TD2 – Bases du C

Partie TD

Exercice 1 – [Calcul de la surface d'un disque]

1. Écrire un programme comportant la déclaration d'une variable `pi` initialisée avec une (bonne) valeur approchée de π.
2. Ajouter la déclaration d'une variable flottante dont l'identificateur sera `rayon`.
3. Ajouter les instructions qui permettent d'initialiser la variable `rayon` avec la valeur 10, puis calculer la surface du disque en stockant le résultat dans une variable dont l'identificateur sera `resultat`.
4. Enfin, afficher le contenu de la variable `resultat` à l'écran.

Exercice 2 – [Échange du contenu de deux variables]

1. Écrire un programme C dans lequel sont (déclarées et) initialisées deux variables entières `a` et `b`.
2. Écrire les instructions permettant d'échanger le contenu des variables (le contenu de `a` doit être remplacé par celui de `b` et inversement).
3. Même question sans utiliser de variable auxiliaire.

Exercice 4 – [Valeur absolue]

Écrire un programme permettant de calculer, avec un test **`if`**, la valeur absolue d'un nombre réel x dont la valeur est saisie par l'utilisateur.

Exercice 5 – [Tests de conformité d'un colis]

Un colis est conforme si ses trois dimensions ($L \times l \cdot h$, avec $h \leq l \leq L$) vérifient les conditions suivantes :

1. un rectangle de 10 cm $\times$ 7 cm doit pouvoir s'inscrire dans une face du colis
2. $L + l + h \leq 100$ cm

3. $L \leq 60$ cm

Ecrire un programme `C` qui :

- demande à l'utilisateur de saisir les dimensions du colis en centimètres;
- vérifie que la saisie est correcte (à savoir $h \leq l \leq L$) ;
- vérifie si le colis est conforme ou non ;
- et affiche les messages appropriés.

Exercice 6 – [Première programmation scientifique en C]

Soit l'équation du second degré à coefficients réels à résoudre dans $\mathbb{C}$:

$$ax^2 + bx + c = 0$$

avec a, b, c dans $\mathbb{R}$.

1. Identifier en fonction des valeurs de a, b et c les six cas qui demandent un traitement différent.

2. Ecrire le programme C permettant de résoudre une telle équation, dont les paramètres seront entrés par l'utilisateur. On précise que l'expression `fabs(X)`, où `X` est une variable réelle, retourne la valeur absolue (réelle) de `X`. De même, l'expression `sqrt(X)`, où `X` est une variable réelle, retourne la racine carrée (réelle) de `X`.

Un corrigé est disponible en annexe.

Partie TME

Un programme est dit :

- *correct* s'il répond aux exigences du problème initial ;

- *robuste* s'il réagit correctement en présence de données erronées ;
- *fiable* s'il est correct *et* robuste ;
- *commenté* si les commentaires aident le lecteur à comprendre ce que fait le programme;
- *clair* s'il contient des commentaires et des identificateurs significatifs.

La compilation permet de traduire un programme écrit dans un éditeur de texte en instructions machines compréhensibles par l'ordinateur. Cette écriture doit être soignée pour facilement corriger des erreurs ou pour modifier ultérieurement le programme. On ne saurait trop vous recommander les règles suivantes :

- mettre des commentaires (courts et clairs de préférence) ;
- choisir des noms de variables explicites ;
- indenter le programme

> *Sous `gedit`, on peut indenter le programme ligne par ligne à l'aide de la touche `tab` (tabulation). Le nombre d'espaces utilisés par tabulation est réglable dans le menu Édition > Préférences > Éditeur (choisir au moins `2` espaces par tabulation).*

La commande `gcc` invoque le compilateur C dans l'environnement Linux. Le fichier exécutable produit s'appelle alors par défaut `a.out`. L'option -o permet de spécifier un nom différent pour ce fichier exécutable (indiqué par le mot qui suit immédiatement -o). Si le programme source est contenu dans le fichier `prog.c`, et qu'on souhaite appeler le fichier exécutable `prog`, il suffit donc de taper la commande suivante pour la compilation :

```
gcc -o prog prog.c
```

puis la commande suivante pour exécuter `prog` :

```
./prog
```

Exercice 7

Voici un programme sensé répondre à l'exercice 3 :

```c
#include <stdio.h>

int main () {
    const float a=0, b=0 ;

    printf("Entrez le premier reel\n");
    scanf(&a);
    printf("a = \n", a);

    printf("Entrez le second reel\n");
    scanf(&b) ;
    printf(" b = \n", b) ;

    somme=a+b ;
    printf("a + b= %f\n", somme) ;

    if (somme >= 0)
        printf('Somme negative\n');
    else
        printf('Somme positive (ou nulle)\n');
    return 0;
}
```

Compiler ce programme tel quel, et pour chaque erreur, utiliser les messages d'erreurs du compilateur pour détecter son numéro de ligne dans le code, pour obtenir des renseignements sur sa nature et pour la corriger. Une fois que le programme compile sans erreur, vérifier à l'aide d'un jeu de test exhaustif que le programme fournit toujours un résultat correct.

Exercice 8 – [Division réelle et division entière]

1. Ecrire un programme qui affiche le résultat des opérations suivantes :

   ```
   3./4.
   3./4
   3/4.
   3/4
   ```

   Pourquoi obtient-on de tels résultats ?

