Séance 7 : Jeudi 28/02/2019

Fin des exercices du TP2 : fonctions et procédures

1. 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.

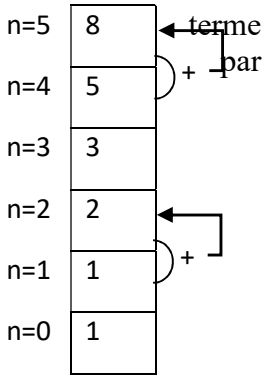
```
#include<iostream>
#include<math.h>
using namespace std;

int SommePuissanceDeux (int n)
{
    int i,som = 0 ;
    // calcul de la somme en fonction du paramètre n
    for (i=0;i<n;i++)
    {
        som = som + pow (2,i);// som += pow(2,i);
    }
    return som;
}

int main (void)
{
    int val;

    // saisie des informations et appel de la fonction écrite
précédemment
    cout<<"donnez un entier positif"<<endl;
    cin>>val;
    cout<<SommePuissanceDeux(val)<<endl;
    return 0;
}
```

2. 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).




```
#include<iostream>
#include<math.h>
using namespace std;

int Fibonacci (int n)
{
   int i,f,fn1,fn2;
   fn1 = 1;
   fn2 = 1 ;
   for (i= 2 ; i<=n ; i++)
   {
       f=fn1 + fn2 ;
       fn2 = fn1;
       fn1 = f;
```

```
    }

    return f;
}

int main (void)
{
    int val;

    // saisie des informations et appel de la fonction écrite
précédemment
    cout<<"donnez un entier positif"<<endl;
    cin>>val;
    cout<<Fibonacci(val)<<endl;
    return 0;
}
```

3. 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 n)
{
  while (n!=0)
  {
      cout<<n%10;
      n/=10;
  }
}

int main (void)
{
    int val;
    cout<<"Donnez un entier "<<endl;
    cin>>val;
    renverse(val);
    return 0;
}
```

CM4 : Passage de paramètres

Exercice polynomes du CM4

```
#include<iostream>
#include<math.h>
using namespace std;
 void racines (float a, float b, float c, int & nb_racines, float
&r1, float &r2)
{
    float delta;
    delta = b*b - (4*a*c);
    cout<<"valeur du discriminant :"<<delta<<endl;
    if (delta <0)
    {
        nb_racines = 0;
    }
    else if (delta == 0)
    {
        nb_racines = 1;
        r1 = (-b) / (2*a);
        r2 = r1 ;
    }
    else
    {
        nb_racines = 2;
        r1 = ((-b)+ (sqrt(delta))) / (2*a);
        r2 = ((-b)- (sqrt(delta))) / (2*a) ;
    }
}

int main (void)
{
    float c1,c2,c3, rac1,rac2;
    int nbr;
    c1=1;
    c2=1;
    c3=1;
    racines(c1,c2,c3,nbr,rac1, rac2);
    if (nbr == 0) cout <<"Pas de racines reelles"<<endl;
    else if (nbr==1) cout<<"une racine double "<<rac1<<endl;
    else cout<<"Deux racines distinctes "<<rac1<< " et
"<<rac2<<endl;
    return 0;
}
```

Exercices du TP3

1. Kezako (Compréhension : fonction/procédure, paramètres et appels)

1. Que fait le programme ci-dessous ? Réfléchissez et écrivez le texte que vous pensez voir apparaître à l'exécution de ce programme :

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

void proc_mult(int a, int b, int& ab)
     {
```

```
        cout << "execution de la procedure proc_mult" << endl;
        ab = a*b;
}
int fonc_mult(int a, int b)
     {  cout << "execution de la fonction fonc_mult" << endl;
        return a*b;
}
void kezako(int x, int y, int r1, int& r2)
{
        proc_mult( x, y, r1);
   r2 = fonc_mult(x,y);
        cout << "A la fin de kezako r1=" << r1 << " r2=" << r2
<<endl;
}

int main(void)
{  int a, y, res1, res2;
   a = 5;       y = 4;      res1 = 0;       res2 = 1;
        cout << "Dans main avant kezako res1=" << res1 << " res2="
<< res2  <<endl;
        kezako( a, y, res1, res2);
        cout << "Dans main apres kezako res1=" << res1 << " res2="<<
res2 << endl;
        return 0;
}
```

2. Téléchargez le programme Kezako à partir de la page du cours afin de vérifier si votre intuition est correcte. Sinon demandez une explication à votre encadrant de TP.

```
      #include <iostream>
      Using namespace std;
      void proc_mult(int a, int b, int& ab)
      {
5        cout << "execution de la procedure proc_mult" << endl;
6        ab = a*b;
}
int fonc_mult(int a, int b)
8     {  cout << "execution de la fonction fonc_mult" << endl;
9        return a*b;
}
void kezako(int x, int y, int r1, int& r2)
{
4        proc_mult( x, y, r1);
7 9      r2 = fonc_mult(x,y);
10       cout << "A la fin de kezako r1=" << r1 << " r2=" << r2
<<endl;
}
1     int main(void)
{  int a, y, res1, res2;
   a = 5;       y = 4;      res1 = 0;       res2 = 1;
2        cout << "Dans main avant kezako res1=" << res1 << " res2="
<< res2  <<endl;
3        kezako( a, y, res1, res2);
11            cout << "Dans main apres kezako res1=" << res1 << "
res2="<< res2 << endl;
12       return 0;
}
```

Exécution du programme :

1 Le programme commence toujours par la fonction main. Les variables sont crées et prennent leur valeur initiale. a=5, y=4, res1=0, res2=1

2 Le programme affiche la première ligne :

Dans main avant kezako res1=0 res2=1

3 Appel à la fonction kezako en passant a pour x(x=5),y pour y(y=4),res1 pour r1(r1=0) et res2 pour r2(***r2=res2=1***). Le passage par référence se fait entre res2 et r2.

