CORRIGES

Cahier de TD n° 1

Cahier de TD n° 1

THEME 1 : Stockages et types de variables, affectations, expressions

Identifiants

Variables et initialisations

Variables et types

Calculs simples

THEME 2 : tests, conditions, sélection

Test simple, si…alors…sinon

Test if…else if…else

Sélection switch…case

THEME 3 : boucle while et do while

THEME 4 : boucle for

THEME 1 : STOCKAGES ET TYPES DE VARIABLES, AFFECTATIONS, EXPRESSIONS

Identifiants :

Indiquez si les noms de variables suivants sont corrects, et sinon , pour quelle raison

1_entier un-entier un_entier x_435 X435 a' 6o6on identificateur_de_variable CoMPTeuR	**CORRIGE:** int main() { **// IDENTIFIANTS** **int 1_entier; // invalid suffix "_entier" on integer constant** **int un-entier; // error: expected initializer before '-' token** int un_entier; int x_435; int X435; **int a'; // error : missing terminating ' character** **int 6o6on; // error: invalid suffix "o6on" on integer constant** int identificateur_de_variable; int CoMPTeuR; return 0; }

Variables et initialisations :

remplissez le tableau suivant avec les valeurs prises par les variables

```
int main()
{
      long a, b , compteur, limite;
      double x, deriv, vitesse;
      char lettre, let2;
}
```

	a	b	compteur	limite	x	deriv	vitesse	lettre	let2
a = -14;									
lettre = 'a'+a;									
b = 3-a;									
compteur = 0;									
limite = b+b;									
b = b-1;									
x = 6.022E+23;									
vitesse = x * 1.0E-17;									
deriv = vitesse*(b-a);									
compteur = compteur % limite;									

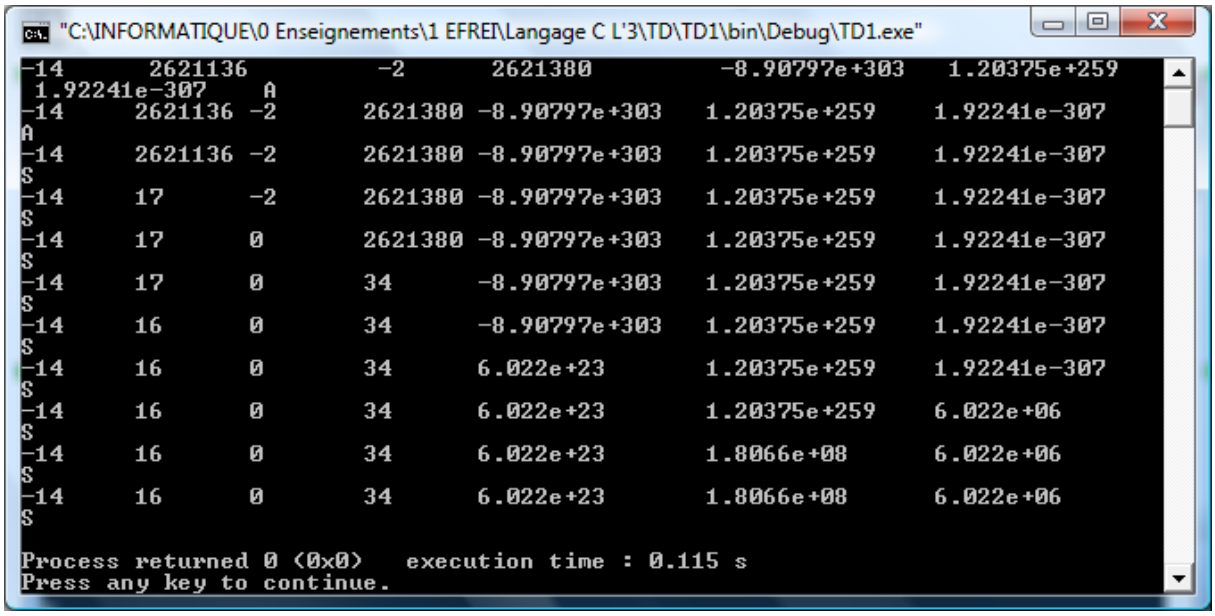
CORRIGE:

```
#include <stdio.h>      // les E/S en C
#include <iostream>   // les E/S en C++

using namespace std;// espace de noms pour le C++ (étudié plus tard)

int main()
{
// IDENTIFIANTS
//  int 1_entier; invalid suffix "_entier" on integer constant
//  int un-entier; error: expected initializer before '-' token|
    int un_entier;
    int x_435;
    int X435;
//  int a'; error : missing terminating ' character
//  int 6o6on; error: invalid suffix "o6on" on integer constant
    int identificateur_de_variable;
    int CoMPTeuR;

// VARIABLES ET INITIALISATIONS
    long a, b, compteur, limite;
    double x, deriv, vitesse;
    char lettre, let2;

    a = -14;
    // en C
    // %g mieux que %lf car notation exponentielle
   printf("%ld \t %ld \t %ld \t %ld \t %g \t %g \t %g \t %c \t %c \n", a, b, compteur, limite, x,
deriv, vitesse, lettre, let2);
     // en C++
    cout << a << '\t' << b << '\t' << compteur << '\t'  << limite << '\t' << x << '\t' << deriv << '\t'
                                          << vitesse << '\t' << lettre << '\t' << let2 << endl;
    // a vaut -14 puis les autres variables prennent des valeurs aléatoires
    lettre = 'a'+a; // 'a' code ascii de a vaut 97, 97 – 14 = 83 code ascii de S
    cout << a ….
    b = 3-a; // 3 – (-14) = 17
    cout << a ….
    compteur = 0;
    cout << a ….
    limite = b+b; // 17 + 17 = 34
    cout << a ….
    b = b-1; // 17 – 1 = 16
    cout << a ….
    x = 6.022E+23;
    cout << a ….
    vitesse = x * 1.0E-17; //  6.022E+23 * 1.0E-17 =  6.022E+6
    cout << a ….
    deriv = vitesse*(b-a); // 6.022E+6 * (16 – (-14)) = 1.8066E+8
    cout << a ….
    compteur = compteur % limite; // 0 * 34 = 0
```

```
    cout << a ….

    return 0;
}
```

