LIFAP1 – TP2 : 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

Utilisez vos notes des TD 2 et 3 afin de rédiger les programmes suivants :

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>     /*   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.

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

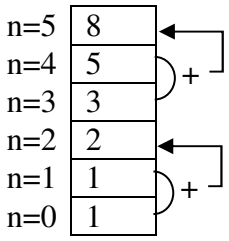
a. Utilisez ce code pour écrire un sous-programme permettant de tirer aléatoirement une valeur et de la retourner au programme principal.

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

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);
}
```

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,a,b,c;

  b =1;
  c =1;
  for(i=3;i<=n;i++)
  {
    a =b;
    b =c;
    c =a+b;
  }
  return c;

}
int main (void)
{

  cout <<fibonacci (7);
```

```
  return (0);
}
```

4. 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);
}
```

5. Écrivez un programme `renverse.cpp` 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;
}
```

6. 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 ;
  }
```

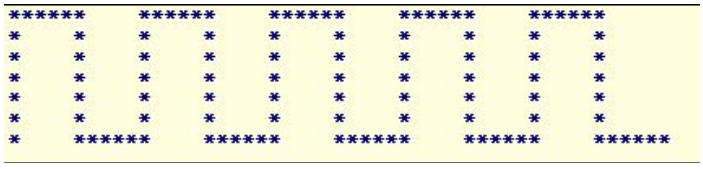
c. Ecrire le sous-programme `affiche_carre` permettant d'afficher le contour d'un carré en utilisant les deux procédures précédentes. Exemple : afficherCarre(10, '*');

```
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);
}
```

7. Écrivez une procédure qui affiche une frise.

```
procedure afficherFrise(n,l,h : donnée Entier)
// n est le nombre de fois que se repete le motif
// l est la demi longueur d'un motif
// h est la hauteur d'un motif
```

Exemple :
afficherFrise(5, 6, 7);



```
void aff_frise(int n, int l, int h)
{
  int i,j;
  char e=' ';
  //~ aff_ncar(1,'*');
  for(i=0;i<n;i++)
```

```
    {
      aff_ncar(l,'*');
      aff_ncar(l-2,e);
    }
    cout<<endl;

    for(i=1;i<h-1;i++)
    {
      for(j=0;j<n;j++)
      {
        aff_ncar(1,'*');
        aff_ncar(l-2,e);
        aff_ncar(1,'*');
        aff_ncar(l-2,e);
      }
      cout<<endl;
    }

    aff_ncar(1,'*');
    for(i=0;i<n;i++)
    {
      aff_ncar(l-2,e);
      aff_ncar(l,'*');
    }
    cout<<endl;

}
```