le terminal. Assurez-vous d'être dans le répertoire `tp1` avec la commande `pwd`. Compilez votre programme avec `gcc bonjour.c`. Vérifiez que l'exécutable `a.out` a bien été créé. Exécutez votre programme en saisissant `./a.out`. Supprimez enfin `a.out`

Compilez de nouveau votre fichier source mais en indiquant cette fois à gcc que l'exécutable doit s'appeler `bonjour` au lieu de `a.out`. Exécutez-le.

6 Programme faisant appel à la librairie mathématique

Dans kate, créez un nouveau fichier texte (Fichier→Nouveau). Saisissez alors le programme suivant :

```
#include <stdio.h>
#include <math.h>

int main(void)
{
   /* initialisation des variables */

   int i=1;
   double somme=1.0;
   double epsilon=1e-2;
   double terme=1.0,coeff;
   double x=2;

   /* debut de la boucle de calcul */
```

TABLE 1 – Analyse du programme de calcul de $\exp(x)$

i	coeff	terme	somme
1			
2			
3			
4			
5			
6			
7			
8			

```
   do{
      coeff = x / i;
      terme = terme * coeff;
      somme += terme;
      i++;}
   while(terme>epsilon);

   /* affichage du résultat */

   printf("Valeur calculée par le programme : exp(2) = %e\n" , somme);
   printf("                 Valeur attendue : exp(2) = %e\n" , exp(2.0));

   /* tout s'est bien passé, on renvoie 0 au système d'exploitation */
   return 0;
}
```

Enregistrez ce code source dans le fichier exp.c. Comme vous vous en doutez, ce programme calcule l'exponentielle de 2 à l'aide d'un développement limité. En effet, on a :

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} \cdots + \frac{x^n}{n!}$$

Essayez de compiler votre programme. Vous devriez avoir une erreur de compilation du type :

```
/tmp/ccIp2Dqj.o(.text+0xa4): In function 'main':
: undefined reference to 'exp'
collect2: ld a retourné 1 code d'état d'exécution
```

Que s'est-il passé ? Quelle ligne de commande faut-il utiliser pour compiler de programme ?

Expliquez le fonctionnement de ce programme. Comment calcule-t-il la valeur de $\exp(2)$ sans faire appel à la fonction puissance ou à la fonction factorielle ? Pour vous aider, remplissez le tableau 1 pour chaque valeur de i.

Modifiez le programme précédent pour afficher à chaque itération les valeurs de `i, coeff, terme` et `somme`.

On souhaite maintenant optimiser les performances de ce programme. Pour cela, nous allons chronométrer le temps d'exécution du programme. Modifiez le programme précédent pour qu'il ressemble à ceci :

```
#include <stdio.h>
#include <math.h>
#include <sys/time.h>

int main(void)
{
   /* initialisation des variables */
   double somme=1.0;
   int i=1;
   double epsilon=1e-15;
   double terme=1.0,coeff;
   double x=2;

   /* variables pour le chronométrage */
   struct timeval tv1,tv2;
   struct timezone tz;
   long long diff;

   /* début du chromonétrage */
   gettimeofday(&tv1, &tz);

   /* debut de la boucle de calcul */
   do{
      coeff = x / i;
  terme = terme * coeff;
  somme += terme;
  printf("terme : %e\tcoeff : %e\tsomme : %lf\n",terme, coeff,somme);
      i++;}
   while(fabs(terme)>epsilon);

   /* fin du chronométrage */
   gettimeofday(&tv2, &tz);

   printf("Valeur calculée par le programme : exp(2) = %e\n" , somme);
   printf("                 Valeur attendue : exp(2) = %e\n" , exp(2.0));

   /* affichage du temps écoulé */
   diff=(tv2.tv_sec-tv1.tv_sec) * 1000000L + (tv2.tv_usec-tv1.tv_usec);
   printf("temps de calcul = %d usec\n",diff);

   /* tout s'est bien passé, on renvoie 0 au système d'exploitation */
   return 0;
}
```

Expliquez les modifications apportées au programme. Dans quelle unité est exprimé le temps de calcul affiché en fin de programme ? Vous vous aiderez pour cela de la section sur gettimeofday du manuel du programmeur Linux fournie en annexe.

Essayez ensuite d'optimiser la durée d'exécution de ce programme. De quel facteur arrivez-vous à diminuer le temps de calcul ? Quelles sont les instructions les plus gourmandes en temps ?

7 Programmes faisant appel à la GSL

7.1 localisation des fichiers

La GSL dispose de la commande `gsl-config` permettant de connaître l'emplacement des différents fichiers (fichiers d'inclusion et bibliothèques) ainsi que les options de ligne de commande à passer à gcc. Depuis le terminal, saisissez `gsl-config` et validez par entrée. Vous aurez alors la liste des options disponibles de cette commande. Déterminez alors :

- le numéro de version de la gsl
- le répertoire d'installation de la gsl
- les options de compilation à passer à gcc (sans cblas). D'après ce qui a été vu en cours, déduisez-en le nom et l'emplacement du fichier contenant la bibliothèque de la gsl. À l'aide de la commande `ls -lh`, déterminez la taille de la librairie.
- à votre avis, où se situent les fichiers d'inclusion (.h) de la gsl ?

Dans kate, ouvrez par exemple le fichier d'inclusion `gsl_const_mksa.h`. Que contient ce fichier ? Déduisez-en la valeur et l'unité de la constante de Plank.

7.2 Premier programme GSL

Dans kate, saisissez et complétez le code suivant et enregistrez-le sous le nom `distances.c` :

```
#include <stdio.h>
#include <>

int
main (void)
{
  /* célérité de la lumière */
  double c  =

  /* unité astronomique */
  double au =

  /* nombre de secondes dans une minute */
  double minutes =

  /* distance stored in meters */
  double r_earth = 1.00 * au;
  double r_mars  = 1.52 * au;

  double t_min, t_max;

  t_min = (r_mars - r_earth) / c;
  t_max = (r_mars + r_earth) / c;

  printf ("light travel time from Earth to Mars:\n");
  printf ("minimum = %.1f minutes\n", t_min / minutes);
  printf ("maximum = %.1f minutes\n", t_max / minutes);

  return 0;
}
```

Compilez le programme et exécutez-le.

7.3 Nombres complexes

Le code source sera enregistré sous le nom `complexe.c` et l'exécutable s'appellera `complexe`.

On définit le nombre complexe :

$$z = x + iy$$

Écrivez un programme qui :

- demande à l'utilisateur d'entrer x et y
- affiche le module de z
- affiche le module carré de z
- affiche l'argument de z
- affiche la valeur de 1/z

Vous utiliserez pour cela les fonctions relatives aux complexes de la librairie gsl. L'exécution du programme donnera quelque chose ressemblant à ceci :

```
z = x + i y
x = 2
y = 3
------------
|z|    = 3.606
|z|2  = 13.000
arg(z) = 0.983
1/z    = 0.154 + i * -0.231
```

8 Pour les plus rapides

8.1 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 ?

8.2 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.

8.3 Transformées de Fourier

Reprenez le code du programme correspondant au TD 9. Vérifiez le bon fonctionnement du programme avec le fichier pulse.dat qui vous est fourni.