WARNINGS du compilateur sur les variables inutilisées et celles utilisées mais non initialisées

```
In function 'int main()':|
warning: unused variable 'un_entier'|
warning: unused variable 'x_435'|
warning: unused variable 'X435'|
warning: unused variable 'identificateur_de_variable'|
warning: unused variable 'CoMPTeuR'|
warning: 'let2' is used uninitialized in this function|
warning: 'lettre' is used uninitialized in this function|
warning: 'b' is used uninitialized in this function|
warning: 'compteur' is used uninitialized in this function|
warning: 'limite' is used uninitialized in this function|
warning: 'x' is used uninitialized in this function|
warning: 'deriv' is used uninitialized in this function|
warning: 'vitesse' is used uninitialized in this function|
||=== Build finished: 0 errors, 13 warnings ===|
```



Variables et types :

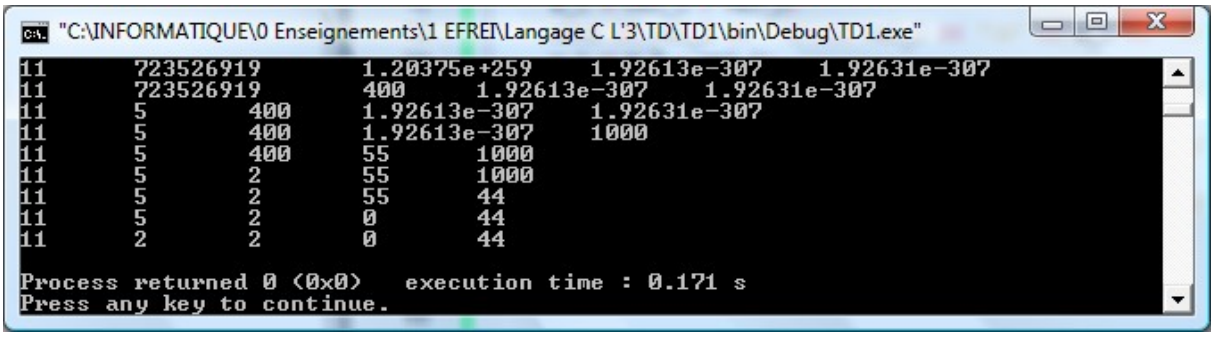
remplissez le tableau suivant avec les valeurs prises par les variables

```
int main()
{
long   val_1, val_2;
double  x_1, x_2, y_1;
}
```

	val_1	val_2	x_1	x_2	y_1
val_1 = 11;					
x_1 = 4.0E+2;					
val_2 = val_1 - 6;					
y_1 = 3 * x_1 - 200.0;					
x_2 = val_1 * val_2;					
x_1 = val_1 / val_2;					
y_1 = x_2 - val_1					
x_2 = val_2 / val_1					
val_2 = x_1;					

CORRIGE:

```
int main()
{
// IDENTIFIANTS
  ...
// VARIABLES ET INITIALISATIONS
   ...
// VARIABLES ET TYPES
   long   val_1, val_2;
   double  x_1, x_2, y_1;

   val_1 = 11;
   cout << val_1 << '\t' << val_2 << '\t' << x_1 << '\t' << x_2 << '\t' << y_1 << endl;
   // 11 puis des valeurs aléatoires pour les autres variables
   x_1 = 4.0E+2; // AFFICHAGE 400
   cout << a ….
   val_2 = val_1 – 6; // 11 – 6 = 5
   cout << a ….
   y_1 = 3 * x_1 – 200.0; // 3 * 400 -  200.0 = 1000
   cout << a ….
   x_2 = val_1 * val_2; // 11 * 5 = 55
   cout << a ….
   x_1 = val_1 / val_2; // 11 / 5 = 2 DIVISION ENTIERE
   cout << a ….
   y_1 = x_2 – val_1; // 55 – 11 = 44
   cout << a ….
   x_2 = val_2 / val_1; // 5 / 11 = 0  DIVISION ENTIERE
   cout << a ….
   val_2 = x_1; // 2
   cout << a ….
   return 0;
}
```



Calculs simples :

- Ecrivez un programme qui calcule et affiche la moyenne de trois nombres entiers saisis au clavier
- Ecrivez un programme qui calcule et affiche l'écart-type de trois nombre entiers saisis au clavier.

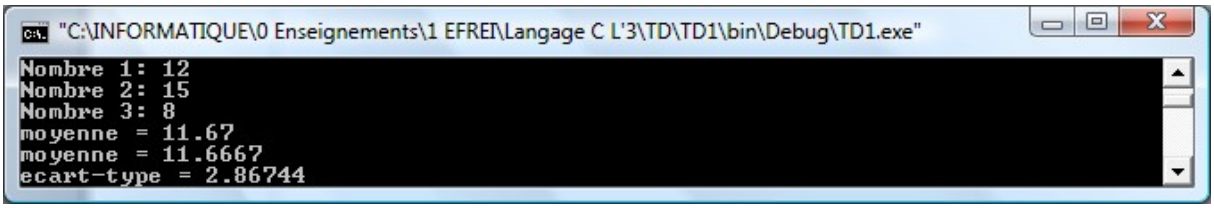
Formule de l'écart-type : $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{m})^2$, où n est le nombre de valeurs, ici n=3.

$\overline{m}$ est la moyenne des valeurs.

