TD n° 1 (Correction)

Révisions

Exercice 1.

Au musée de Marly, les tarifs d'entrée sont définis par les règles suivantes : gratuité pour les enfants d'au plus 6 ans, tarif réduit (3 euros) pour les enfants entre 7 et 12 ans et les personnes agées à partir de 65 ans, et tarif plein (5 euros) pour les autres. De plus, les habitants de la ville de Marly ont droit à une réduction de 1 euro.

Pour diminuer le temps d'attente aux caisses, on souhaite programmer des machines en libre service pour l'achat du billet d'entrée.

1. Écrire une fonction **tarif** qui demande à l'utilisateur l'âge de la personne à laquelle le billet est destiné, puis qui demande si cette personne habite à Marly, et enfin qui affiche le prix à payer.

2. Donner une instruction contenant un appel à votre fonction (par exemple pour la tester).

Correction : *Il faut essayer de faire les 4 premiers exos, (pas plus de 30 mn sur le 1er), les autres sont pour les tres bons etudiants, l'exo 6 sera probablement fait en TD2*

```
void tarif() {
    int age;
    int prix;
    string marly;
    cout << "Entrer l'âge du visiteur ";
    cin >> age;
    cout << "Est-ce que le visiteur habite à Marly ? (oui/non) ";
    cin >> marly;
    if (0 <= age and age < 7) {
        prix = 0;
    } else {
        if (12 < age and age < 65)
            prix = 5;
        else
            prix = 3;
        if (marly == "oui")
            prix = prix - 1;
    }
    cout << "prix a payer " << prix << " euros" << endl;
}

int main() {
    tarif();
}
```

Exercice 2. Un nombre premier est un entier qui n'a pas d'autre diviseur que 1 et lui-même.

1. Réaliser une fonction **estDiviseur** qui prend en argument deux entiers et renvoie `true` si le premier entier est un multiple du deuxième, `false` sinon. On rappelle que `a % b` est le reste de la division de a par b.

2. Réaliser une fonction **estPremier** qui prend en argument un entier et renvoie `true` si cet entier est un nombre premier, `false` sinon. Vous devez utiliser la fonction **estDiviseur**.

3. Réaliser une fonction **premiersBornes** qui prend en argument un entier n et qui affiche tous les nombres premiers inférieurs à n. Vous devez utiliser la fonction **estPremier**.

4. Réaliser une fonction **premiersPremiers** qui prend en argument un entier n et qui affiche les n premiers nombres premiers. Vous devez utiliser la fonction **estPremier**.

5. Écrire un programme qui demande à l'utilisateur un entier n et qui affiche les n premiers nombres premiers en faisant appel à la fonction **premiersPremiers**.

Correction :

```
bool estDiviseur(int a, int b) {
    return a%b==0;
}

bool estPremier(int m) {
    for(int i=2; i<m; i++) {
        if (estDiviseur(m,i)) {
            return false;
        }
    }
    return true;
}

int premiersBornes(int n) {
    for(int k=2; k<n; k++) {
        if (estPremier(k)) {
            cout << k << " ";
        }
    }
}

int premiersPremiers(int n) {
    int compteur = 0;
    int nombre = 2;
    while(compteur<n) {
        if (estPremier(nombre)) {
            cout << nombre << " ";
            compteur = compteur + 1;
        }
        nombre = nombre + 1;
    }
}

int main() {
    int n;
    cout << "Combien de nombres premiers voulez-vous ? ";
    cin >> n;
    premiersPremiers(n);
}
```

Exercice 3.

Écrire une fonction **lectureVB** (pour lectureValeurBornée) avec deux paramètres, un min et un max, qui demande à l'utilisateur d'entrer la valeur d'une variable réelle comprise entre ces deux bornes, puis qui renvoie cette valeur. La fonction doit afficher un message d'erreur si la valeur saisie n'est pas comprise entre les bornes et re-demander à l'utilisateur une nouvelle

saisie jusqu'à pouvoir renvoyer une valeur correcte. Par exemple on peut utiliser cette fonction pour demander de saisir un nombre d'heures en imposant qu'il soit compris entre 0 et 23.
Correction :

```
float lectureVB(float min, float max) {
   float res, temp;
   // on verifie que les bornes sont correctes sinon on les inverse
   if (min > max) {
       temp = min;
       min = max;
       max = temp;
   }

  // la saisie est a faire avec un do while
  // puisqu'il faut executer le bloc au moins une fois
  // meme s'ils vont preferer la version while
   do {
       cout << "entrez un entier compris entre " << min << " et " << max << endl
       cin >> res ;
       if (res < min) {
           cout << "recommencez  en donnant une valeur plus elevee" << endl;
       } else if (res > max)  {
            cout << "recommencez  en donnant une valeur plus basse" << endl;
       }
   }  while ( ( res < min) or (res > max) );

   // Autre solution: la saisie avec un while
   cout << "entrez un entier compris entre " << min << " et " << max << endl ;
   cin >> res;
   while ( (res < min) or (res > max)) {
       cout << " valeur incorrecte, recommencez  en donnant une valeur " ;
       if (res < min) {cout << "plus elevee" << endl;
       } else {
             cout << "plus basse" << endl;
       }
       cout << "entrez un entier compris entre " << min << " et " << max ;
       cin >> res ;
    }
   return res;
}
```

Exercice 4.

