Correction avec explication des séries 1 et 2 du TD Informatique / programmation C : SMP4/SMC4

2019-2020

Correction de la série 1

Exercice 1

Algorithme

```
Algorithme  division_entiere
Déclarations
   Variables :
        A, b, q, r : entiers ;

Début
   Afficher(" Saisir deux entiers a et b ") ;
   Lire (a) ;
   Lire (b) ;
   Si (b=0) retour (-1) ;
   r=a ;
   q=0 ;
  tant que (r>=b) faire
        r=r-b ;
        q=q+1 ;
  fin tant que
  Afficher(" Le quotient = ",q, " le reste= ",r) ;
fin algorithme
```

Programme (en C):

```
#include <stdio.h>

int main()
{
   int a,b, q, r;

   printf("Saisir deux entiers a et b ");
   scanf("%d",&a);
   scanf("%d",&b);
   if (b==0) return (-1);
   r=a;
   q=0;
   while (r>=b)
   {
      r=r-b;
      q=q+1;
   }

   printf("le quotient q = %d  le reste r=%d",q,r);
   return (0);
}
```

Exercice 2

Algorithme

```
Algorithme  resistance_equivalante
Declarations
   Variables :
        R1, R2, R3, Rs, Rp : entiers ;

Debut
   Afficher(« Saisir les valeurs de R1,R2, R3») ;
   Lire (R1) ;
   Lire (R2) ;
   Lire (R3) ;
   Rs=R1+R2+R3 ;
   Rp=(R1*R2*R3)/(R1*R2+R1*R3+R2*R3) ;
   Afficher(" Rs = ",Rs, " Rp= ",Rp) ;
fin algorithme
```

Programme (en C):

```
#include <stdio.h>

main()
{
   float R1,R2,R3,Rs, Rp;

   printf("Saisir les valeurs de R1, R2 et R3 \n");
   scanf("%f",&R1);
   scanf("%f",&R2);
   scanf("%f",&R3);
   Rs=R1+R2+R3;
   Rp=(R1*R2*R3)/(R1*R2+R2*R3+R1*R3);

   printf("la resistance Rs = %10.2f Rp=%10.2f
",Rs,Rp);
```

```
}
```

Exercice 3

Algorithme

```
Algorithme  permutation
Declarations
    Variables :
        A, B, C, T : entiers ;

Debut
    Afficher(" Saisir les valeurs de A,B, C") ;
    Lire (A) ;
    Lire (B) ;
    Lire (C) ;
    T=A ;
    A=B ;
    B=C ;
    C=T ;
    Afficher(" les valeurs de A, B, C : ", A,B,C) ;
fin algorithme
```

Programme (en C):

```
#include <stdio.h>

main()
{
   int A, B, C, T;
   printf("Saisir les valeurs de A, B et C \n");
   scanf("%d",&A);
   scanf("%d",&B);
   scanf("%d",&C);
   T=A;
   A=B;
   B=C;
   C=T;
   printf("A = %d B=%d   C=%d ",A,B,C);
}
```

Exercice 4

Algorithme

```
Algorithme  calcul_salaire
Declarations
   Constantes :
        tauxb=3.397 ;
        tauxs=4.322 ;
        tauxr=2.2 ;
   Variables :
       nhb, nhs, sb, sr, sf : reels ;

Debut
 Afficher(" Saisir le nombre d'heure de base") ;
    Lire (nhb) ;
 Afficher(" Saisir le nombre d'heure supp") ;
    Lire (nhs) ;
    sb=nhb*tauxb+nhs*tauxs ;
    sr=sb*tauxr/100 ;
    sf=sb-sr ;
    Afficher(" le salaire brut : ", sb) ;
    Afficher(" le retenu : ", sr) ;
    Afficher(" le salaire final : ", sf) ;
  fin algorithme
```

Programme (en C):

```
#include <stdio.h>

const tauxb=3.397, tauxs=4.322, tauxr=2.2;

main()
{
   float nhb, nhs, sb, sr, sf ;

   printf("Saisir le nombre d'heure de base :");
   scanf("%f",&nhb);
   printf("Saisir le nombre d'heure supp :");
   scanf("%f",&nhs);
   sb=nhb*tauxb+nhs*tauxs;
   sr=sb*tauxr/100;
   sf=sb-sr;
   printf("Le salaire brut : %10.2f \n",sb);
   printf("Le retenu       : %10.2f \n",sr);
   printf("Le salaire final: %10.2f \n",sf);
}
```

