Feuille 2

TD de Cpp

Exercice 1 :

Proposez un programme consitué de plusieurs fonctions :

- `init(int n, int T[])` qui initialise un tableau d'entiers de taille n avec des entiers compris entre 0 et 1000 ;
- `affiche(int n, int T[])` qui affiche un tableau d'entiers de taille n ;
- `tri(int n, int T[])` qui trie un tableau d'entiers de taille n. Vous pourrez choisir le tri bulle, le tri par insertion ou le tri rapide comme vous voulez !

Solution de l'exercice 1 :

```cpp
#include <iostream>
#include <stdlib.h>
#include <time.h>

const int N=25, val_max=25;

void affiche(int n, int T[]){
    for (int k=0; k<n; k++){
        std::cout << T[k] << "␣";
    }
    std::cout << std::endl;
}

void init(int n, int T[]){
    std::srand((unsigned) std::time(NULL));
    double r=1.*val_max/RAND_MAX;
    for (int k=0; k<n; k++){
        T[k] = std::rand()*r;
    }
}

void tri(int n, int T[]){
    int issorted=0;
    while (issorted == 0) {
        issorted=1;
        for (int k=0; k<n-1; k++) {
            if (T[k] > T[k+1]) {
                int bulle = T[k];
                T[k] = T[k+1];
                T[k+1] = bulle;
                issorted=0;
            }
```

```
        }
    }
}

int main(){
    int T[N];
    init(N, T);
    affiche(N, T);
    tri(N, T);
    affiche(N, T);
    return 0;
}
```

Exercice 2 :

Proposez un programme qui calcule $\sqrt{2}$ en résolvant $x^2 = 2$ par la méthode de la dichotomie.

Solution de l'exercice 2 :

```
#include <iostream>
#include <math.h>

const int precision=32;
double tol=1.e-16;

void affiche(double x) {
    std::cout << (int) x << ".";
    for (int i=0; i<precision; i++) {
        x -= (int) x;
        x *= 10;
        std::cout << (int) x;
    }
}

double f(double x) {
    return x*x - 2;
}

double dichotomy(double (*phi)(double), double a, double b, double tol) {
    double fa=phi(a), fb=phi(b);
    if (fa*fb>0) {
        std::cout << "Error in dichotomy: wrong initialization!" << std::endl;
        return 0;
    }
    double c, fc;
    int n=0, nmax=1000;
    while (std::fabs(b-a)>2*tol && n<nmax) {
        n++;
        c=.5*(a+b);
        fc=phi(c);
        if (fa*fc>=0) {
            a=c;
            fa=fc;
        }
```

```
        if (fb*fc>=0) {
            b=c;
            fb=fc;
        }
    }
    if (n==nmax)
        std::cout << "Warning: dichotomy did not converge!" << std::endl;
    return .5*(a+b);
}

int main() {
    std::cout << "Solve f(x)=0 with the dichotomy method" << std::endl;
    std::cout << "  to find the value of sqrt(2)!" << std::endl;
    std::cout << "  f(x) = x*x-2" << std::endl;
    //std::cout << std::setprecision(32) << std::sqrt(2) << std::endl;
    std::cout << "The exact solution is   ";
    affiche(std::sqrt(2));
    std::cout << std::endl;
    double x=dichotomy(f, 0, 2, tol);
    std::cout << "The numeric solution is ";
    affiche(x);
    std::cout << std::endl;
    std::cout << "Error                   ";
    affiche(std::sqrt(2)-x);
    std::cout << std::endl;
    return 0;
}
```

Exercice 3 :

Proposez une classe `Vector` qui permet de travailler avec des vecteurs comme vous en avez envie !