CORRIGE:

```
int main()
{
// IDENTIFIANTS
  ...
// VARIABLES ET INITIALISATIONS
  ...
// VARIABLES ET TYPES
  ...
// CALCULS SIMPLES
   int d, e, f;
   float moyenne;
   // en C
   printf("Nombre 1: ");
   scanf("%d", &d);
   printf("Nombre 2: ");
   scanf("%d", &e);
   // en C++
   cout << "Nombre 3: ";    cin >> f;
```

```
    moyenne = (d + e + f) /3.0; // pour un calcul en flottant
    printf("moyenne = %.2f \n", moyenne); // en C avec 2 décimales
    cout << "moyenne = " << moyenne << endl;

    float ecart_type;
    // sqrt fonction racine carree : square root
    ecart_type = sqrt(1/3.0 * ((d-moyenne)*(d-moyenne) + (e-moyenne)*(e-moyenne)
                                        + (f-moyenne)*(f-moyenne)));
    cout << "ecart-type = " << ecart_type;
    return 0;
}
```



- Ecrire un programme qui calcule la différence, en secondes, entre deux dates (en heures, minutes, secondes) de la même journée.

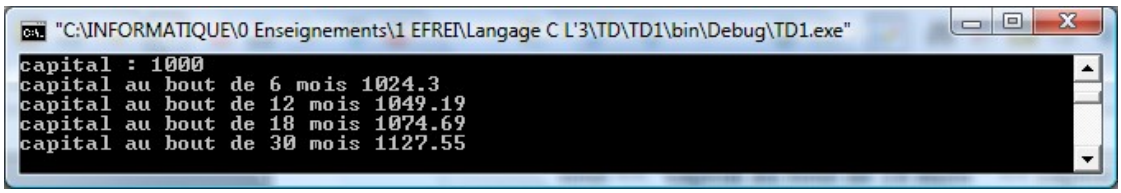
CORRIGE

```
int main()
{
// IDENTIFIANTS
  ...
// VARIABLES ET INITIALISATIONS
   ...
// VARIABLES ET TYPES
  ...
// CALCULS SIMPLES
   ... moyenne
   ... écart-type
// On ne vérifie pas la cohérence des donneés saisies
// heures dans [0, 23], minutes et secondes dans [0, 59]
   int heures1, heures2, minutes1, minutes2, secondes1, secondes2;
   long totalSecondes1, totalSecondes2, difference;
   cout << "première date" << endl;
   cout << "heures: "; cin >> heures1;
   cout << "minutes: "; cin >> minutes1;
   cout << "secondes: "; cin >> secondes1;
   cout << "seconde date" << endl;
   cout << "heures: "; cin >> heures2;
   cout << "minutes: "; cin >> minutes2;
   cout << "secondes: "; cin >> secondes2;
   totalSecondes1 = 3600 * heures1 + 60 * minutes1 + secondes1;
   totalSecondes2 = 3600 * heures2 + 60 * minutes2 + secondes2;
  // fabs fonction valeur absolue
   difference = fabs(totalSecondes1 - totalSecondes2);
   cout << "il y a " << difference << " secondes d'écart entre les 2 dates" << endl;
   return 0;
}
```

- Ecrivez un programme de suivi d'épargne : un capital *C* est placé sur un compte dont le taux d'intérêt semestriel *T* est de 2,43 %. Le programme doit calculer et afficher le capital au bout de : 6 mois, 12 mois, 18 mois et 30 mois.

CORRIGE

```
int main()
{
// IDENTIFIANTS
  ...
// VARIABLES ET INITIALISATIONS
   ...
// VARIABLES ET TYPES
  ...
// CALCULS SIMPLES
   ... moyenne
   ... écart-type
    ... différence de dates en secondes
        float capital, tauxSemestriel = 2.43;
        float capital6mois, capital12mois, capital18mois, capital30mois;
        cout << "capital : "; cin >> capital;
        capital6mois = (1 + tauxSemestriel/100) * capital;
        capital12mois = (1 + tauxSemestriel/100) * (1 + tauxSemestriel/100) * capital;
        capital18mois = (1 + tauxSemestriel/100)* capital12mois;
        capital30mois = (1 + tauxSemestriel/100) * (1 + tauxSemestriel/100) * capital18mois;
        cout << "capital au bout de 6 mois " << capital6mois << endl;
        cout << "capital au bout de 12 mois " << capital12mois << endl;
        cout << "capital au bout de 18 mois " << capital18mois << endl;
        cout << "capital au bout de 30 mois " << capital30mois << endl;
        return 0;
}
```



THEME 2 : tests, conditions, sélection

Rappel : lorsqu'on propose une saisie dans un programme, il faut systématiquement afficher un message pour indiquer à l'utilisateur ce qu'il doit saisir !

Test simple, si…alors…sinon

- Analyse d'un programme simple

```
int main()
{
// Test simple, si…alors…sinon
        double borne_basse, borne_haute;
        double valeur;
        long test_bas, test_haut;

        borne_basse = -3.2;
        borne_haute = 5.56;

        printf("saisissez une valeur reelle :");
        scanf("%lf",&valeur);

        test_bas = (valeur < borne_basse);
        test_haut = (valeur > borne_haute);

        if (test_bas == 1)
        {
        printf("la valeur est en dessous de l'intervalle");
        }
        if (test_haut == 1)
        {
        printf("la valeur est au dessus de l'intervalle");
        }
        if ((test_bas ==0) && (test_haut == 0)) // ET LOGIQUE
        {
        printf("la valeur est dans l'intervalle");
        }
        return 0;
}
```

les variables test_bas et test_haut peuvent-elles prendre d'autres valeurs que 0 et 1 ? justifiez votre réponse.

Combien de messages le programme peut-il afficher lors d'une exécution ? Expliquez pourquoi.