Exercice 5

Algorithme : 1er cas : 5 variables au total

```
Algorithme  calcul_somme5v
Declarations
   Variables :
        V1, V2, V3, V4, S : réels ;

Debut
   Afficher(" Saisir les valeurs de V1, V2, V3,V4) ;
   Lire (V1) ;
   Lire (V2) ;
   Lire (V3) ;
   Lire (V3) ;
   S=V1+V2+V3+V4 ;
   Afficher(" La sommes S =  ", S) ;
fin algorithme
```

Algorithme : 2er cas : 2 variables seulement

```
Algorithme  calcul_somme2v
Declarations
   Variables :
        V, S : réels ;

Debut
   Afficher(" Saisir les valeurs de V1, V2, V3,V4) ;
   Lire (V) ;
   S=V ;
   Lire (V) ;
   S=S+V ;
   Lire (V) ;
   S=S+V ;
   Lire (V) ;
   S=S+V ;
   Afficher(" La sommes S =  ", S) ;
fin algorithme
```

Programme (en C):

```
 // 1er Cas : 5 variables au total
#include <stdio.h>

main()
{
   float v1, v2, v3, v4, s;

   printf("Saisir la valeur de v1, v2, v3, v4  :");
   scanf("%f",&v1);
   scanf("%f",&v2);
   scanf("%f",&v3);
   scanf("%f",&v4);
   s=v1+v2+v3+v4;
   printf("LA somme des variables v1..v4  : %f",s);

}
```

```
 // 2eme cas : 2 variables au total
#include <stdio.h>

main()
{
   float v, s;

   printf("Saisir la valeur de v  :");
   scanf("%f",&v);
   s = v ;
   scanf("%f",&v);
   s=s+v ;
   scanf("%f",&v);
   s=s+v ;
   scanf("%f",&v);
   s=s+v;
   printf("La somme des 4 valeurs v1..v4  : %f",s);

}
```

Exercice 6

prod_a : correct,
Newbal : correct,
9ab6 : incorrect car le nom ne commence pas par un chiffre,
c123 : correct,
sum.of : incorrect, pas de caractères spéciaux y compris le point,
Abcd : correct,
$total : incorrect, pas de caractères spéciaux
moyenne : correct,
_c3 : correct, _ le seul caractère accepté,
new bal : incorrect, espace interdit,
grade1 : correct,
1234 : incorrect, le nom doit commencer par une lettre ou _.

Exercice 7

Algorithme

```
Algorithme calcul_surface
Declaration
    Variables
        R, S : réels ;
Début
        R=2.59 ;
        S=3.14*R*R ;
        Afficher("La surface du disque : ", S) ;
Fin algorithme
```

Programme (en C):

```
#include <stdio.h>

main()
{
   float R, S;
   R=2.59;
   S=R*R*3.14;

   printf("La surface S = %f ",S);

}
```

Exercice 8

Algorithme

```
Algorithme calcul_volume_cylindre
Declaration
    Variables
        R, V, p : réels ;
Début
        R=5.2 ;
        P=2.8
        V=3.14*R*R*P ;
        Afficher("Le volume du cylindre : ", V) ;
Fin algorithme
```

Programme (en C):

```
#include <stdio.h>

main()
{
   float R, V, P;
   R=5.2;
   P=2.8;
   V=R*R*3.14*P;

   printf("Le volume du cylindre V = %f ",V);

}
```

Exercice 9

Algorithme :

```
Algorithme  conversion_temperature
Declarations
   Variables :
        tf, tc : réels ;

Debut
   Afficher("Saisir la température Fahrenheit ") ;
   Lire (tf) ;
   Tc = 5*(tf-32)/9 ;
   Afficher("La température en degré Celsius : ",tc) ;

Fin algorithme
```

Programme (en C):

```
#include <stdio.h>

main()
{
   float tc, tf;

   printf("Saisir la valeur de la temperature f :  ");
   scanf("%f",&tf);
   tc=5*(tf-32)/9;
   printf("La valeur de la temperature °c  : %f ",tc);

}
```

Correction de la série 2

Bien que tous les exercices demandent de construire un algorithme pour résoudre le cas traité, je suppose que les exercices traités dans la série 1 sont suffisant pour comprendre le principe global d'un algorithme. Cependant, présenter le programme traitant le cas de chaque exercice de la série 2 avec des explications en marges serait plus intéressant du fait qu'on peut retaper le programme et le compiler pour le tester.

