Séance 4 : Jeudi 14/02/2019

Traduction des exercices du TD3 : fonctions et procédures

1. Écrire l'algorithme d'un sous-programme qui retourne la moyenne de deux réels a et b donnés en paramètre. Écrire le programme principal qui utilise le sous-programme précédent et affiche le résultat produit.

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

// On va mettre tus les sous programmes à ecrire
// Exercice moyenne de 2 réels

float moyenne (float a, float b)
{
    float moy;
    moy = (a+b) / 2;
    return moy;
}

int main (void)
{   float v1,v2, m;
    //appel des différents sous programmes
    cout<<moyenne(4.56,7.88);
    cout<<"donnez deux réels "<<endl;
    cin>>v1>>v2;
    m=moyenne(v1,v2);
    cout<<m<<endl;
    return 0;
}
```

2. Écrire l'algorithme d'un sous-programme qui affiche les dix nombres suivants la valeur n donnée en paramètre. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

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

void dix_suivants (int n)
{
    int i;
    for (i=1;i<=10 ; i++) // for (i=n+1; i<=n+10 ; i++)
    {                     // {
        cout<<n+i<<" ";       //    cout<<i;
    }                     // }
}

int main (void)
{   int v;
    cout<<"donnez un entier "<<endl;
    cin>>v;
    dix_suivants (v);
    return 0;
}
```

3. Écrire l'algorithme d'un sous-programme qui demande à l'utilisateur et retourne au programme principal une valeur entière comprise entre 0 et 20. La saisie sera recommencée tant que la valeur choisie n'appartient pas à l'intervalle [0 ; 20].

```
#include <iostream>
using namespace std;
int saisie_valeur()
{
    int n;
    do
    {
        cout << "donnez un entrier entre 1 et 20"<<endl;
        cin>>n;
    }
    while (n<1 || n>20);
    return n;
}
int main (void)
{   int v;
    v=saisie_valeur();
    return 0;
}
```

4. Écrire l'algorithme d'un sous-programme qui calcule la somme des n premiers entiers. Rappel : $1 + 2 + 3 + \ldots + n = n(n+1) / 2$

```
#include <iostream>
using namespace std;
int somme (int n)
{
    return n*(n+1) / 2;
}
int main (void)
{
    int val;
    cout<<"donnez un entier"<<endl;
    cin>>val;
    cout<<somme(val);
    return 0;
}
```

Autre solution

```
#include <iostream>
using namespace std;
int somme (int n)
{   int v,i;
    v=0;
    for (i=1;i<=n;i++)
    {
        v=v+i;
    }
    return v;
}
int main (void)
{
    int val;
    cout<<"donnez un entier"<<endl;
    cin>>val;
    cout<<somme(val);
    return 0;
}
```

5. Un nombre parfait est un nombre naturel n non nul qui est égal à la somme de ses diviseurs stricts (n exclus).
   Exemple : 6 = 1 + 2 + 3
   1. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un nombre parfait, faux sinon.
   2. Écrire en langage algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.

```
#include <iostream>
using namespace std;
bool est_parfait (int n)
{
    int som,i;
    som=0;
    for (i=1;i<n;i++)
    {
        if (n%i==0) // i est un diviseur de n
            som = som + i ; // som+=i;
    }
    if (som == n)       // return (som == n);
        return true;
    else
        return false ;
}
int main (void)
{   int i;
    for (i=1;i<=10000;i++)
    {
            if (est_parfait(i)) //==true)
                cout<<i<<" est parfait"<<endl;
            //else cout<<i<<" n'est pas parfait"<<endl;
    }
    return 0;
}
```

6. Écrire l'algorithme d'un sous-programme qui dessine un carré de côté N à l'écran. L'utilisateur pourra choisir le caractère du contour du carré lors de l'appel du sous-programme.

Première version : en utilisant un sous-programme ligne_pleine et un sous-programme ligne_creuse.

```
#include <iostream>
using namespace std;
void ligne_pleine (int n)
{
    int i;
    for (i=1;i<=n;i++)
        cout<<"*";
    cout<<endl;
}
void ligne_creuse (int n)
{
    int i;
    cout<<"*";
    for (i=1;i<=n-2;i++)
        cout<<" ";
    cout<<"*";
    cout<<endl;
```

```
        }
        void dessin_carre (int n)
        {   int i;
            ligne_pleine(n);
            for (i=1;i<=n-2;i++)
                ligne_creuse(n);
            ligne_pleine(n);
        }
        int main (void)
        {
            dessin_carre(8);
            return 0;
}
```

Deuxième version en dessinant directement le carré

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

        void carre (int n)
        {   int i,j;
            for (i=1;i<=n;i++)
            {
                for (j=1;j<=n;j++)
                {
                    if (i==1 || i==n || j==1 || j==n)
                        cout<<"*";
                    else cout<<" ";
                }
                cout<<endl;
            }
        }

        int main (void)
        {
            carre(8);
            return 0;
}
```

Amélioration : en choisissant le caractère de contour et le caractère de l'intérieur