Indiquez les valeurs des variables et des conditions au cours du déroulement du programme lorsque l'utilisateur saisit :

a) 0.05

b) 6.33

CORRIGE

test_bas = (valeur < borne_basse);
test_haut = (valeur > borne_haute);
les variables test_bas et test_haut recoivent le résultat d'une expression logique valant 0 ou 1.

les 3 tests sont en fait exclusifs, donc 1 seul message s'affiche lors de l'exécution

a) 0,05 dans l'intervalle donc les 2 variables test_bas et test_haut valent 0
les 2 premiers tests sont faux et le 3ième vrai

b) 6.33 est en dehors de l'intervalle et est supérieur à la borne droite donc la variable test_bas vaut 0 et la variable test_haut vaut 1
les premiers et troisième tests sont faux, le second est vrai

- Ecrire un programme qui fait : la saisie de trois niveaux de pollution, exprimés en microgrammes par m^3 ($\mu g/m^3$), qui calcule la moyenne de ces valeurs, et indique, à l'aide d'un message, si un seuil de 480 $\mu g/m^3$ est dépassé, auquel cas le niveau est considéré comme dangereux. Les données sont des nombres à virgule (type réel);

  ## CORRIGE

```
#include <iostream>

using namespace std;

int main()
{
   float niveau1Pollution,  niveau2Pollution,  niveau3Pollution;
   float moyenne;

   cout << "Niveau 1 de pollution en microgrammes par m3 : "; cin >> niveau1Pollution;
   cout << "Niveau 2 de pollution en microgrammes par m3 : "; cin >> niveau2Pollution;
   cout << "Niveau 3 de pollution en microgrammes par m3 : "; cin >> niveau3Pollution;

   moyenne = (niveau1Pollution + niveau2Pollution + niveau3Pollution) / 3;

   if (moyenne > 480)
      cout << "Niveau dangereux de pollution" << endl;
   else
      cout << "Niveau de pollution en dessous du seuil dangereux" << endl;

   return 0;
}
```

- Ecrire un programme qui affiche une question et 4 possibilités de réponses, chacune précédée d'un numéro entre 1 et 4. L'utilisateur doit sélectionner le numéro de la réponse choisie, et le programme doit indiquer s'il s'agit de la bonne réponse ou si c'est une erreur.

CORRIGE:

```
int main()
{
// Test simple, si…alors…sinon
  . . .
   int choix;
   cout << "Quelle est l'ancienne monnaie italienne?" << endl;
   cout << "1 La couronne italienne" << endl;
   cout << "2 L'euro" << endl;
   cout << "3 La Lire" << endl;
   cout << "4 La Livre" << endl;
   cout << "votre numero choisi: "; cin >> choix;
   if (choix == 3)
      cout << "Bonne reponse!" << endl;
   else
      cout << "Mauvaise reponse" << endl;
  return 0;
}
```

Test if…else if…else

Reprenez le programme `intervalle` de la section précédente avec une ou des structures si…alors…sinon si Indiquez pourquoi cette structure est bien adaptée au problème traité.

CORRIGE : Les 3 tests étant exclusifs, la structure **if else if else** est appropriée.

```
int main()
{
// Test simple, si…alors…sinon
   . . .
// Test if…else if…else
   double borne_basse, borne_haute;
   double valeur;
   long test_bas, test_haut;

   borne_basse = -3.2;
   borne_haute = 5.56;
   printf("saisissez une valeur reelle :");
   scanf("%lf",&valeur);
   test_bas = (valeur < borne_basse);
   test_haut = (valeur > borne_haute);

   if (test_bas == 1)
   {
      printf("la valeur est en dessous de l'intervalle");
   }
   else if (test_haut == 1)
   {
      printf("la valeur est au dessus de l'intervalle");
   }
   else if ((test_bas == 0) && (test_haut == 0)) // ET LOGIQUE
   {
      printf("la valeur est dans l'intervalle");
   }
   else
      printf("On ne passe jamais ici!\n");
   return 0;
}
```

- Ecrire un programme qui indique si une année est bissextile ou non : une année multiple de 4 est en général bissextile. Attention toutefois, les années séculaires (multiples de 100) ne sont pas bissextiles, sauf si elles sont multiples de 400 ! (1400 et 1900 ne sont pas bissextiles, mais 1200 et 2000 le sont).

CORRIGE:

```
int main()
{
 // Test if…else if…else
            . . .
   int annee;
   cout << "annee: ";
   cin >> annee;
    // pour 2000 le test (annee%4) == 0 est évalué 2 fois
   // Manque d'optimisation!
   if ( ((annee%4) == 0) && ((annee%100) != 0) )
      cout << annee << " est une annee bissextile" << endl;
   else if ( ((annee%4) == 0) && ((annee%400) == 0) )
      cout << annee << " est une annee bissextile" << endl;
   else
      cout << annee << " n'est pas une annee bissextile" << endl;

   // meilleure programmation
   if ( (annee%4) == 0)
   {
      if (((annee%100) != 0) || ((annee%400) == 0)) // le OU LOGIQUE
         cout << annee << " est une annee bissextile" << endl;
      else
         cout << annee << " n'est pas une annee bissextile" << endl;
   }
   else
      cout << annee << " n'est pas une annee bissextile" << endl;

   // meilleur style
   if ( (annee%4) != 0)
       cout << annee << " n'est pas une annee bissextile" << endl;
   else if (((annee%100) != 0) || ((annee%400) == 0))
      cout << annee << " est une annee bissextile" << endl;
   else
      cout << annee << " n'est pas une annee bissextile" << endl;
}
```

- Ecrire un programme qui saisit un poids en kg, une taille en m et un âge; qui calcule un coefficient de poids C = poids / taille²; puis en fonction de l'âge, qui indique si la personne est en surpoids selon le barème suivant :

Tranche d'âge	Seuil de surpoids
16 ans et moins	$C > 27,3$
De 17 à 48 ans	$C > 23$
49 ans et plus	$C > 21,5$