Exercice 1 - 1 : solution proposée

```
#include <stdio.h>

main()
{
   int N, i, S1, S2;
   printf("Donner la valeur de N : ");
   scanf("%d",&N);
   S1=0;
   S2=0;
   for(i=0; i<=N; i++)
   {
      S1=S1+i;
      S2=S2+(2*i+1);
   }
   printf("S1=%d  ,  S2 = %d",S1,S2);
}
```

Exercice 1 – 1 explications

- La première partie de l'exercice concerne le calcul des deux sommes S1 et S2.
- Puisqu'il s'agit du même nombre de termes à additionner donc il suffit de faire une seule boucle
- De même, le nombre de terme est connu, donc la boucle convenable est for…

Exercice 1 - 2 : solution proposée

```
#include <stdio.h>
main()
{
   int S, d, f, i ;
   printf("Donner la valeur de d et f : \n");
   scanf("%d",&d);
   scanf("%d",&f);

   S=0;
   for (i=d; i<=f; i++)
   {
      if (i<0) S=S+(-i) ; else S=S+i;
   }
   printf("La somme S = %d \n",S);
}
```

Exercice 1 - 2 : explications

- La boucle convenable est for , pour la même raison que celle dans le 1er cas de l'exercice,
- La somme des valeurs absolues donc :
  - Si i<0 abs(i) = -i, donc S=S+(-i)
  - Si i>0 abs(i)= I, donc S=S+i

Exercice 2 : solution proposée

```
#include <stdio.h>

main()
```

Exercice 2 : explications

- Le nombre d'itération n'est pas connu au préalable, donc la boucle convenable est while() {…} ou do { …}

```
{
   int N, s, v;
   printf("Donner une valeur v : ");
   scanf("%d",&v);
   s=0;
   N=0;
   while (s<=v)
   {
      N++;
      s=s+N;
   }
   printf("v=%d  ,  s = %d   N =%d" ,v,s,N);
}
```

while (),

- On incrémente N de 1, puis on l'ajoute à la somme initialisé à 0 avant la boucle.

Exercice 3 : solution proposée

```
#include <stdio.h>

main()
{
   int N;
   float r, S;

   printf("Saisir des nombres réels (0 pour quitter) : ");
   N=0;
   S=0;
   do
   {
      scanf("%f",&r);
      N++;
      S=S+r;
   } while (r!=0);
   printf("%d nombres saisis dont la somme =%f" ,N,S);
}
```

Exercice 3 : explications

- Le nombre d'itération n'est pas connu au préalable, donc la boucle convenable est while() {…} ou do { …} while (),
- On incrémente N de 1, puis on ajoute le nombre réel saisi à la somme initialisé à 0 avant la boucle.

Exercice 4 : solution proposée

```
#include <stdio.h>

main()
{
   int N;
   float num, den, S, Pi;
   S=0;
   num=-1;
   for (N=0; N<=100; N++)
   {
      num=(float)num*(-1);
      den=(float)2*N+1;
      S=S+num/den;
   }
   Pi=4*S;
   printf("La valeur de Pi =%f" ,Pi);
}
```

Exercice 4 : explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for() {…},
- Pour simplifier davantage, on utilise deux variables num (numérateur) et den (dénominateur),
- La puissance de (-1) est obtenue par la multiplication en chaque itération par (-1),
- (float) dans l'expression de num et de den veut dire que le résultat du calcul serait converti en un réel. Car, N est un entier.
- Pi=4*S ; car nous avons Pi/4 =S

Exercice 5 : solution proposée

```
#include <stdio.h>

main()
{
   int n;
   float signe, puissance, s, x;
   printf("Saisir la valeur de x :");
   scanf("%f",&x);
   s=1.0;
   signe =1.0;
   puissance = 1.0;
   for (n=1; n<=10; n++)
   {
      signe=signe*(-1);
      puissance=puissance*x;
      s=s+signe*puissance;
   }
   printf("La valeur de S =%f" ,s);
}
```

Exercice 5 : explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for() {…},
- Pour simplifier davantage, on utilise deux variables signe pour designer (-1)^n, et puissance pour designer x^n,
- s, signe et puissance sont initialisées à 1 avant la boucle pour éliminer le 1er terme de l'itération notamment n=0.

