LIFAP1 – TP4 : Fonctions et Procédures … suite

1. Ecrivez un sous-programme qui affiche le triangle de Pascal jusqu'à la ligne n en effectuant les étapes suivantes :
   a. Ecrivez la fonction `int combinaison(int n, int p)` (qui utilise la fonction *factorielle*).

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

```
int factorielle (int n)
{
  int i;
  int f=1;

  for (i=1;i<=n;i++)
    f*=i;
  return f;
}

int combinaison (int n , int p)
{
  return (factorielle(n)/(factorielle (p)*factorielle (n-p)));
}
```

   b. 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
```

```
void trianglePascal (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)
{
  int haut;
  cout<<"saisir la hauteur du triangle de pascal a afficher";
  cin>>haut;
  triangle(haut);
  return (0);
}
```

2. Écrivez un sous-programme `renverse` qui demande un entier `n` et affiche `n` en inversant l'ordre des chiffres. Par exemple, si l'utilisateur fournit n=123, le programme affichera n=321.

Indication : n modulo 10 (*n%10* en C) donne le dernier chiffre (celui de droite) ; la partie entière de n/10 donne tous les chiffres de gauche (sauf le dernier). Il suffit alors d'itérer. En C, si vous divisez deux entiers entre eux, vous obtenez la partie entière de la division.

Par exemple :

```
int n,i;

n=125;

i = n/10; /* i prend pour valeur 12 */
```

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

void renverse (int val)
{
 int i,reste;
 while (val>=10)
 {
    reste=val%10;
    val=val/10;
    cout<<reste;
 }
 cout<<val;
}

int main (void)
{
 int v;
 cout<<"donnez une valeur";
 cin>>v;
 renverse(v);
 return 0;
}
```

3. Ecrivez un programme permettant de dessiner le contour d'un carré en choisissant le caractère du contour. Pour cela, on effectuera les étapes suivantes :

a. Ecrivez une procédure `ligne_pleine` qui affiche n fois le caractère c sur une seule ligne (n et c étant donnés en paramètres)

```
void ligne_pleine(int n, char c)
{
  int i,j;

  for (i=0;i<n;i++)
  {
    cout<<c;
  }
  cout << endl;
}
```

```
**********
*        *
*        *
*        *
*        *
*        *
*        *
*        *
**********
```

b. Ecrivez une procédure `ligne_creuse` qui affiche le caractère c une fois en début de ligne et 1 fois en fin de ligne (n longueur totale de la ligne et c caractère étant donnés en paramètres)

```
void ligne_creuse(int n, char c)
{
  int i,j;

    cout<<c;
    for(j=1;j<n-1;j++)
```

```
        {
          cout<<" ";
        }
        cout<<c;
        cout<<endl ;
    }
```

c. Ecrire le sous-programme `affiche_carre` permettant d'afficher le contour d'un carré en utilisant les deux procédures précédentes. Exemple : afficherCarre(10, '*');

```
void affiche_carre(int n, char c)
{
  int i,j;

  ligne_pleine(n,c) ;
  for (i=1;i<n-1;i++)
  {
    igne_creuse(n,c);
  }
  ligne_pleine(n,c) ;

int main (void)
{

  affiche_carre(10,'*');
  return (0);
}
```

4. Reprendre et coder l'exercice 4 du TD3 (mini-jeu pour retrouver une valeur aléatoirement choisie)

```
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);
}
```

Pour aller plus loin

Transformez tous les exercices du TP2 pour faire des sous-programmes avec les différents motifs.