CORRIGE

```
int main()
{
 // Test if…else if…else
    . . .
    . . .
   float poids, taille, coefficient;
   int age;

   cout << "poids en kg: "; cin >> poids;
   cout << "taille en m: "; cin >> taille;
   cout << "age: "; cin >> age;

   coefficient = poids / (taille * taille);

   if ( (age <= 16) && (coefficient > 27.3) )
      cout << "Vous avez moins de 16 ans et vous etes en surpoids" << endl;
   else if ( (age >= 17) && ((age) <= 48) && (coefficient > 23) )
      cout << "Vous avez entre 17 et 48 ans et vous etes en surpoids" << endl;
   else if ( (age >= 49) && (coefficient > 21.5) )
      cout << "Vous avez plus de 48 ans et vous etes en surpoids" << endl;
   else
      cout << "Vous n'êtes pas en surpoids pour votre age" << endl;

   return 0;
}
```

- Ecrire un programme qui indique les services disponibles sur une ligne ADSL en fonction de la distance au NRA (Nœud de Raccordement Abonné). Il faut saisir la distance D en mètre entre l'habitation et le NRA, puis calculer l'atténuation théorique A (en db) en utilisant la formule : A = D / 68;

Atténuation théorique (en db)	Services ADSL
A < 37	Internet 5 Mb; téléphonie IP; TV
37 <= A < 56	Internet 5 Mb; téléphonie IP
A >= 56	Internet 2 Mb

CORRIGE

```
int main()
{
   float distance, attenuation;

   cout << "distance en m: "; cin >> distance;

   attenuation = distance / 68;

   if (attenuation < 37)
      cout << " Services ADSL: Internet 5 Mb; telephonie IP; TV" << endl;
   else if ( (attenuation >= 37) && (attenuation < 56))
       cout << " Services ADSL: Internet 5 Mb; téléphonie IP" << endl;
   else // test inutile car on a nécessairement attenuation >= 56
       cout << " Services ADSL: Internet 2 Mb" << endl;
   return 0;
}
```

Sélection switch…case

- Dans la cuisine d'un restaurant où vous êtes en stage, on dispose d'un four à 5 niveaux de thermostat (numérotés 4,5,6,7,8) , et de 6 plats à faire cuire : un pavé de saumon, un gigot d'agneau piqué à l'ail, une tarte à la quetsche, un poulet en croûte de sel, un canard laqué et un gratin de légumes au parmesan. Le chef de cuisine à mis le four à préchauffer avec un certain thermostat, et votre programme doit, à partir de cette valeur, afficher le plat à mettre au four avec le temps de cuisson. Vous savez en outre que :

  Le saumon est cuit en 8 minutes au thermostat 4,

  Le gigot se cuit au thermostat 7 en 25 minutes, et à la même température, une tarte cuit en 35 minutes;

  Les volailles se cuisent à la température la plus chaude, le poulet en 40 minutes, le canard en une heure;

  Le thermostat 5 convient parfaitement pour les légumes, 15 minutes suffisent.

  Ecrivez le programme correspondant, bon appétit !

  Avec la structure de contrôle if else if else if ….

```
if (thermostat == 4)
  …..
else if (thermostat == 5)
   …..
else if (thermostat == 6)
          ….
else if (thermostat == 7)
       ….
else if (thermostat == 8)
       ….
else
    pas prévu
```

CORRIGE:

```
int main()
{
 // Test if…else if…else
    . . .
    int thermostat;
   cout << "thermostat du four: "; cin >> thermostat;

    switch (thermostat)
   {
      case 4:  cout << "Le saumon est cuit en 8 minutes" << endl; break;
      case 5: cout << "Les legumes se cuisent en au moins 15 minutes" << endl; break;
      case 6: cout << "Aucun plat pour cette valeur du thermostat" << endl;break;
      case 7: cout << "Le gigot se cuit en 25 minutes et une tarte cuit en 35 minutes;" << endl; break;
      case 8: cout << "Le poulet se cuit en 40 minutes et le canard en une heure;" << endl; break;
     // break  dans le cas par défaut
    // car SI JAMAIS IL Y AVAIT UN CASE APRES LE DEFAULT, il serait prise en compte et
     // et ce serait une erreur de logique
     default: cout << "Aucun plat pour cette valeur du thermostat" << endl;  break;
     case 3: cout << "…....." << endl; break; // on peut écrire d'autres cas après celui par défaut
   }
   return 0;
}
```

- On veut maintenant améliorer le programme concernant les services ADSL en proposant à l'abonné de choisir les services qu'il veut parmi ceux qui sont disponibles en considérant que son atténuation lui permet d'accéder à tous les services (cas où A <37). L'internet 5 Mb vaut 19,99 €/mois, la TV 4,99 € par mois, la téléphonie IP 6 € / mois. Ecrire un programme qui permet de choisir parmi les options disponibles et qui indique le montant à payer par mois.

CORRIGE

```
int main()
{
   int choix;

   cout << "CHOIX DES SERVICES" << endl;
   cout << " 1 Services ADSL: Internet 5 Mb; telephonie IP; TV" << endl;
   cout << " 2 Services ADSL: Internet 5 Mb; telephonie IP" << endl;
   cout << " 3 Services ADSL: Internet 2 Mb" << endl;
   cout << "votre choix: "; cin >> choix;

   switch(choix)
   {
      case 1: cout << "Tarif: " << (19.99 + 6 + 4.99)  << " euros/mois" << endl; break;
      case 2: cout << "Tarif: " << (19.99 + 6) << " euros/mois" << endl; break;
      case 3: cout << "Tarif: " << 19.99 << " euros/mois" << endl; break;
      default: cout << "choix invalide" << endl; break;
   }
   return 0;
}
```

- Enfin, le FAI (Fournisseur d'Accès Internet) propose les remises suivantes : 5% de remise pour un engagement de 12 mois, 12 % pour un engagement de 24 mois. Modifiez le programme pour que l'utilisateur saisisse le nombre de mois d'engagement, le programme indiquera : le montant de chaque mensualité, et le montant total sur toute la durée de l'engagement.

