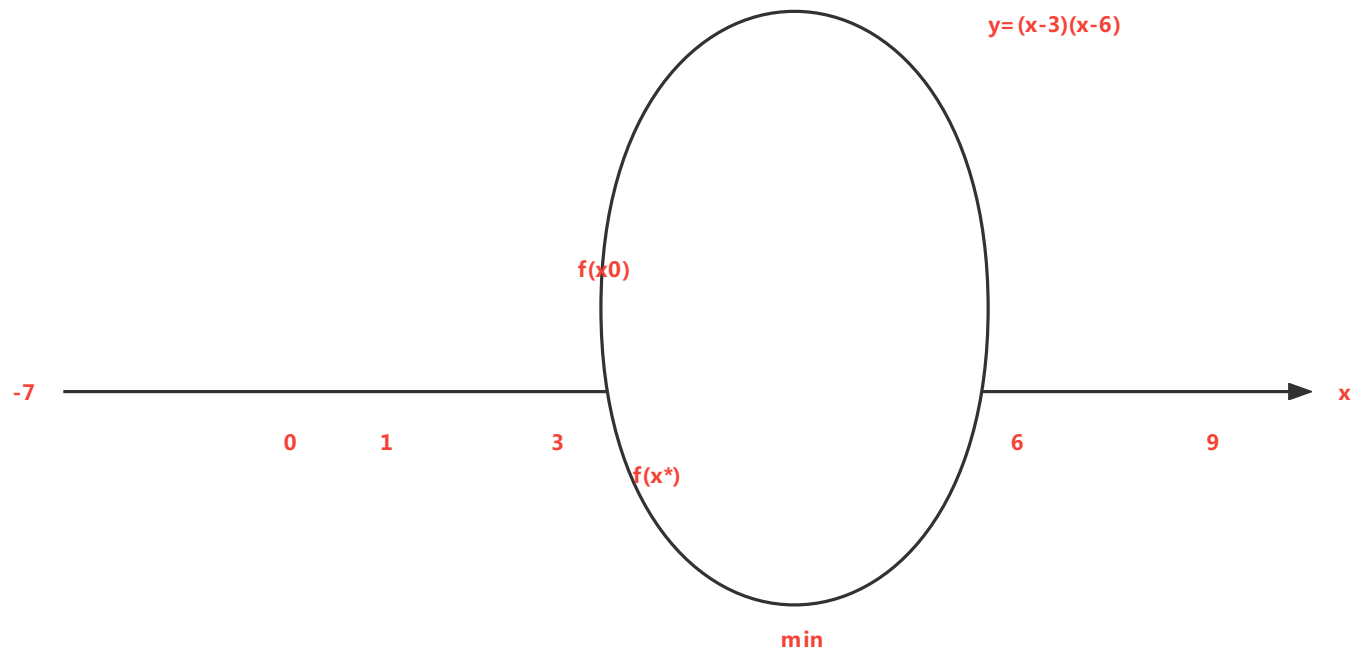
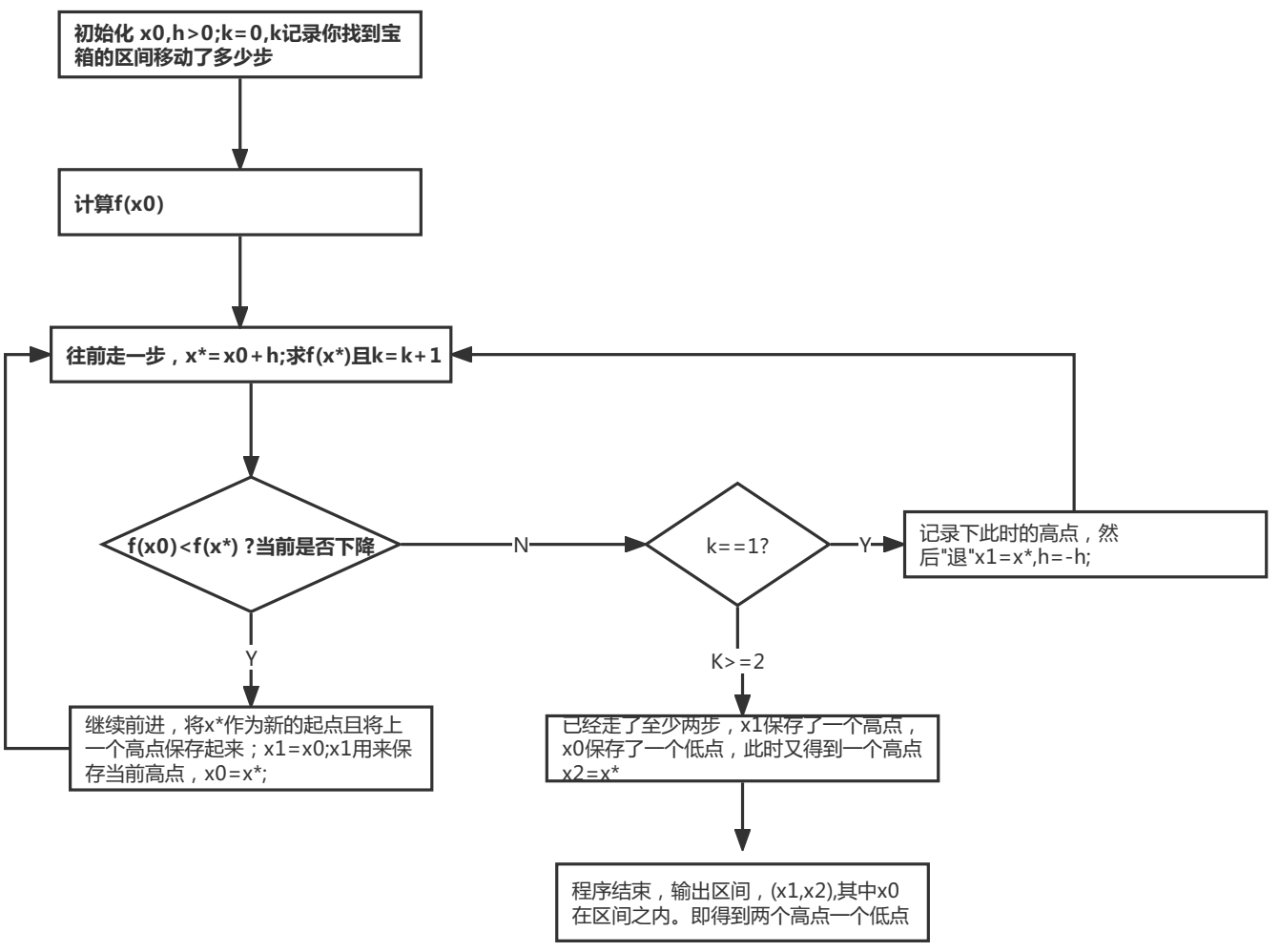
朴素一维搜索算法

