LIFAP1 – TP3 : Fonctions et Procédures

Objectifs : Le but de ce TP est de revoir et manipuler toutes les notions que vous avez vues jusqu'à maintenant. A la fin, vous devrez bien maîtriser :

- l'importance de la fonction main ;
- l'utilisation des procédures et des fonctions ;
- les boucles simples (tant que, faire) ;
- les boucles imbriquées ;
- la structure conditionnelle (si…alors…sinon).

1. Premiers programmes

   a. Fonction permettant de retourner le maximum de deux réels passés en paramètres.

```
float maximum (float a, float b)
{
  if (a>b) return a ;
else return b;
}
int main (void)
{
 float r1,r2;
 cout<<"donnez la premiere valeur"<<endl;
 cin>>r1;
  cout<<"donnez la deuxieme valeur"<<endl;
 cin>>r2;
 cout<<"le maximum est "<<maximum(r1,r2);
 return 0;
}
```

   b. Fonction 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.

```
int facto (int n)
{
   int i,res;
   i=1;
   res=1;
   while (i<=n)
   {
        res=res*i;
        i=i+1;
   }
   return res;
}
int main (void)
{
 int i;
 for(i=0;i<15;i++)
 {
   cout<<facto(i)<<endl;
 }
return 0;
}
```

```
Pour factorielle de 13 on obtient : 1932053504
Pour factorielle de 14 on obtient : 1278945280
  On constate bien qu'il n'y a pas un facteur 14 entre les deux valeurs !!!
```

On dépasse ici le domaine de définition de l'entier int. Avec double on a des résultats corrects pour ces deux valeurs (mais on débordera vite aussi !!)

c. Fonction permettant de calculer la somme des n premières puissances de 2. On devra utiliser pour cela la fonction `pow(x, y)` de la bibliothèque `math.h` qui calcule et retourne x^y.

```
#include<math.h>
int somme_puissances2(int n)
{
  int i, som=0;
  for(i=1;i<=n;i++)
  {
    som+=pow(2,i);
  }
  return som;
}
int main()
{
  int nb ;
  cout << " donnez la valeur de n";
  cin>>nb;
  cout <<"la somme des "<<nb<<" premieres puissances de 2 est :
"<<somme_puissances2(nb);
return 0;
}
```

2. 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>      /* 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;
}
```

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

```
valea=rand()% 30 + 1;
valea=rand()% 16 + 10;
```

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

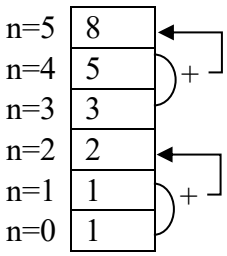
```
int valeur_aleatoire()
{
  return rand()% 30 + 1 ;
}
```

c. 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);
}
```

3. La suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence par fibonacci (0) = fibonacci (1) = 1.
Affichez le nième terme de la suite de Fibonacci (n étant passé en paramètre).




```
int fibonacci (int n)
{
  int i, un, un1, un2;

  un2 =1; // initialisation de u0
  un1 =1; // initialisation de u1
  un = 1 ; // pour le cas où n <2
  for(i=2;i<=n;i++)
  {
    un = un1 + un2 ;
    un2 = un1 ;
    un1 = un ;
  }
  return un ;

}
int main (void)
{

  cout <<fibonacci (7);
  return (0);
}
```