Une banque fait un prêt à une personne pour un montant total de s_0 euros. Cette personne rembourse chaque mois un montant fixe r et paye en plus un intérêt variable $i = i_m * s$ où i_m est le taux d'intérêt mensuel fixe et s la somme restant à rembourser (avant déduction du remboursement mensuel).

Écrire un programme qui demande à l'utilisateur le montant emprunté (s_0), le montant remboursé par mois (r) et le taux d'intérêt mensuel (i_m, qui doit être compris entre 0,1 % et 1,5 % pour ne pas dépasser le taux d'usure annuel fixé à 19,96 %), puis qui calcule et affiche la durée du remboursement et la somme des intérêts versés. Pour les lectures, vous pourrez utiliser la fonction **lectureVB** écrite précédemment quand cela vous paraitra pertinent.

Correction :

```
int pret() {
    double s0 = 0, r = 0, im = 0 ;

    // saisie des valeurs initiales ----------------------------------
    do {
        cout << "Montant emprunte, > 0 " ;
        cin >> s0 ;
    } while (s0 <= 0.0);

    cout << "Taux d interet ";
    im = lectureVB(0.1,1.5)/100 ;

    do {
        cout << "Montant rembourse chaque mois, r > s0*im ";
        // (il faut au moins rembourser les intérêts, s0*im)
        cin >> r;
    } while ( (r <= s0*im) or (r >= s0) ) ;

//ou bien avec LVB
// cout << "Montant rembourse chaque mois :";
// on n'a pas besoin de préciser les bornes, lectureVB le fait déjà
// r = lectureVB(s0*im, s0);


// declaration variables  ----------------------------------
    double cumul = 0.0 ;    // somme des interets
    double S = s0 ;         // somme restant a rembourser
    int nbR = 0 ;           // nb de remboursements effectues

    while (S > 0.0) {
        nbR = nbR +1;
        cumul = cumul + S*im;
        S = S*(1+im) - r;
        cout << nbR << ": S=" <<  S << ", cumul=" << cumul << endl;
    }

    cout << " Sur " << nbR <<  "mois, Interets verses : "<< cumul << endl;
    return 0;
}
```

Exercice 5.

Réaliser un programme qui lit deux entiers positifs a et b tels que $a \leq b$ et un réel x, puis calcule $\sum_{i=a}^{b} \frac{x^i}{i!}$

1. en utilisant les fonctions **factorielle** et **puissance**, que vous réaliserez ;

Correction :

```
double puissance( double z, int e) {
   double p = 1.0;
   // la boucle suivante ne fait rien si e est <= 0 //
   for (int i=1; i <= e; i++) p= p*z;
   return p;
}

int factorielle(int e) {
    int fac=1;
    for (int i=1; i <= e; i++) fac = fac*i;
    return fac;
}

int main() {
// lire deux entiers positifs a,b, avec a <= b, un reel x
//   et calculer la formule en faisant appel aux fonctions
//   puissance et factorielle a chaque iteration

  int a=0, b=0;
  double x=0.0, som=0.0;

 do {
    cout << "entrez 2 entiers positifs, le 2eme >= au 1er, et un reel > 0";
    cin >> a >> b >> x;
 } while ((a < 0) or  (b < a) or  (x < 0));

  for (int i=a; i <= b; i++) {
      som = som + puissance(x,i) / factorielle(i);
  }
  cout << "somme =" << som;
 }
```

2. en n'utilisant pas ces fonctions (et en réalisant un algorithme plus efficace en nombre d'opérations effectuées).

Correction :

```
int main() {
   int a=0, b=0, fact=1;
   double x=0.0, som=0.0, puis=1.0;

 do {
   cout << "entrez 2 entiers positifs, le 2eme >= au 1er, et un reel > 0";
   cin >> a >> b >> x;
 } while ((a < 0) or  (b < a) or  (x < 0));

 // pour ne pas repartir a 0 chaque fois, on calcule  x^(a-1)/(a-1) !
 for (int i=1; i < a ; i++) {
      puis = puis * x;
      fact = fact * i;
 }

 for (int i=a ; i <= b ; i++) {
     puis = puis * x;
     fact = fact * i;
     som = som + puis / fact;
```

```
    }
    cout << "somme =" << som;
  }
```

Exercice 6.