Exercice 9 – [Conversions entiers réels]

Essayer de prédire l'affichage effectué par le programme suivant. Puis vérifier vos prédictions en écrivant, compilant et testant ce programme.

```c
#include <stdio.h>   // librairie standard entrees/sortie
#include <math.h>    // pour nearbyint

int main() {

    float r, r1, r2 ;
    int i, i1, i2  ;

    i = 14 ; r = i ;
    printf("i = %d, r = %f\n", i,r) ;
    i = 1999999999 ; r = i ;
    printf("i = %d , r = %f\n", i,r) ;

    i = 0.999 ;
    printf ("i = %d\n", i) ;

```

```
17      r = 7/2   ;
18      printf("r = %f\n", r );
19      r = 7./2. ;
20      printf("r = %f\n", r );
21      i = 3/4   ;
22      printf("i = %d\n", i ) ;
23      i = 3./4. ;
24      printf("i = %d\n", i ) ;
25
26      r1=7. ;
27      r2=2. ;
28      printf("r1/r2 = %f\n", r1/r2);
29
30      r1=7  ;
31      r2=2  ;
32      printf("r1/r2 = %f\n", r1/r2);
33
34      i1=7  ;
35      i2=2  ;
36      printf("i1/i2 = %f\n", i1/i2);
37
38      i1=7. ;
39      i2=2. ;
40      printf("i1/i2 = %d\n", i1/i2);
41
42      r1=7. ;
43      i2=2  ;
44      printf("r1/i2 = %f\n", r1/i2);
45
46      i = 3.14  ;
```

```
47      printf("i = %d\n", i) ;
48
49      i = -3.14 ;
50      printf("i = %d\n", i) ;
51
52      printf("floor(3.14) = %f\n", floor(3.14));
53      printf("floor(-3.14) = %f\n", floor(-3.14));
54      printf("ceil(3.14) = %f\n", ceil(3.14));
55      printf("ceil(-3.14) = %f\n", ceil(-3.14));
56      printf("nearbyint(3.81) = %f\n", nearbyint(3.81));
57      printf("nearbyint(-3.81) = %f\n", nearbyint(-3.81));
58      return 0;
59  }
```

Note : certains calculs nécessitent des fonctions de la bibliothèque mathématique. Pour l'utilisez, vous devez écrire `#include <math.h>` au début de votre code et ajouter l'option `-lm` sur votre commande de compilation (`gcc -lm ...` dans le terminal).

Exercice 10

Ecrire et tester les programmes C écrits lors des exercice 4 et 6.

Voir solutions dans la partie TD.

Exercice 11 – [(pour aller plus loin...)]

Ecrire un programme C permettant de résoudre un système de 2 équations linéaires à 2 inconnues.

Pour déterminer les solutions, on peut utiliser la **Règle de Cramer** :

Soit le système linéaire

$$\begin{cases} ax + by = e \\ cx + dy = f \end{cases}$$

où a, b, c, d, e et f sont des constantes connues.

Si l'on a $ad - bc \neq 0$, alors les solutions peuvent s'écrire :

$$x = \frac{ed - fb}{ad - bc} \quad \text{et} \quad y = \frac{af - ec}{ad - bc}.$$

La correction est disponible en annexe.

Annexe : corrections

Correction de l'exercice 6

Exemple de code :

```c
#include <stdio.h>   // librairie standard entrees/sortie
#include <math.h>    // fournit (entre autres) sqrt : racine carree

int main () {
    float a,b,c,delta,x1,x2,w1,w2 ;
    printf("Resolution de ax**2+bx+c\n");
    printf( "Saisir a, b et c:");
    scanf("%f %f %f",&a,&b,&c);
    printf("Equation : %f*x**2 + %f*x + %f\n", a,b,c);
```

```
    if (a==0.0) {
        if (b==0.0) {
            if (c==0.0) printf("Equation indeterminee\n");
            else printf("Equation impossible\n") ;
        }
        else printf("Equation de premier degre x=%f\n",-c/b);
    }
    else {
        delta=b*b-4*a*c ;
        w1=-b/2./a ;
        w2=sqrt(fabs(delta))/2./a ;
        if (delta > 0.0) {
            x1=w1+w2 ;
            x2=w1-w2;
            printf("2 racines reelles distinctes x1 = %f, x2 = %f\n", x1, x2);
        }
        else if(delta <0.0) {
            printf("2 racines complexes conjuguees\n");
            printf("partie reelle= %f, partie immaginaire = %f\n",w1,w2);
        }
    else printf("1 racine reelle double = %f\n",w1);
    }
}
```

Correction de l'exercice 11

Le code C qui vous est donné ci-dessous effectue le calcul des déterminants qui interviennent dans la règle de Cramer puis la résolution :

```c
#include <stdio.h>   // librairie standard entrees/sortie
#include <math.h>    // fournit (entre autres) fabs : valeur absolue
int main () {
    float    a, b, c, d, e, f;
    float delta;
    float x,y;

    /* Resolution de :
       a.x + b.y = e
       c.x + d.y = f */

    // Initialisation des coefficients.
    a = 2; c = 4; b = 5; d = 11;  e = 7; f = 6 ;

    // Calcul du determinant principal.
    delta = a*d - c*b ;
    if ( fabs(delta) < 1e-6 ) { // le determinant est nul
        printf("Le systeme n'a pas de solution unique.\n");
        return 1 ;      // sortie du programme
    }

    // calcul des solutions.
    x = (e*d - f*b)/delta ;
    y = (a*f - c*e)/delta ;

    // Affichage des solutions.
    printf("x = %f, y = %f\n ",x, y);
    return 1;
}
```

Remarque : Dans le code ci-dessus, `1e-6` correspond à la notation scientifique 1×10^{-6}. À cause des erreurs d'arrondi, pour savoir si un nombre réel vaut zéro, on ne teste pas si sa valeur est égale à '0', mais si sa valeur absolue est suffisament petite...