INFO-H-100 - Programmation
TP 10 - Exercices de synthèse
Corrections

Juin 2009

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

const double EPS = 1.0e-4;
const int BORNE = 1000;

int euler(int n);
double sech(double x);

double sech(double x)
{
  int signe = 1;
  double terme = 1;
  double somme = 1;
  double x2 = x*x;
  double x2N = 1;
  int fact2N = 1;

  for(int i=1; i<BORNE && terme >= eps; i++)
  {
    signe = - signe;
    x2N = x2N*x2;
    fact2N = fact2N*((2*i)-1)*(2*i);

    terme = ((euler(2*i)/fact2N)*x2N);
    somme += terme*signe;
  }
  return somme;
}

int main()
{
  int x;
  cin >> x;
  cout << sech(x);
  return 0;
}
```

Juin 2004

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

```
const int TAILLE_POL = 15;

void derivKieme(int[], int&, int);


int main ()
{
        int pol[TAILLE_POL];
        int deg = 3;
        pol[0] = 2;
        pol[1] = 3;
        pol[2] = 4;
        pol[3] = 5; // 2 + 3x + 4x^2 + 5x^3
        derivKieme(pol,deg,2); //8+30x;
        cout << pol[0] << " + "<< pol[1] << "*x" <<endl;

        return 0;
}

void derivKieme(int P[], int &degre, int k)
{
        degre = degre-k; // degré du polynôme dérivé

        int coef=1; //(k+i)!/i!
        for(int i = k; i>1;i--) coef*=i;//initialisation à k!

        for(int i=0;i<=degre;i++)
        {
                P[i] = P[k+i] * coef;
                coef *= (k+i+1)/(i+1); //preparation pour i=i+1
        }
}
```

Septembre 2004

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

const int TAILLE_POL = 15;

void derive(double P[], double dP[], int degreP);
double valeur(double P[], int degre, double x);
double newton(int P[], int degre, double eps);

int main ()
{
        int pol[TAILLE_POL];
        int deg = 3;
        pol[0] = 2;
        pol[1] = 3;
        pol[2] = 4;
        pol[3] = 5; // 2 + 3x + 4x^2 + 5x^3
        cout << newton(pol, deg, 1.0e-4) << endl;

        return 0;
}


void derive(int P[], int dP[], int degreP)
{
        for(int i=0; i<=degreP-1 ; i++)
                dP[i] = (i+1)*P[i+1];
}

double valeur(int P[], int degre, double x)
{
        double xn = 1;
        double result = 0;
        for(int i = 0; i<=degre; i++)
        {
                result += P[i]*xn;
                xn = xn*x;
        }
        return result;
}

double newton(int P[], int degre, double eps)
{
        double rk=0;
        int dP[TAILLE_POL];
        double ancienTerme = 1;
        derive(P,dP,degre);
        while(ancienTerme - rk > eps)
        {
                ancienTerme = rk;
                rk -= valeur(P,degre,rk)/valeur(dP,degre-1,rk);
        }
        return rk;
}
```