CHAPITRE

TP 4

Algo sympa et bibliothèques

Numéro 1 : Le carré du cercle donne π ..

On reprend l'exercice optionnel du TP3 et la génération de nombres aléatoires.
On imagine une boucle itérative avec un compteur i courant de 1 à N.
Soit pour chaque valeur de i un couple de nombres aléatoires (x, y) chacun compris entre $[0, 1]$. Programmer un algorithme tel que si la norme $\sqrt{x^2 + y^2} < 1$ alors un second compteur que l'on nommera $cible$ est incrémenté de +1. (Il vaut 0 au départ.)
Montrez que le rapport $cible/N$ tend vers le nombre $\pi/4$ lorsque N devient suffisamment grand. Vous accomplirez ceci au moyen d'un code C complet en expliquant votre algorithme.

Numéro 2 : Bibliothèque math.h

Cet exercice vous amène à utiliser une bibliothèque importante, *math.h* où sont définies la plupart des fonctions mathématiques commune, par exemple la fonction *sqrt* qui extrait la racine carrée d'un flottant. Les fonctions trigonométriques sont elles aussi définies dans math.h.

– Localisez sur votre ordinateur le fichier *math.h* en utlisant la commande Unix *find*. Exemple : *find /usr -name math.h -print*.
– Le nombre irrationnel π est (normalement) conservé dans le fichier math.h comme une constante symbolique, par exemple


```
# define M_PIl 3.14159265358979323846264338327950291 /* pi */
```

en double précision. Retrouvez cette définition dans le fichier math.h.
– Développer un algorithme pour calculer le sinus d'un angle θ prenant 10 valeurs comprises dans l'intervalle $[0, \pi/2]$ et qui affichera à l'écran la valeur de l'angle et son sinus.
– Faites l'implantation de cet algorithme au moyen d'une boucle FOR. Le fichier source s'appellera *sinus.c*. A la compilation, ajoutez l'option -lm au compilateur gcc. Que se passe-t-il si vous omettez cette option ? (Réponse : Le lien à la bibliothèque math.h n'est pas effectué.)

Reprenez votre fichier source *monexpo.c* pour y inclure maintenant la bibliothèque *math.h*. Cette librairie contient une fonction appelée *exp()* prenant un flottant comme argument.

– Recompilez *monexpo.c* avec la nouvelle librairie et en n'oubliant pas l'option -lm du compilateur.
– Lorsque tout compile, faites un appel à la fonction *exp()* avec le même argument que votre polynome de degré 10. Comparez vos résultats. Quelles valeurs de x (comme argument) vous permettent de maintenir une précision de 10% ?

Numéro 3 : Tri d'une suite de nombres

Développer un programme C faisant l'implantation d'un algorithme faisant le tri (partiel ou complet) d'une suite d'entiers par ordre croissant.
Vous ferez deux versions de cet algorithme :

– une premiŕe version où seulement deux éléments d'un tableau de nombres seront comparés puis interchangé si le deuxième élément est supérieur au premier. Exemple : si les indices i = 1 et j = 2 et si le tableau *tab* est tel que tab[i] $>$ tab[j], alors il faut faire en sorte que les valeurs numériques change de place dans le tableau.
– la deuxième version cherchera a appliquer cette idée à l'ensemble des éléments du tableau, pour que celui-ci se retrouve trié en ordre croissant au final.

Le tableau est à déclarer et initialiser par vous.
Il sera pratique de visualiser à l'écran chaque étape où des entiers sont échangés dans le tableau. Se référer aux notres de cours pour l'utilisation de la fonction *printf* ou encore consulter les exemples de programmes au site internet du cours.

Numéro 4 : Nombres complexes

Le but de cet exercice est de découvrir une application du type composé STRUCT afin d'opérer une multiplication entre deux nombres complexes (avec nombre imaginaire i) après saisie au clavier.

Familiarisez-vous tout d'abord avec le programme suivant que vous devrez modifier :

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

/* prototype, declaration d'une fonction .. */

void fonc ( void ) ;

/*  struct est un mot reserve du C - declaration d'une structure.
    il s'agit donc d'un nouveau type de variables,  un type composes
    et personnalise (perso),  selon mes besoins :  */

struct complexe {   double reel ; double imaginaire ;  };

/* Utilisation de typedef: voir support de cours */

typedef struct complexe Complexe ; // Noter la lettre majuscule!
typedef int Entier ;

/* Debut fonction main */

int main()
{
 struct complexe z1, z2;
 Complexe z3[4];             /* TABLEAU de complexes */

 z1.reel = 1.L;
 z1.imaginaire = 2.L ;
 z2 = z1 ; z3[1] = z2 ;

printf("\n") ;
printf( "Elements of z1 %g %g \n", z1.reel, z1.imaginaire );
printf( "Elements of z2 %g %g \n", z2.reel, z2.imaginaire );
printf( "Elements of z3? %f %f\n",z3[1].reel,z3[1].imaginaire);
printf("\n") ;

   fonc() ;

 return 0 ;
```



```
}

void fonc( void )
{ /* Autre maniere d'initaliser un 'complexe'*/
  struct complexe toto = {3.0, 1.0 };

  printf( "Dans fonc, toto = ( %f, %f ) \n", toto.reel,
           toto.imaginaire );
return ;
}
```

– Modifez ce programme afin de saisir chacun des champs de deux nombres complexes, effectuer l'opération de multiplication par une fonction, et retourner le résultat à l'écran.