第六周上机

窦闻

2025-3-27

1. 上机指导书8.1, 追加要求: 编写函数对Point指针数组排序, 排序依据分别为 (1) 点到原点的距离, 升序排列; (2) 点坐标的字典序, 即先比较x的大小, 再比较y的大小, 升序排列. 排序函数的函数签名为 void SortPts(Point \*pPts[], int N, comp f); 其中函数指针类型别名定义为 using comp = bool (\*)(Point \*, Point \*);

#include <iostream>  
#include <cmath>  
using namespace std;  
   
struct Point  
{  
 double x;  
 double y;  
};  
   
// 定义比较函数的类型  
// typedef bool (comp)(Point \*, Point \*);  
using comp = bool (Point \*, Point \*);  
   
Point Input(void);  
void Output(Point p);  
void PrintPts(Point \*pPts[], int N); //调用Output输出点数组  
   
double Dist(Point &p1, Point &p2);   
void \_swap(Point \*&p1, Point \*&p2); //交换两个Point指针变量  
   
//根据排序依据f对点数组排序  
void SortPts(Point \*pPts[], int N, comp \*f);  
   
// 排序依据  
bool CompareByDistance(Point \*a, Point \*b); // 根据点到原点的距离比较, a距离原点更近时返回true  
bool CompareLexicographically(Point \*a, Point \*b); //按字典序比较点, a按字典序在b之前时返回true  
bool CompareBySlope(Point \*a, Point \*b); // 根据点相对于原点的斜率比较, a斜率较小时返回true. (选做)  
   
int main()  
{  
 Point pt1, pt2, pt3;  
 double dist;  
   
 // 输入两个点  
 cout << "Input Point 1." << endl;  
 pt1 = Input();  
 cout << "Input Point 2." << endl;  
 pt2 = Input();  
   
 // 计算p1, p2 之间的距离并输出  
 dist = Dist(pt1, pt2);  
 cout << "Point 1 is at ";  
 Output(pt1);  
 cout << "Point 2 is at ";  
 Output(pt2);  
 cout << "Distance between Pt 1 and Pt 2 is " << dist << endl;  
   
 // 对Point指针数组进行排序  
 const int N = 6;  
 Point Pts[N]{{2, 4}, {3, 3}, {1, 2}, {5, 1}, {1, 1}, {3, 4}};  
 Point \*pPts[N];  
 for (int i = 0; i < N;++i)  
 pPts[i] = &Pts[i];  
   
 cout << "按距离排序: " << endl;  
 SortPts(pPts, N, CompareByDistance);   
 PrintPts(pPts, N);  
 cout << "按字典序: " << endl;  
 SortPts(pPts, N, CompareLexicographically);   
 PrintPts(pPts, N);  
   
 //按斜率排序(选做)  
 // cout << "按斜率排序:" << endl;  
 // SortPts(pPts, N, CompareBySlope);   
 // PrintPts(pPts, N);  
   
 return 0;  
}

预期输出

按距离排序：  
POINT(1 1)  
POINT(1 2)  
POINT(3 3)  
POINT(2 4)  
POINT(3 4)  
POINT(5 1)  
按字典序:  
POINT(1 1)  
POINT(1 2)  
POINT(2 4)  
POINT(3 3)  
POINT(3 4)  
POINT(5 1)  
按斜率排序:  
POINT(5 1)  
POINT(1 1)  
POINT(3 3)  
POINT(3 4)  
POINT(2 4)  
POINT(1 2)