4 Appel à la fonction proc_mult en passant x pour a(a=5), y pour b(b=4),r1 pour ab (***ab=r1=0***). Le passage par référence se fait entre r1 et ab.

5 Dans proc_mult, une ligne est affichée :

execution de la procedure proc_mult

6 Dans proc_mult, ab prend pour valeur a*b (5*4=20), ab=20. A la fin de proc_mult, ab=20 donc r1=20.

7 Dans kezako, appel à la fonction fonc_mult en passant x pour a (a=5) et y pour b(b=4)

8 Dans fonc_mult, une ligne est affichée :

execution de la fonction fonc_mult

9 fonc_mult renvoie a*b=5*4=20. Dans kezako, r2 prend pour valeur 20.

10 Dans kezako, une ligne est affiché :

A la fin de kezako r1= 20 r2= 20

A la fin de kezako, r2=20, donc res2 = 20

11 Dans main, la dernière ligne est affichée :

Dans main apres kezako res1=0 res2=20

res1 reste inchangé mais res2 a comme valeur 20.

12 Fin de programme.

2. Passage de paramètres. Ecrire pour chaque exercice le sous-programme demandé ainsi que le programme principal permettant de le tester.
   a. Permutation circulaire de 3 variables

```
#include <iostream>
using namespace std;void permutation_ciculaire(int &a, int &b, int
&c) // 3 paramètres
{
    int t;
    t=a;
    a=c;
    c=b;
    b=t;
}
int main(void)
{
    //utilisation de la procédure pour vérifier
    int v1,v2,v3;
    cout<<"Donnez 3 entiers"<<endl;
    cin>>v1>>v2>>v3;
    cout<<"avant permutation : v1 = "<<v1<<" v2= "<<v2<<" et v3 =
"<<v3<<endl;
    permutation_ciculaire(v1,v2,v3);
    cout<<"apres permutation : v1 = "<<v1<<" v2= "<<v2<<" et v3 =
"<<v3<<endl;
    return 0;
}
```

b. Nombre de combinaisons (fonction **ET** procédure)

```
#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 combinaison_proc (int n, int p, int &cnp) // p<=n
{
     cnp=(factorielle(n)/ (factorielle(p)*factorielle(n-p)));
}
int main (void)
{
        int res ;
        combinaison_proc(6,4, res);
        cout<<res<<endl;;
        cout<<combinaison(6,4);

    return 0;
}
```

c. Écrivez l'algorithme d'une procédure permettant d'effectuer la division euclidienne de deux entiers a et b. On appellera q le quotient et r le reste de cette division. On rappelle la formule de la division : a = b*q + r, avec r < b.

```
#include <iostream>
using namespace std;
void division_euclidienne (int a, int b, int &q, int &r)
{
    //on suppose que b est non nul
    q = a / b;
    r = a % b ;
}
int main (void)
{
    int val1,val2, quo, rest ;
    cout<<"donnez la numérateur"<<endl;
    cin>>val1;
    do
    {
        cout<<"Donnez le dénomminateur "<<endl;
        cin>>val2;
    } while (val2 ==0);

    division_euclidienne(val1,val2,quo,rest);
    cout<<quo<<" et "<<rest;
    return 0;
}
```

d. Écrivez l'algorithme d'une fonction perimetre_cercle permettant de retourner le périmètre d'un cercle en fonction de son rayon (passé en paramètre). Écrivez ensuite une fonction aire_cercle qui retourne l'aire d'un cercle. On souhaite maintenant écrire un sous-programme (qui utilise les deux fonctions précédentes) permettant à partir du rayon d'un cercle de calculer son périmètre et sa surface. Écrivez l'entête de ce sous programme de deux manières différentes.

```
#include <iostream>
#include<math.h>
using namespace std;

float perimetre_cercle(float r)
{
    return 2*M_PI * r;
}

float surface_cercle(float r)
{
    return M_PI * r *r ;
}

void perimetre_surface_proc (float r, float &p, float &s)
{
    p = perimetre_cercle(r);
    s = surface_cercle(r);
}

float perimetre_surface_fonc (float r, float &p)
{
    p= perimetre_cercle(r);
    return surface_cercle(r);
}

int main (void)
{
    float surf, peri;
    perimetre_surface_proc(4,peri,surf);
    cout<<"perimetre : "<<peri<<" et surface "<<surf<<endl;
    surf = perimetre_surface_fonc(4,peri);
    cout<<"perimetre : "<<peri<<" et surface "<<surf<<endl;

    return 0;
}
```