Examen partiel - semestre 1
Initiation à la programmation – novembre 2017

Électronique et Physique appliquée

Tous documents non électroniques autorisés
Durée 1h

Nom, prénom :

- **Répondre sur le document.** Utilisez le dos de la feuille si vous manquez de place.
- Écrire vos programmes en langage C en respectant la syntaxe et l'indentation usuelle ;
- Soignez votre présentation (des points pourront être enlevés dans le cas contraire) ;
- Les questions sont relativement indépendantes ;
- Chaque question est notée sur 3 points.

1. Analyse de code

Qu'affiche le programme ci-dessous ?

```
#include<stdio.h>
int main (void) {
    int a = 4/3 ;
    float b = 4/3 ;
    float c = 4/3.0 ;
    printf ("a = %d, b = %f, c = %f\n", a, b, c) ;
    int d = 32%20 ;
    int e = 4, f = 18;
    int g = (e = f );
    printf ("d = %d, e = %d, f = %d, f = %x, g = %d\n", d, e, f, f, g) ;
    char h ='A' ;  // pour info, le code ASCII de 'A' est 65
    h++ ;
    int i = (3*4==12) ? 8 : 9 ;
    printf ("h = %c, h= %d, i=%d (3*4==12) = %d \n", h, h, i, (3*4==12)) ;
}
```

```
a = 1, b = 1.000000, c = 1.333333
d = 12, e = 18, f = 18, f = 12, g = 18
h = B, h= 66, i=8 (3*4==12) = 1
```

2. Programmation : fonctions

On désire écrire le code de la fonction pgcd(…) qui calcule le Plus Grand Commun Diviseur de 2 nombres entiers. LE PGCD sera calculé par l'algorithme d'Euclide suivant.

PGCD (a,0) = a
Si b != 0, PGCD (a,b) = PGCD (b, a % b)

Voici le *main* permettant d'utiliser la fonction :

```
#include<stdio.h>
int main (void) {
    int a ,b ;
    printf ("Entrez 2 nombres dont vous voulez déterminer le PGCD : ") ;
    scanf ("%d %d", &a, &b) ;
    printf ("PGCD (%d, %d) = %d\n", a, b, pgcd(a,b)) ;
}
```

Et une trace d'exécution :

```
Entrez 2 nombres dont vous voulez déterminer le PGCD : 36 60
PGCD (36, 60) = 12
```

1) **Écrire** la version récursive de la fonction PGCD

```
int pgcdRecur (int a, int b) {
     if (b ==0)  return (a) ;
     else return (pgcdRecur(b,a%b));
}
```

2) Écrire maintenant le code de la fonction *PGCD* de manière **non** récursive.

```
int pgcd (int a, int b) {
     int r ;
     if (b ==0)  return (a) ;
     do {
          r=a%b ;
          a=b ;
          b=r ;
     } while(r!=0) ;
     return (a) ;
}
```

3. fonctions – passage par adresse

On désire écrire le code de la fonction cartesienne(..) qui calcule les coordonnées cartésiennes d'un point en fonction de ses coordonnées polaires.

*x = ro * cos(theta)*
*y = ro *sin (theta)*

Le prototype de la fonction sera le suivant

```
void cartesienne (double rho, double theta, double *x, double *y)
```

Les fonctions cos et sin calculent respectivement le cosinus et le sinus d'un angle exprimé en radian.
Le paramètre *theta* de la fonction cartesienne() sera exprimé en degré.
On rappelle que la constante *M_PI* en C est la valeur de π. au format double.

1) **Écrire** le code de la fonction cartesienne (…)

```
void cartesienne (double rho, double theta, double *x, double *y) {
      *x = rho * cos(theta/180*M_PI) ;
      *y = rho *sin (theta/180*M_PI) ;
}
```

2) **Écrire** le code du main() appelant la fonction cartésienne et dont la trace d'exécution serait la suivante :

```
Entrez les coordonnées polaires d'un point : rho theta : 4 60
x= 3.464102, y=2.000000
```

```
int main (void) {
      double rho, theta, x, y ;
      printf ("Entrez les coordonnées polaires d'un point : rho theta : ") ;
      scanf ("%lf %lf", &rho, &theta) ;
      cartesienne (rho, theta, &x, &y) ;
      printf ("x= %lf, y=%lf\n", x, y) ;
}
```

4. Aléatoire

Écrire un programme principal qui tire aléatoirement N=10 nombres compris entre 0 et 99, et affiche la moyenne *m* et la variance σ^2 de cette série. On rappelle que la définition de la variance : $\sigma^2 = 1/N * \Sigma (x-m)^2$
Remarque, le programme devra très certainement utiliser un tableau.

```
int main (void) {
      float m=0, v, x ;
      float t[10] ;
      int i ;
      srand(time(NULL)) ;
      for ( i=0 ; i < 10 ; i++ ) {
            x = rand()%100 ;
            m = x + m ;
            t[i]= x;
      }
      m = m/10 ;
      for (int i=0 ; i < 10 ; i++ ) {
            v=v+(t[i]-m)*(t[i]-m) ;
      }
      v = v/10 ;
      printf("x=%f, v=%f\n", m, v) ;
}
```

5. Analyse de code (2 pts)

Qu'affiche le programme ci-dessous ? Expliquez pourquoi.

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

int main (void) {
    float x=0, y = 0 ;
    while(1) {
        y=x;
        x=x+1;
        if (x==y) break ;
        if (x==1e12) {
              printf("y=%f\n", y) ;
              return(EXIT_SUCCESS) ;
        }
    }
    printf("x=%f\n", x) ;
}
```

```
x=16777216.000000 =  1.6777216e7
Lorsqu'il y a plus de 7 chiffres significatifs, la précision des float fait que l'ajout de 1 ne change plus le résultat.
```