  Afin d'avoir 2 chiffres après la virguke pour les flottants, on spécifie cout << fixed << setprecision(2) avant leur affichage.

CORRIGE

```
#include <iostream>
#include <iomanip> // les manipulateurs du C++ cf fixed et setprecision
using namespace std;

int main()
{
   int choixServices, choixEngagement, nombreMois;
   float remise, tarif;

   cout << "CHOIX DES SERVICES" << endl;
   cout << " 1 \t Services ADSL: Internet 5 Mb; telephonie IP; TV" << endl;
   cout << " 2 \t Services ADSL: Internet 5 Mb; telephonie IP" << endl;
   cout << " 3 \t Services ADSL: Internet 2 Mb" << endl;
   cout << "votre choix: "; cin >> choixServices;
   cout << "DUREE D'ENGAGEMENT" << endl;
   cout << " 1 \t 12 mois remise 5%" << endl;
   cout << " 2 \t 24 mois remise 12%"<< endl;
   cout << " 3 \t sans engagement de durée" << endl;
   cout << "votre choix: "; cin >> choixEngagement;

   switch(choixEngagement)
   {
      case 1: nombreMois = 12; remise = 0.05; break;
      case 2: nombreMois = 24; remise = 0.12; break;
      case 3: nombreMois = 1; remise = 0.0; break;
      default: cout << "choix invalide" << endl; return 0; // sortie du programme
   }
```

```
    switch(choixServices)
    {
        case 1: tarif = (1-remise) * (19.99 + 6 + 4.99);
                cout << fixed << setprecision(2) << "Tarif: " << tarif << " euros/mois" << endl;
                 cout << "Tarif sur " << nombreMois << " mois: " << nombreMois * tarif << "
euros/mois" << endl;; break;
        case 2: tarif = (1-remise) * (19.99 + 6);
                cout << "Tarif: " << tarif << " euros/mois" << endl;
                cout << "Tarif sur " << nombreMois << " mois: " << nombreMois * tarif << "
euros/mois" << endl;; break;
        case 3: tarif = (1-remise) * 19.99;
                 cout << "Tarif: " << tarif << " euros/mois" << endl; break;
        default: cout << "choix invalide" << endl; break;
    }
    return 0;
}
```

THEME 3 : boucle while et do while

Saisie sécurisée : pour rendre les programmes plus fiables et sécurisés, on peut contrôler que les saisies sont conformes à ce qu'on attend avant de continuer le programme.

- Ecrire un programme qui calcule la racine carrée d'une valeur à virgule : il faut vérifier que la valeur saisie est positive avant de faire le calcul : on utilisera donc une boucle while ou do...while.

  Pour le calcul de la racine carrée, on utilisera la fonction sqrt().

  Attention à bien choisir la boucle : while, ou do…while ?

  CORRIGE

```
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
   float nombre;

   do
   {
      cout << "nombre: "; cin >> nombre;
   } while (nombre < 0);

   cout << "La racine carree de " << nombre << " vaut " << sqrt(nombre);
   return 0;
}
```

OU BIEN mais **mauvais style! CODE DUPLIQUE**

```
    cout << "nombre: "; cin >> nombre;
     while (nombre < 0)
   {
     cout << "nombre: "; cin >> nombre;
     }
```

- Ecrire un programme auquel on fournit un rang entier *p*, et qui calcule la valeur des suites suivantes au rang *p* :

  Suite *Un* : $\begin{cases} U_0 = 4{,}5 \\ U_{n+1} = 2.U_n - 3; \end{cases}$

  Suite *Vn* : $\begin{cases} V_0 = 6 \\ V_{n+1} = V_n/2 - 2.5; \end{cases}$

  Suites *Wn* et *Xn* : $\begin{cases} W_0 = 2; X_0 = -1 \\ W_{n+1} = W_n + X_n; \\ X_{n+1} = X_n - W_n; \end{cases}$

  Suite *Yn* : $\begin{cases} Y_0 = 4; Y_1 = 2 \\ Y_{n+1} = Y_n/Y_{n-1} + 1; \end{cases}$

  CORRIGE

```
#include <iostream>
using namespace std;

int main()
{
   int rang;
// saisie du rang
 do
   {
      cout << "rang des suites: "; cin >> rang;
   } while (rang < 0);

   // terme u0
   float u = 4.5;    // pas équivalent à float u; ……..…  u = 4.5;
   int n = 0;
   while (n < rang)
   {
      u = 2 * u – 3;     // un+1 = 2 un - 3
      n++; // équivalent à n = n + 1;
   }
   cout << "u" << rang << " = " << u << endl;
```

```
    float v = 6;
    i = 0;
    while (i < rang)
    {
       v = v/2 – 2.5;      // vn+1 = vn/2 – 2.5
       i++;
    }
    cout << "v" << rang << " = " << v << endl;

    float w = 2, x = -1, save;
    i = 0;
    while (i < rang)
    {
       save = w;           // sauvegarde de wn
       w = w + x;          // wn+1 = wn + xn
       x = x - save;       // xn+1 = xn -wn
       i++;
    }
    cout << "w" << rang << " = " << w << endl;
    cout << "x" << rang << " = " << x << endl;

   float y0 = 2, y1 = 4, y2 = 2;
    i = 0;
    while (i < rang)
    {
       y2 = y1/y0 + 1;
       y0 = y1;
       y1 = y2;
       i++;
    }
    cout << "y" << rang << " = " << y2 << endl;
    return 0;
}
```

- Le nombre mystère :

  On cherche à faire deviner à l'utilisateur un nombre, compris entre 1 et 10 000, choisi au hasard par l'ordinateur. L'utilisateur procède par essais successifs : à chaque essai, l'ordinateur indique si le nombre rentré est plus grand ou plus petit que le nombre mystère. Le jeu continue jusqu'à ce que l'utilisateur trouve le nombre mystère.

