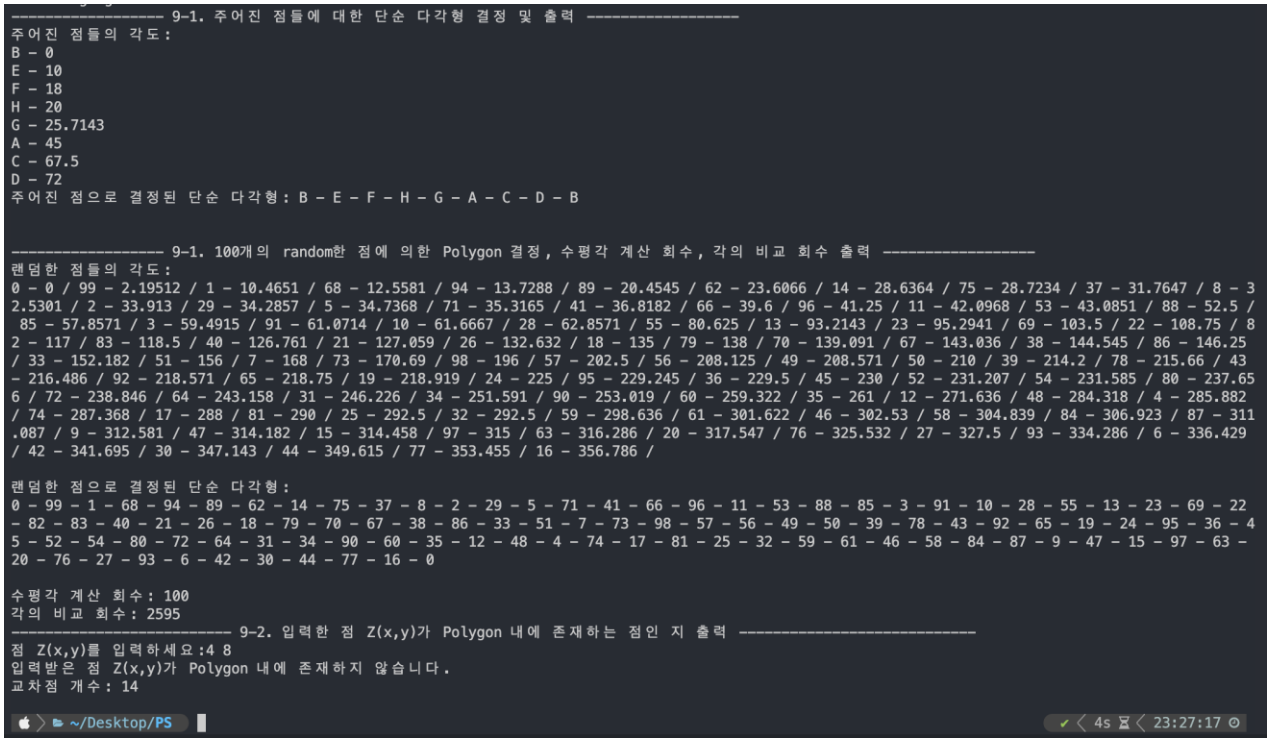
컴퓨터알고리즘과실습 실습9

2020112046 이재원

실행 결과



ComputeAngle()로 한 점을 기준으로 다른 점들의 수평각을 계산. (360도로 변환)

이후 수평각을 기준으로 점의 배열(vector)를 정렬. 그 순서대로 점을 이으면 단순 다각형을 만들 수 있다.

주어진 점이 아닌 랜덤한 N개(100개로 설정함)의 점도 같은 방법으로 단순 다각형을 만들 수 있음.

점을 입력받아 랜덤한 N개의 점으로 생성된 단순다각형 polygon의 내부에 있는지를 IsInsidePoint로 판별.

모든 선분(100개)에 대하여 입력받은 점 Z에서 그은 수평의 반직선과 교점이 존재하는 지 확인하고, 그 개수가 홀수라면 다각형 내부에 있다고 판별할 수 있음.

점 Z의 y좌표가 선분을 구성하는 두 점의 y좌표 사이에 있는 지 우선 확인한 후에,

교점의 x좌표보다 점 Z의 x좌표가 작다면 교점이 있다고 판별. 이것을 모든 선분에 대해 반복.

