Programmation Impérative: TD n°6
Structure des programmes

Exercice 1 *
Écrire un programme qui permet de saisir un nombre complexe.
Le modifier pour qu'il affiche le module au carré du nombre complexe saisi.

Exercice 2 ** ***if imbriqués***
Écrire un programme qui:

- *Demande à l'utilisateur de rentrer trois nombres a, b, c.*
- *Détermine les points d'intersections de la droite d'équation $ax + by + c = 0$ avec les axes des abscisses et des ordonnées.*
- *Affiche le resultat.*

Exercice 3 * ***Paradigme: demander puis lire***
En programmation, on ne repart pas toujours de zéro. Il y a des choses qui reviennent souvent, qu'il faut donc savoir faire par cœur: ce sont les paradigme de programmation.
Vous êtes déjà capable d'implémenter deux paradigmes:

- *Le paradigme "demander puis lire"*
- *Le paradigme "entrée - traitement - sortie"*

Pour chacun de ces paradigme, trouvez un petit problème très simple qui l'utilise. Réalisez le programme qui correspond.

Exercice 4 *** ***Constantes - priorité - portée***
Les constantes servent à stocker des valeurs qui ne changent pas durant l'exécution du programme. Elles peuvent servir à présenter des résultats différents selon qu'on est en train de tester un programme ou de l'utiliser réellement.
Écrire un programme permettant d'afficher les racines (si elles existent) d'un polynôme du second degré.
Modifier ce programme pour qu'il affiche la valeur du déterminant.
Cette valeur n'est utile que lors de la phase de test. Modifier ce programme pour qu'il ne l'affiche que si la constante `DEBUG` *vaut* 1.
Note: Pour calculer une racine carrée, il faut utiliser la fonction `sqrt`*. Pour cela, ajouter la ligne* `#include "math.h"` *au début du programme, puis utiliser la fonction sous la forme* `x = sqrt(y)`*.*

Exercice 5 ** ***if imbriqués***
Pour calculer les coordonnées polaires (ρ, θ) d'un point à partir de ses coordonnées cartésiennes (x, y), on utilise les règles suivantes:
si $x = 0$ alors $\rho = |y|$ et $\theta = \pi/2$ si $y > 0$ et $-\pi/2$ sinon
sinon:

- *$\rho = \sqrt{x^2 + y^2}$*
- *si $y > 0$ alors $\theta = atan(y/x)$*
- *sinon $\theta = -atan(y/x)$*

Écrire un programme permettant de réaliser cette conversion.

1 Correction

Dans les exercices, les programmes se terminent pas un `return 0PV`. Pour information, cette instruction renvoie le code de l'erreur (`0`= pas d'erreur). Pour les étudiants, on peut omettre cette instruction pour l'instant.

Exercice 1 La première question ne concerne que la saisie, la deuxième la saisie, le traitement et l'affichage.

```
#include <stdio.h>

int main() {
  float re,im;
  printf("Entrez la partie réelle:");
  scanf("%f",&re);
  printf("Entrez la partie imaginaire:");
  scanf("%f",&im);
  return 0;
}
```

```
#include <stdio.h>

int main() {
  float re,im;
  float module2;

  /* saisie du nombre complexe */
  printf("Entrez la partie réelle:");
  scanf("%f",&re);
  printf("Entrez la partie imaginaire:");
  scanf("%f",&im);

  /* calcul du module */
  module2=re*re+im*im;

  /* affichage du module au carré */
  printf("Le module au carre vaut %f.\n",module2);
  return 0;
}
```

Exercice 2 Solution:

```
# include <stdio.h>

int main() {
  float a,b,c;
  float solx,soly;

  /* Lecture des parametres decrivant la droite */
  printf("Pour l'equation de droite ax+by+c=0, entrer la valeur de a:\n");
  scanf("%f",&a);
  printf("entrer la valeur de b:\n");
  scanf("%f",&b);
  printf("entrer la valeur de c:\n");
  scanf("%f",&c);

  /* Test pour savoir si ax+by+c=0 definit bien une droite */
  if(a==0 && b==0) {
    printf("a et b ne peuvent etre tous les deux nuls!\n");
    return(1);
  } else if (a==0) { /* cas d'une droite horizontale */
      soly=-c/b;
      printf("la droite %f x + %f y + %f = 0 intersecte seulement l'axe des ordonnees en : y=%f\n",a,b,c,soly);
  } else if (b==0) {
    solx=-c/a;
```

```
    printf("la droite %f x + %f y + %f = 0 intersecte seulement l'axe des abscisses en : x=%f\n",a,b,c,solx);
  } else {
    solx=-c/a;
    soly=-c/b;
    printf("la droite %f x + %f y + %f = 0 intersecte l'axe des abscisses en : x=%f et l'axe des ordonnees en : y=%f\n",a,b,c,solx,s
  }
  return(0);
}
```