  Pour faire un tirage de nombre au hasard, on utilisera l'instruction : rand( ); cette instruction donne une valeur entière comprise entre 0 et un très grand entier nommé RAND_MAX. Vous pouvez utiliser l'opérateur % (modulo).

CORRIGE

```
#include <iostream>
#include <stdlib.h>     // srand, rand
#include <time.h>       // time

using namespace std;

int main()
{
   int mystere, nombre;

   srand(time(NULL)); // initialisation de l'aléatoire
   mystere = 1+ rand() % 10000;
   cout << "Recherche d'un nombre mystere entre 0 et 10 000" << endl;
do
   {
      do
      {
          cout << "nombre: "; cin >> nombre;
      } while (nombre < 0 || nombre > 10000) ;

      if(nombre < mystere)
      {
         cout << "Trop petit" << endl;
      }
      else if(nombre > mystere)
      {
                  cout << "Trop grand" << endl;
         }
   }
```

```
    while(nombre != mystere);
    cout << "Bravo, vous avez trouve le nombre mystere " << mystere << endl;

    return 0;
}
// OU BIEN
  do
   {
      cout << "nombre: ";  cin >> nombre;
      if(nombre < mystere)
      {
         cout << "Trop petit" << endl;
       }
       else if(nombre > mystere)
         cout << "Trop grand" << endl;
       else // inutile if (nombre == mystere) Principe du tiers-exclu
           cout << "Bravo, vous avez trouve le nombre mystere " << mystere << endl;
   }
   while(nombre != mystere);
```

- Méthode de newton : il existe une méthode simple pour calculer la racine carrée d'un nombre *x* à l'aide d'une suite :

  La suite : $U_0 = x$; $U_{n+1} = U_n/2 + x/(2.U_n)$ converge vers $\sqrt{x}$ (c'est à dire que la valeur de U_n se rapproche de plus en plus de $\sqrt{x}$ quand *n* augmente). Ecrire un programme qui calcule la racine d'un nombre *x* et qui s'arrête lorsque la **précision obtenue est de 10^{-8}.**

  CORRIGE

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

using namespace std;

int main()
{
   double x;
   cout << "RACINE CARREE D'UN NOMBRE POSITIF" << endl;
   do
   {
      cout << "nombre: "; cin >> x;
   } while (x <= 0);

   double u = x;
   do
   {
      u = u/2 + x/(2 * u);
      cout << "u = " << u << endl;
   } while ( fabs(u*u-x) > 1E-8);
   cout << "La racine carree a E-8 pres de " << x << " est " << u << endl;

   return 0;
}
```

- Ecrire un programme de gestion de compte d'épargne :

  On dépose un montant initial de *M* €, et tous deux mois, un montant *m*. Tous les 6 mois, les intérêts, à un taux *T*, sont versés sur le compte. Tous les ans, si le montant dépasse un seuil *S*, une taxe à un taux T_{taxe} est prélevé. Egalement tous les ans, des frais de gestion de compte, d'un montant de 28,75 €, sont prélevés.

  Ecrire un programme qui saisit le montant initial M et le montant du versement m, et qui affiche le solde du compte par périodes de deux mois. On affiche aussi les autres évènements : versement des intérêts, prélèvements.
  On arrête l'affichage : au bout de 4 ans, ou si le solde du compte atteint ou dépasse un plafond *P*.
  Données :
  Le montant *M* doit être de 15 € minimum, le montant *m* de 150 € minimum, le taux *T* est de 4,5 %, le taux T_{taxe} de 1,023 %. Le seuil *S* est de 6 000 €, le plafond *P* de 12 000 €.

  CORRIGE

```
#include <iostream>
#include <iomanip> // les manipulateurs du C++ cf fixed et setprecision
#include <math.h>
using namespace std;

int main()
{
   long i = 0;
   double M,m;
   double tauxSemestriel = .045;
   double tauxTaxe = .023;
   double fraisGestion = 28.75;
   double seuil = 6000;
   double plafond = 12000;

   cout << fixed << setprecision(2);
   cout << "Bienvenue dans votre gestion de compte d'epargne." << endl;
   cout << "Entrez votre montant de depart : ";

   do
```

```
{
   cout << "Une somme raisonnable (au moins 15 euros)!" << " Quel montant ? : ";
   cin >> M;
} while (M < 15);
cout << "Bien, vous disposez de : " << M << " euros sur votre compte." << endl;

cout << "Quel montant bimensuel souhaitez-vous déposer? : ";
do
{
   cout << "Déposez au moins 150 euros!" << " Quel montant ? : ";
   cin >> m;
} while (m < 150);
cout << "Votre déposez : " << m << " euros tous les 2 mois.";

do
{
   cout << "------------------------------" << endl;
   M = M + m;
   cout << "Rapport bimensuel NO : " << i/2 << endl;

   if ((i%6) == 0)
   {
      cout << "Vos interets vous ont rapporte : " << tauxSemestriel * M << " euros." << endl;
      M = (1 + tauxSemestriel) * M;
   }
   if ((i%12) == 0)
   {
      if (M > seuil)
      {
         cout << "Vous avez paye une surtaxe de : " << tauxTaxe * M  << " euros." << endl;
         M = (1 - tauxTaxe) * M;
      }
      M = M - fraisGestion;
      cout << "Frais de gestion annuels : " << fraisGestion << endl;
   }
   cout <<"Vous avez : " << M << "euros." << endl;

   i = i+2;
}while ((i == 48) || (M <= plafond));
return 0;
```

THEME 4 : boucle for

- Reprendre l'exercice sur les suites abordés dans le thème précédent : choisissez deux suites parmi celles proposées et modifiez le programme existant afin de le réaliser avec un boucle pour. Peut-on utiliser une boucle for à la méthode de Newton vue dans le thème précédent ?

CORRIGE