```
        #include <iostream>
        using namespace std;
        void carre (int n, char fond, char tour)
        {   int i,j;
            for (i=1;i<=n;i++)
            {
                for (j=1;j<=n;j++)
                {
                    if (i==1 || i==n || j==1 || j==n)
                        cout<<tour;
                    else cout<<fond;
                }
                cout<<endl;
            }
        }
        int main (void)
        {   int cote;
            char c1,c2;
            cout<<"Donnez la taille du carre"<<endl;
```

```
    cin>>cote;
    cout<<"Donnez le caractere de fond et le caractere de
contour"<<endl;
    cin>>c1>>c2;
    carre(cote,c1,c2);
    return 0;
}
```

7. Factorielle et son utilisation

a- Ecrire une permettant de calculer et de retourner la factorielle d'un nombre passé en paramètre. Utilisez ce sous-programme pour afficher les 15 premières valeurs des factorielles. Comparez les résultats de factorielle(13) et factorielle(14). Ces résultats vous semblent-ils cohérents et corrects ? Pourquoi ? Modifiez le type de retour de la fonction factorielle en "double" au lieu de "int" et observez les nouvelles valeurs obtenues.

```
#include <iostream>
using namespace std;
double factorielle (int n)
{
    double f,i;
    f=1;
    for (i=2;i<=n;i++)
        f*=i;

    return f;
}
int main (void)
{
    int i;
    for (i=1;i<=25;i++)
        cout<<i<<"! = "<<factorielle (i)<<endl;;
    return 0;
}
```

b- Ecrivez un sous-programme qui affiche le triangle de Pascal jusqu'à la ligne n en effectuant les étapes suivantes :

Ecrivez la fonction `int combinaison(int n, int p)` (qui utilise la fonction *factorielle*).

Rappel : $C_n^p = \frac{n!}{p!(n-p)!}$

Ecrivez la procédure `void trianglePascal(int n)` qui utilise la fonction combinaison pour afficher le triangle de rang n.

Exemple : trianglePascal(5); ➔

```
    1
  1   1
1   2   1
1 3   3   1
1   4   6   4   1
1   5 10 10   5   1
```

```
#include <iostream>
using namespace std;
int factorielle (int n)
```

```
{
    int f,i;
    f=1;
    for (i=2;i<=n;i++)
        f*=i;
    return f;
}
int combinaison (int n, int p) // p<=n
{
    return (factorielle(n)/ (factorielle(p)*factorielle(n-p)));
}
void triangle_pascal (int h)
{
    int i,j;
    for (i=0;i<=h;i++)
    {
        for (j=0;j<=i;j++)
        {
            cout<<combinaison(i,j)<<" ";
        }
        cout<<endl;
    }
}

int main (void)
{
    triangle_pascal(7);
    return 0;
}
```

8. L'exemple suivant permet de choisir aléatoirement une valeur comprise entre 0 et 29. La fonction `rand` retourne un entier aléatoire compris entre 0 et une constante RAND_MAX (32767).

```
#include <iostream>
#include  <time.h>     /*  necessaire  pour  l'initialisation

                                          avec srand */
#include <stdlib.h>    /* librairie contenant rand() */


using namespace std;

int main (void)
{
 int valea;
 srand(time(NULL)); /* une seule fois en début de programme */
 valea = rand()% 30;   /* a chaque fois qu'on veut une valeur */
 cout<<"la valeur aleatoire est : "<<valea<<endl;
 return 0;
}
```

- Modifiez le code précédent afin d'obtenir une valeur aléatoire comprise entre 1 et 30 puis entre 10 et 25.

```
rand() % 30 + 1
rand() % 16 + 10
```

a. Utilisez ce code pour écrire un sous-programme permettant de tirer aléatoirement une valeur et de la retourner au programme principal.

```
#include<iostream>
#include<time.h>
#include<stdlib.h>
using namespace std;
int tirage_aleatoire(int mod, int decal)
{
    return rand()% mod + decal;
}
int main (void)
{
    int valea;
    srand(time(NULL));
    valea =  tirage_aleatoire(16,10);
    cout<<"la valeur aleatoire est : "<<valea<<endl;
    return 0;
}
```

b. Ecrivez le programme principal permettant de faire deviner au joueur la valeur choisie par l'ordinateur. Il disposera de 5 essais pour trouver la valeur.

```
int main (void)
{
     srand(time(NULL)) ;
     int a_trouver,valeur, nb_essais;
     int max = 30 ;
     a_trouver=valeur_aleatoire();
     cout<<a_trouver;
     nb_essais=0;
     do
     {
                 cout<<"donnez une valeur";
                 cin>>valeur;
                 if(valeur>a_trouver)
                       cout<<"trop grand"<<endl;
                 else if (valeur < a_trouver) cout<<"trop
petit"<<endl;
                 nb_essais++;

     }
     while ((valeur!=a_trouver) && (nb_essais<5));
     if (valeur==a_trouver) cout<<"gagne en "<<nb_essais;
                                  else cout<<"perdu trop d essais";
     return (0);
}
```