Exercice 6 : solution proposée

```
#include <stdio.h>

main()
{
   long N, i, H;

   printf("Saisir un entier N :");
   scanf("%d",&N);

   if (N<20)
   {
      H=1;
      for (i=1; i<N+1; i++) H=H*i;      // N!
   }
   else
   {
      H=1;
      for (i=1; i<11; i++) H=H*N;      // N^10
   }

   printf("La valeur de H =%d" ,H);
}
```

Exercice 6 : explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for() {…},

Exercice 7 : solution proposée

```
#include <stdio.h>

main()
{
  long n, i, b, p;

   printf("Saisir deux nombre b et n : \n");
   scanf("%d",&b);
   scanf("%d",&n);
   if (n>0 )
   {
      p=1;
      for (i=1; i<n+1; i++) p=p*b;  // b^n
      printf("La valeur de %d^%d  =%d" ,b,n,p);
   }
   else printf("La valeur de n est incorrecte ");
}
```

Exercice 7 : explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for() {…},

Exercice 8 : solution proposée

```
#include <stdio.h>
main()
{
   int n, i, r, N;

   printf("donner un nombre entier ");
   scanf("%d",&n);
   N=0;
   do
   {
      r=n%2;
      if (r) N++;
      n=n/2;
      printf("%d",r);
   } while(n>0);

   printf("\n Le nombre N = %d", N);
}
```

Exercice 8 : explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for() {…},
- La conversion en binaire se fait par la division sur 2, alors les 1 dans la représentation binaire représente le reste de la division sur 2,
- Donc, r= n%2, cad, le reste de la division, si r<>0 cad r=1, alors on le compte par l'incrémentation de N (cad, N++),
- Ici if(r) N++ est suffisant, car r=1 donc une valeur logique vrais c'est équivalant à if(r !=0)

Exercice 9 -a : solution proposée

```
#include <stdio.h>
main()
{
   int n, r, S, i;

   printf("Donner un entier : ");
   scanf("%d",&n);
   i=1;
   S=1;
   for (i=1; i< (n/2); i++)
   {
```

Exercice 9 -a : explications

- Le nombre d'itération connu au préalable, donc la boucle convenable est for {…},
- On initialise S et i à 1 car, on sait d'avance que 1 divise tous les nombres
- La boucle s'arrête à n/2 car, les diviseurs de n sont tous compris entre 1 et n/2

```
      r=n%i;
      if (r==0) S=S+i;
      printf("%d  %d \n",i,r);
   }
   if (n==S) printf("Le nombre %d est parfait ",n) ;
    else printf("Le nombre %d n'est parfait ",n);
}
```

Exercice 9 - b : solution proposée

```
#include <stdio.h>

main()
{
   int n, r, S, i;

   printf("Donner un entier : ");
   scanf("%d",&n);
   S=0;
   for(i=2; i<=n/2; i++)
   {
      r=n%i;
      if (r==0) S=S+i;
   }
   if (S==0) printf("Le nombre %d est premier ",n) ;
    else printf("Le nombre %d n'est premier ",n);
}
```

Exercice 9 - b: explications

- Le nombre d'itération est connu au préalable, donc la boucle convenable est for{…},
- On initialise S à 0 puis si un nombre compris entre 2 et n/2 divise n, alors on ajout i à S, ou 1 au lieu de i, juste pour marquer qu'un nombre divise n,
- Enfin si S n'est pas nul, cad, un nombre divise n ; donc n n'est pas premier.

Exercice 9 - c : solution proposée

```
#include <stdio.h>

main()
{
   int n, N, r, S, i;

   printf("Donner un entier : ");
   scanf("%d",&N);
   for (n=1; n<=N; n++)
   {
      S=0;
      for (i=2; i<=n/2;i++)
      {
         r=n%i;
         if (r==0) S=S+i;
      }
      if (S==0) printf("Le nombre %d est premier \n",n);
}
```

Exercice 9 - c: explications

- on teste les nombres de 1 à N par le programme précédent, et on affiche seulement ceux qui sont premiers

Exercice 10 : solution proposée

```
#include <stdio.h>
main()
{
   int n, m, i, r,sn, sm;
   printf("Donner la valeur de n :");
   scanf("%d",&n);
   printf("Donner la valeur de m :");
   scanf("%d",&m);
   i=2;
   sn=2;
   while (i<=(n/2))
   {
      r=n%i;
      if (r==0) sn=sn+i;
      i++;
   }
   i=2;
   sm=2;
   while (i<=(m/2))
   {
      r=m%i;
      if (r==0) sm=sm+i;
      i++;
   }
   if ((n==sm) && (m=sn))
      printf("Les nombres %d et %d sont amis ",n,m);
   else
      printf("Les nombres %d et %d ne sont pas amis ",n,m);
 }
```

Exercice 10 : explications

- on calcul la somme des diviseurs de n soit sn, et la somme des diviseurs de m soit sm,
- puis si n=sm et m=sn alors n et m sont amis si non ils ne sont pas amis.