```
#include <iostream>

using namespace std;

int main()
{
   int rang;

   do
   {
      cout << "rang des suites: "; cin >> rang;
   } while (rang < 0);
```

float u = 4.5; int n; **for (n = 0; n < rang; n++)** { u = 2 * u – 3; }	**float u = 4.5;** int **n = 0;** **while (n < rang)** { **u = 2 * u – 3;** **n++;** }

```
   cout << "u" << rang << " = " << u << endl;
   float v = 6;
   for (i = 0; i < rang; i++)
   {
      v = v/2 - 2.5;
   }
   cout << "v" << rang << " = " << v << endl;
```

```
    float w = 2, x = -1, save;
    for (i = 0; i < rang; i++)
    {
       save = w; // sauvegarde de wn
       w = w + x; // wn+1 = wn + xn
       x = x - save; // xn+1 = xn -wn
    }
    cout << "w" << rang << " = " << w << endl;
    cout << "x" << rang << " = " << x << endl;

    float y0 = 2, y1 = 4, y2 = 2;
    for (i = 0; i < rang; i++)
    {
       y2 = y1/y0 + 1;
       y0 = y1;
       y1 = y2;
    }
    cout << "y" << rang << " = " << y2 << endl;
    return 0;
}
```

- Ecrire un programme qui calcule et affiche les carrés des 25 premiers entiers.

  CORRIGE

```
#include <iostream>
#define N 25 // macro instruction, N substitué par 25

using namespace std;

int main()
{
   int i;

   cout << "Carre des << N << premiers nombres entiers" << endl;
   for (i = 0; i < N; i++)
   {
      cout << i * i << endl;
   }
   return 0;
}
```

- Nombres premiers. Il existe une méthode simple (mais très peu efficace) permettant de déterminer les nombres premiers. Un nombre premier est un nombre qui n'est divisible que par 1 et par lui même (1 n'est pas un nombre premier selon cette définition). Pour déterminer si un nombre est premier, il suffit de voir s'il est divisible par un nombre qui est plus petit que sa racine carrée. S'il est divisible, il n'est pas premier.

  Ecrire un programme qui indique si un nombre entier saisi au clavier est un nombre premier.
  Indication sur la méthode : on utilisera une variable dont le seul but est de stocker l'information 'le nombre est premier'. On pourra utiliser un nombre entier qui vaudra 1 si le nombre est premier, et qui vaudra 0 sinon. Au départ, cette variable vaudra 1 (on suppose donc que le nombre est premier).

  CORRIGE

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

using namespace std;

int main()
{
   int nombre;
   int drapeau = 1; // au départ le nombre est supposé être premier
   int i;
  do
  {
            cout << "nombre: ";    cin >> nombre;
  } while (nombre < 0);
  double racine = sqrt(nombre);
```

for (i = 2; (i < racine) && (drapeau == 1); i++) { **if ((nombre%i) == 0)** { cout << "divisible par " << i << endl; **drapeau = 0;** } }	**for (i = 2; i < racine; i++)** { **if ((nombre%i) == 0)** { cout << "divisible par " << i << endl; **drapeau = 0;** **break;** } }

```
   if (drapeau == 1)
      cout << nombre << " est premier." << endl;
   else
      cout << nombre << " n'est pas premier." << endl;

   return 0;
} // fin du main
```

- A partir du programme précédent, faire un programme qui fait la liste des nombres de 1 à 1000, et indique si chacun de ces nombres est premier ou non.

CORRIGE

```
#include <iostream>

using namespace std;

int main()
{
   int nombre, drapeau;
   int i;

    cout << "\t\t\t\t 1 n'est pas premier." << endl;
     double racine = sqrt(nombre); // LE COMPILATEUR LE FERAIT TOUT SEUL
   // boucle for imbriquées
   for(nombre = 1; nombre < 1000; nombre++)
   {
      drapeau = 1; // au départ le nombre est suppose être premier
       for (i = 2; (i < racine) && (drapeau == 1); i++)
      {
         if ((nombre%i) == 0)
         {
            drapeau = 0;
         }
      } // fin de la seconde boucle for

      if (drapeau == 1)
         cout << "\t\t\t\t" << nombre << " est premier." << endl;
      else
         cout << nombre << " n'est pas premier." << endl;
   } // fin de la première boucle for

   return 0;
}
```

- Ecrire un programme qui affiche les lettres de l'alphabet, en majuscule, et leur code de la table ASCII, puis qui fait de même avec les minuscules.

CORRIGE

```
#include <stdio.h>

int main()
{
   int caractere;

   for(caractere = 'A'; caractere <= 'Z'; caractere = caractere + 1)
   {
      printf("caractere %c \t code Ascii %d\n", caractere, caractere);
   }
   printf("\n");
   for(caractere = 'a'; caractere <= 'z'; caractere += 1)
   {
      printf("caractere %c \t code Ascii %d\n", caractere, caractere);
   }
   return 0;
}
```

- Ecrire un programme qui saisit deux nombres à virgule, qui sont les deux bornes d'un intervalle, et un nombre entier : ce nombre entier est le nombre de points, régulièrement espacés, que l'on veut placer dans cet intervalle (par exemple, quand on veut faire le graphe d'une fonction avec un ordinateur, on demande, dans l'intervalle [-1,5; +2] de tracer 1000 points. Le programme devra afficher la coordonnée de chaque point.

CORRIGE

```
#include <iostream>

using namespace std;

int main()
{
   double borneInf, borneSup, pas;
   long a;

   cout << "borne inferieure "; cin >> borneInf;
   do
   {
        cout << "borne superieure "; cin >> borneSup;
   } while(borneSup <= borneInf);
    cout << "nombre : "; cin >> a;

   // calcul du pas
   pas = (borneSup - borneInf) / a;

   double i; // ATTENTION le compteur n'est pas un entier
   for ( i = borneInf; i <= borneSup; i = i + pas)
   {
         cout << i << endl;
   }

   return 0;
}
```