Soit le programme ci-dessous qui n'est pas un exemple de bonne programmation. Précisez la portée de chacune des variables et donnez les valeurs affichées.

```
(1) #include <iostream>
(2) using namespace std;
(3) int x = 10;
(4) int main() {
(5)   { int x = 5;
(6)     cout << "Un, x vaut :" << x << endl;
(7)   }
(8)   { cout << "Deux, x vaut :" << x << endl;
(9)   }
(10)  for (int x = 0; x < 3; x++) {
(11)    cout << "Trois, x vaut :" << x << endl;
(12)  }
(13)  cout  << "Quatre, x vaut :" << x << endl;
(14)  for (x = 0; x < 3; x++) {
(15)    cout << "Cinq, x vaut :" << x << endl;
(16)  }
(17)  cout << "Six, x vaut :" << x << endl;
(18)  return 0;
(18) }
```

Correction :

```
(1) #include <iostream>
(2) using namespace std;
(3) int x = 10;    // ceci est une variable globale
(4) int main() {

// ici un nouveau bloc, et variable locale a ce bloc1
(5)   { int x = 5;
(6)     cout <<"Un, x vaut :" << x << endl;   // affiche 5
(7)   }

// ci dessous, un nouveau bloc,  bloc2
// ne definissant pas de variable, donc le x est le x global
(8)   { cout <<"Deux, x vaut :" << x << endl;  // affiche 10
(9)   }

// ici un bloc "for" avec une variable locale au bloc
(10)  for (int x = 0; x < 3; x++) {
(11)    cout <<"Trois, x vaut :" << x << endl; // affiche 0, 1, 2
(12)  }

// a nouveau la variable globale, car aucune variable n'est defini dans le main
(13)  cout <<"Quatre, x vaut :" << x << endl;   // affiche 10

// ici un bloc "for" sans  variable declaree dans le bloc
// donc c'est la variable globale
```

```
(14)  for (x = 0; x < 3; x++) {
(15)    cout <<"Cinq, x vaut :" << x << endl;   // affiche 0, 1, 2
(16)  }

// il s'agit toujours de la variable globale qui a ete modifiee dans le "for"
// et qui valait 3 en sortant du "for"
(17)  cout <<"Six, x vaut :" << x << endl;  // affiche 3
(18)  return 0;
(18) }
```

Exercice 7.

Donnez un programme qui pour tout entier strictement positif affiche en sortie son écriture en chiffres romains. Vous pourrez dans un premier temps considérer le système d'écriture simple qui, par exemple, à 1997 associe `MDCCCCLXXXXVII` ; puis considérer le système qui, toujours par exemple, à 1997 associe `MCMXCVII`.

Correction : *Version simple:*

```
int main() {
    int n;
    cin >> n;
    while (n >= 1000)  {
        cout << 'M';
        n = n - 1000;
    }
    if (n >= 500)   {
        cout << 'D';
        n = n - 500 ;
    }
    while (n >= 100)   {
        cout << 'C';
        n = n - 100 ;
    }
    if (n >= 50)    {
        cout << 'L';
        n = n - 50 ;
    }
    while (n >= 10)    {
        cout << 'X';
        n = n - 10 ;
    }
    if (n >= 5)     {
        cout << 'V';
        n = n - 5 ;
    }
    while (n >= 1)     {
        cout << 'I';
        n = n - 1 ;
    }
    cout << endl;
}
```

Et voilà pour le second (écrit en C, à passer en C++ comme ci-dessus):

```
program RomainJoli(input, output);
var
   n : integer;
begin
```

```
   readln(n);
   while n >= 1000 do begin write('M');  n := n - 1000 end;
   if n >= 900 then   begin write('CM'); n := n - 900 end;
   if n >= 500 then   begin write('D');  n := n - 500 end;
   if n >= 400 then   begin write('CD'); n := n - 400 end;
   while n >= 100 do  begin write('C');  n := n - 100 end;
   if n >= 90 then    begin write('XC'); n := n - 90 end;
   if n >= 50 then    begin write('L');  n := n - 50 end;
   if n >= 40 then    begin write('XL'); n := n - 40 end;
   while n >= 10 do   begin write('X');  n := n - 10 end;
   if n >= 9 then     begin write('IX'); n := n - 9 end;
   if n >= 5 then     begin write('V');  n := n - 5 end;
   if n >= 4 then     begin write('IV'); n := n - 4 end;
   while n >= 1 do    begin write('I');  n := n - 1 end;
   writeln
end.
```