원본 코드

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


#include <algorithm>
#include <iostream>
#include <map>
#include <random>
#include <vector>


using namespace std;


typedef struct point {
    int x;
    int y;
    char c;
};


typedef struct line {
    point p1;
    point p2;
};


point polygon[100];


float ComputeAngle(point p1, point p2) {
    int dx, dy, ax, ay;
    float angle;
    dx = p2.x - p1.x;
    ax = abs(dx);
    dy = p2.y - p1.y;
    ay = abs(dy);
    angle = (ax + ay == 0) ? 0.0 : (float)dy / (ax + ay);
    if (dx < 0)
        angle = 2.0 - angle;
    else if (dy < 0)
        angle = 4.0 + angle;
    return angle * 90.0;
}
```

```
// compute a dot is in a polygonVec or not
int crossPointCount = 0;
bool IsInsidePoint(point p, vector<point> polygonVec) {
    int i, j, size = polygonVec.size();
    bool c = false;
    for (i = 0, j = size - 1; i < size; j = i++) {  // i, j is index / j = i - 1 (when
i = 0, j = size - 1)
        if (((polygonVec[i].y > p.y) != (polygonVec[j].y > p.y)) &&
            // 선분과 점의 y좌표가 같은 곳의 x좌표가 p.x보다 작은 경우만 교점이 존재
            // 계산: 선분의 기울기 * (p.y - 선분의 시작점의 y좌표) + 선분의 시작점의 x좌표
            (p.x < (polygonVec[j].x - polygonVec[i].x) * (p.y - polygonVec[i].y) /
(polygonVec[j].y - polygonVec[i].y) + polygonVec[i].x)) {
            c = !c;  // toggle c - 교점이 홀수개면 true, 짝수개면 false
            crossPointCount++;
        }
    }
    return c;
}

int main() {
    // ------------------------------------------------------------------------
    // ------------------ 9-1. 주어진 점들에 대한 단순 다각형 결정 및 출력 --------------
----
    // ------------------------------------------------------------------------
    point poly[8];
    //--
    poly[0].x = 3;
    poly[0].y = 4;
    poly[0].c = 'A';
    poly[1].x = 1;
    poly[1].y = 2;
    poly[1].c = 'B';
    poly[2].x = 2;
    poly[2].y = 5;
    poly[2].c = 'C';
    poly[3].x = 2;
    poly[3].y = 6;
    poly[3].c = 'D';
    poly[4].x = 9;
    poly[4].y = 3;
    poly[4].c = 'E';
    poly[5].x = 5;
    poly[5].y = 3;
    poly[5].c = 'F';
    poly[6].x = 6;
    poly[6].y = 4;
    poly[6].c = 'G';
    poly[7].x = 8;
    poly[7].y = 4;
    poly[7].c = 'H';
    //--

    float polyAngle[8];
```

```
    // 기준점 : B
    // compute Anlges of poly - with respect to B
    for (int i = 0; i < 8; i++) {
        polyAngle[i] = ComputeAngle(poly[1], poly[i]);
    }


    // make map of poly and its angles
    map<char, float> polyMap;
    for (int i = 0; i < 8; i++) {
        polyMap.insert(pair<char, float>(poly[i].c, polyAngle[i]));
    }
    // sort map by value with Insertion sort
    vector<pair<char, float>> polyVec;
    for (auto &p : polyMap) {
        polyVec.push_back(p);
    }
    for (int i = 1; i < polyVec.size(); i++) {
        for (int j = 0; j < i; j++) {
            if (polyVec[i].second < polyVec[j].second) {
                polyVec.insert(polyVec.begin() + j, polyVec[i]);
                polyVec.erase(polyVec.begin() + i + 1);
                break;
            }
        }
    }


    // vector<pair<char, float>> polyVec(polyMap.begin(), polyMap.end());
    // sort(polyVec.begin(), polyVec.end(),
    //      [](const pair<char, float> &l, const pair<char, float> &r) {
    //          return l.second < r.second;
    //      });


    // print sorted map
    cout << "------------------ 9-1. 주어진 점들에 대한 단순 다각형 결정 및 출력 --------
----------" << endl;
    cout << "주어진 점들의 각도: " << endl;
    for (auto &p : polyVec) {
        cout << p.first << " - " << p.second << endl;
    }
    // 결정된 단순 다각형 출력 (print sorted map)
    cout << "주어진 점으로 결정된 단순 다각형: ";
    for (auto &p : polyVec) {
        cout << p.first << " - ";
    }
    cout << polyVec[0].first << endl
         << endl
         << endl;