Exercice 11 : solution proposée

```
#include <stdio.h>
main()
{
   int n, m, r,s, a, b;
   printf("Donner la valeur de n :");
   scanf("%d",&n);
   printf("Donner la valeur de m :");
   scanf("%d",&m);
   a=n; b=m;
   s=0;
   while (a>0)
   {
      r=a%2;
      if (r) s=s+b;
      a=a/2;
      b=b*2;
   }
   printf("%d x %d = %d : %d",n,m,s,n*m);
}
```

Exercice 11 : explications

- *Les nombres à multiplier sont saisis dans les variables n et m,*
- *Les variables a et b sont les variables du traitement qui vont recevoir les valeurs de n et m,*
- *Les résultats affichés sont respectivement :*
  - *n, m les valeurs saisies,*
  - *s la valeur calculer par la méthode russe*
  - *n*m la valeur calculer en utilisant l'opérateur de multiplication ordinaire, ceci juste pour comparer les résultats.*

Exercice 11 : solution proposée

```
#include <stdio.h>
main()
{
   int n, m, r,s, a, b;
   printf("Donner la valeur de n :");
   scanf("%d",&n);
     printf("Donner la valeur de m :");
   scanf("%d",&m);
   a=n; b=m;
   s=0;
   while (a>0)
   {
      r=a%2;
      if (r) s=s+b;
      a=a/2;
      b=b*2;
   }
   printf("%d x %d = %d : %d",n,m,s,n*m);
}
```

Exercice 11 : explications

- Les nombres à multiplier sont saisis dans les variables n et m,
- Les variables a et b sont les variables du traitement qui vont recevoir les valeurs de n et m,
- Les résultats affichés sont respectivement :
    - n, m les valeurs saisies,
    - s la valeur calculer par la méthode russe
    - n*m la valeur calculer en utilisant l'opérateur de multiplication ordinaire, ceci juste pour comparer les résultats.

Exercice 12 : solution proposée

```
#include <stdio.h>

int main()
{
   int X, Y, r;
   // equation 47*X+13*Y=1
   for (X=-100; X<101; X++)
   {
      for (Y=-100; Y<101; Y++)
      {
         r=47*X+13*Y;
         if (r==1)
         {
            printf("La solution de l'equation est : ");
            printf("X0= %d, Y0=%d",X,Y);
            return(1);
         }
      }
   }
   printf("L'equation n'a pas de solution");
   return (0);
}
```

Exercice 12 : explications

- C'est un exercice typique pour comprendre le rôle de return() et le type de la fonction main() ; cad, int main() ou autre.
- Les inconus de l'équation X et Y sont des entiers relatifs compris entre -100 et 100. Donc, il suffit de faire deux boucle imbriquées, pour parcourir tous les X de -100 à 100 et pour chaque X, parcourir Y de -100 à 100, une fois la valeur r=47*X+13*Y est égale à 1, donc l'équation vérifié et donc la solution trouvée.
- Une fois la solution X0 et Y0 trouvée, le programme affiche le résultat et s'arrête. Pour arrêter un programme c, on utilise la fonction return(), seluement, cette fonction est faite pour retourner une valeur. C'est pas par hasard que cette fonction renvoie une valeur, justement pour savoir si la mission a été accomplie ou pas.
- Ici, si la solution est trouvée le programme affiche le résultat et s'arrête en signalant que la mission est accomplie (renvoie 1). Si aucune solution trouvée ; toutes les itérations sont expirer sans aucun résultat, alors, la valeur renvoyée est 0

Exercice 13 : solution proposée

```
#include <stdio.h>
 main()
{
   int n, r, i;
   printf("Saisir une valeur positive :");
   scanf("%d",&n);
   i=0;
   while (n>1)
   {
      r=n%2;
      if (r) n=3*n+1; else n=n/2;
      i++;
      printf("i =%d  n = %d \n",i,n);
   }
   printf("Le nombre de transformation est %d",i);
}
```

Exercice 13 : explications

- n le nombre à saisir,
- r le reste de la division de n par 2, si r=0 ; alors n est pair
- le traitement se poursuit tant que n>1
- i est un compteur qui compte les transformations.