最简单的一维搜索算法

目的： 假设有一个宝藏在两个山谷之间的最低处，它埋在地下，你该如何确定它的位置？假设你下坡的步数模型为x*=x+h,x:你的前一足迹，x*你的当前足迹，h你移动的步长。



移动公式：x*=x0+h,x0初始点，x*你移动后的位置




```
/*
 * @author: MENG ZHEN CHUAN
 * @date: do not edit
 * @description: HADOOP Project
 * @method: note
 */
#include <iostream>
#include <algorithm>
using std::swap;


class OneDemensionSearch{
    private:
        static double x0;/**高点*/
        static double x1;/**低点*/
        static double x2;/**高点*/
        static int h;/**前进或者后退的步数*/
        static double oldFuncationValue;/**之前的函数值*/
        static double newFuncationValue;/**新的函数值*/
        static size_t k;/**找到最小值区间的次数*/
    public:
        OneDemensionSearch(/* args */);
        ~OneDemensionSearch();
    public:
        static double funcation(double x);
        static double findMinValueInterval();
    public:
        static void getMemberInfo(){
            std::cout<<"初始化的x0  "<<x0<<std::endl;
            std::cout<<"初始化的步长  "<<h<<std::endl;
        }
        static void getIntervel(){
            std::cout<<"最小值所在的区间为 "<<std::endl;
            if(x1<x2){
                std::cout<<"("<<x1<<","<<x2<<")";
            }
            else{
                std::cout<<"("<<x2<<","<<x1<<")";
            }

        }

};
/**Outside class initialization of static variables,
* private can be called directly from the class name when initialized
*/

double OneDemensionSearch::x0=1;
double OneDemensionSearch::x1=0;
double OneDemensionSearch::x2=0;
int    OneDemensionSearch::h=8;
size_t OneDemensionSearch::k=0;
double OneDemensionSearch::oldFuncationValue=OneDemensionSearch::funcation(x0);
double OneDemensionSearch::newFuncationValue=0;

OneDemensionSearch::OneDemensionSearch(/* args */){
}

OneDemensionSearch::~OneDemensionSearch(){
}

/**具体的function*/
double OneDemensionSearch::funcation(double x){
    return (x-3)*(x-6);
}


/**
  * If the input function does not have a minimum value,
  * it will cause an infinite loop, which is a disaster.
  * Make sure the input function has a minimum value
*/
double OneDemensionSearch::findMinValueInterval(){
    double xStar=0;
    while (true){
        ++k;/*往前或者往后走一步*/
        xStar=x0+h;
        newFuncationValue=funcation(xStar);
        if(newFuncationValue<oldFuncationValue){
            x1=x0;/**将一个高点存储起来*/
            x0=xStar;/**更新x0*/
           // h=1.5*h;/**更新步长*/
            oldFuncationValue=newFuncationValue;
        }
        else{
            /**第一步反向，更新其方向*/
            if(k==1){
                x1=xStar;/**将一个高点存储起来*/
                h=-h;
            }
            /**在一高一低之后找到第二个高点*/
           if(k>=2){
            /**终止条件，找到第二个高点*/
                x2=xStar;
                break;
            }
        }

    }
}

int main(int argc, char const *argv[]){
    OneDemensionSearch::getMemberInfo();
    OneDemensionSearch::findMinValueInterval();
    OneDemensionSearch::getIntervel();
    return 0;
}
```