    // ------------------------------------------------------------------------------
-------------------------
    // ------------------ 9-1. N개의 random한 점에 의한 Polygon 결정, 수평각 계산 회수,
각의 비교 회수 출력 ------------------
```

```
    // ------------------------------------------------------------------------
------------------------


    vector<int> xList, yList;
    int N = 100;


    for (int i = 0; i < N; i++) {
        xList.push_back(i);
        yList.push_back(i);
    }


    // shuffle the vector
    random_device rd;
    default_random_engine rng(rd());
    shuffle(xList.begin(), xList.end(), rng);
    shuffle(yList.begin(), yList.end(), rng);


    // make random polygon
    for (int i = 0; i < N; i++) {
        polygon[i].x = xList[i];
        polygon[i].y = yList[i];
        polygon[i].c = i;
    }


    // compute angles of polygon
    float polygonAngle[N];
    // 기준점 : 0
    // compute Anlges of poly - with respect to point 0
    int computeAngleCount = 0;
    for (int i = 0; i < N; i++) {
        computeAngleCount++;
        polygonAngle[i] = ComputeAngle(polygon[0], polygon[i]);
    }
    // make map of poly and its angles
    map<char, float> polygonMap;
    for (int i = 0; i < N; i++) {
        polygonMap.insert(pair<char, float>(polygon[i].c, polygonAngle[i]));
    }
    // sort map by value with Insertion sort
    int compareCount = 0;
    vector<pair<char, float>> polygonVec;
    for (auto &p : polygonMap) {
        polygonVec.push_back(p);
    }
    for (int i = 1; i < polygonVec.size(); i++) {
        for (int j = 0; j < i; j++) {
            compareCount++;
            if (polygonVec[i].second < polygonVec[j].second) {
                polygonVec.insert(polygonVec.begin() + j, polygonVec[i]);
                polygonVec.erase(polygonVec.begin() + i + 1);
                break;
            }
        }
    }
```

```
    // print sorted map
    cout << "------------------ 9-1. 100개의 random한 점에 의한 Polygon 결정, 수평각 계산 회수, 각의 비교 회수 출력 ------------------" << endl;
    cout << "랜덤한 점들의 각도: " << endl;
    for (auto &p : polygonVec) {
        cout << to_string(p.first) << " - " << p.second << " / ";
    }
    cout << endl
         << endl;
    // 결정된 단순 다각형 출력 (print sorted map), 수평각 계산 회수, 각의 비교 회수 출력
    cout << "랜덤한 점으로 결정된 단순 다각형: " << endl;
    for (auto &p : polygonVec) {
        cout << to_string(p.first) << " - ";
    }
    cout << to_string(polygonVec[0].first) << endl
         << endl;
    cout << "수평각 계산 회수: " << computeAngleCount << endl;
    cout << "각의 비교 회수: " << compareCount << endl;

    // ------------------------------------------------------------------------------------------
    // ------------------------- 9-2. 입력한 점 Z(x,y)가 Polygon 내에 존재하는 점인 지 출력 ----------------------------
    // ------------------------------------------------------------------------------------------

    // input point Z
    cout << "------------------------- 9-2. 입력한 점 Z(x,y)가 Polygon 내에 존재하는 점인 지 출력 ----------------------------" << endl;
    point Z;
    cout << "점 Z(x,y)를 입력하세요:";
    cin >> Z.x >> Z.y;

    // make sorted polygon vector include x, y with polygonVec
    vector<point> sortedPolygon;
    for (auto &p : polygonVec) {
        sortedPolygon.push_back(polygon[p.first]);
    }

    // 입력 받은 점 Z가 Polygon 내에 존재하는 점인 지 출력
    if (IsInsidePoint(Z, sortedPolygon)) {
        cout << "입력받은 점 Z(x,y)는 Polygon 내에 존재합니다." << endl;
    } else {
        cout << "입력받은 점 Z(x,y)가 Polygon 내에 존재하지 않습니다." << endl;
    }
```

```
    // 교차점 개수 출력
    cout << "교차점 개수: " << crossPointCount << endl;
}
```