LIFAPI – 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. Maximum

   Ecrire une **fonction** `max_reel` permettant de retourner le maximum entre deux réels a et b passés en paramètres. En cas d'égalité, on renverra indifféremment l'une ou l'autre des deux valeurs.

```
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;
}
```

2. Factorielle

   Ecrire une **fonction** `factorielle` permettant de calculer et de retourner la valeur de la factorielle d'un nombre n passé en paramètre.

   Utilisez dans un premier temps ce soit programme pour afficher le résultat de la factorielle d'un nombre choisi par l'utilisateur.

```
int facto (int n)
{
    int i,res;
    res=1;
   for (i=1;i<=n;i++)
    {
          res=res*i;
    }
    return res;
}
int main (void)
{
 int val;
 cout<<"Donnez la valeur don't vous voulez la factorielle"<<endl;
 cin>>val
 cout<<"la factorielle de " <<val<<" est : "<<facto(val)<<endl;
return 0;
}
```

   Transformez ensuite le programme principal (et uniquement celui-ci) 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 !!)
```

3. Somme puissances

Ecrire une fonction `somme_puissances` permettant de calculer la somme des n premières puissances de 2. Par exemple si n= 4 onc calculera $2^0+2^1+2^2+2^3$

On devra utiliser pour cela la fonction `pow(x, y)` de la bibliothèque `math.h` qui calcule et retourne x^y. Par exemple 2^i s'écrira `pow(2,i)`.

```
#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;
}
```

4. Valeurs aléatoires

   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. En vous aidant du TD3, 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 `alea_intervalle` permettant de tirer aléatoirement une valeur comprise entre deux bornes a et b passées en paramètres et de la retourner au programme principal.

```
int valeur_aleatoire(int a, int b)
{
  return rand()% (b-a+1) + a ;
}
```

   c. Ecrivez le programme principal d'utiliser cette fonction.

```
int main (void)
{
 int val, bi, bs;
 srand(time(NULL)); /* une seule fois en début de programme */
 cout<< »donnez les bornes de l'intervalle »<<endl ;
cin>>bi>>bs ;
 val = valeur_aleatoire(bi,bs)
 cout<<"la valeur aleatoire est : "<<val<<endl;
 return 0;
}
```

5. Dessin

   On veut écrire un programme permettant de dessiner le contour d'un carré en choisissant le caractère du contour et sa taille. 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)

   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)

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

   d. Utilisez les sous-programmes précédents

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

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

  for (i=0;i<n;i++)
  {
    cout<<c;
  }
  cout << endl;
}
void ligne_creuse(int n, char c)
{
  int i,j;

    cout<<c;
    for(j=1;j<n-1;j++)
    {
      cout<<" ";
    }
    cout<<c;
    cout<<endl ;
}
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);
}
```

Pour aller plus loin

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