Exercice 3 Certains préconisent d'apprendre l'informatique en se basant uniquement sur l'étude des paradigmes: décortiquer des codes existants, puis les modifier, pour enfin en construire soit-même. Sans vouloir aller aussi loin, je pense qu'il est bon que les étudiants soient conscient de ces recettes. D'abord, cela leur permet d'écrire des programmes plus rapidement, en utilisant ces paradigmes comme des meta-instructions, des grosses instructions qui font plein de choses. Ensuite, et ce n'est pas moins intéressant, cela leur permet de mesurer l'évolution de leurs acquis.
Le paradigme "demander puis lire" (voir l'exercice 1, question 1):

```
printf("Texte d'invite a entrer une valeur:");
scanf("%format",&variable du bon type);
```

Le paradigme "entrée - traitement - sortie" (voir l'exercice 1, question 2):

```
printf("Texte d'invite a entrer une valeur:");
scanf("%format",&variable du bon type);
/* traitement des parametres */
printf("Texte presentant les parametres et les resultats %formats",var1,var2);
```

Exercice 4 La difficulté vient du fait que cet exercice nécessite la maîtrise de beaucoup de notions.

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

#define DEBUG 1

int main() {
  double a,b,c;
  double determinant;

  /* Invite α entrer les valeurs */
  printf("Resolution d'une equation du type ax^2+bx+c=0\n");
  printf("Donner la valeur de a:");
  scanf("%lf",&a);
  printf("Donner la valeur de b:");
  scanf("%lf",&b);
  printf("Donner la valeur de c:");
  scanf("%lf",&c);

  /* Résolution de l'équation et affichage des résultats */
  determinant = b*b-4*a*c;
  printf("L'equation %lfx^2+%lfx+%lf ",a,b,c);
  if (DEBUG == 0) {
    printf("\nDeterminant = %lf\n",determinant);
  }
  if (determinant < 0) { /* pas de racine relle */
    printf("ne possede pas de racine reelle.\n");
  } else {
    if (determinant == 0) { /* une seule racine */
      double sol;
      sol = b/(2*a);
      printf("possede une seule racine reelle : %lf.\n",sol);
    } else { /* det > 0: deux racines reelles */
      double sol1, sol2;
      double rac_det;

      rac_det = sqrt(determinant);
      sol1 = (b-rac_det)/(2*a);
      sol2 = (b+rac_det)/(2*a);
      printf("possede deux racines reelles : %lf et %lf.\n",sol1,sol2);
```

```
    }
  }
  return 0;
}
```

Exercice 5 Notez que lorsqu'il y a plusieurs instructions, on utilise des accolades ({...}) pour délimiter le bloc. Par contre, s'il n'y a qu'une seule instruction, pas d'accolade.

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

int main() {
  float x,y;
  float rho,theta;

  /* Saisie des coordonnees cartesiennes */
  printf("Coordonnée cartesiennes:\n");
  printf("Coordonnée x:");
  scanf("%f",&x);
  printf("Coordonnée y:");
  scanf("%f",&y);

  /* Calcul des coordonnees polaires */
  if (x == 0) { /* imaginaire pur */
    rho = abs(y);
    if (y > 0)
      theta = M_PI/2; /* vers le haut */
    else
      theta = -M_PI/2; /* vers le bas */
  } else {
    rho = sqrt(x*x+y*y);
    if (y > 0)
      theta = atan(y/x); /* vers le haut */
    else
      theta = -atan(y/x); /* vers le bas */
  }

  /* Affichage des coordonnees polaires */
  printf("Coordonnée polaires:\n");
  printf("rho=%f\n",rho);
  printf("theta=%f\n",theta);

  return 0;
}
```