XXXI НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Национален кръг
Кърджали, 24-26 април 2015 г.
Групи А и В, 9-12 клас, Ден 2

Задача АВ5. С ЛИНИЙКА И ПЕРГЕЛ

Автор: Йордан Чапъров

В час по геометрия учителят Пергелко Линийков поставил на учениците си необикновена задача: той им задал точка-цел в равнината чрез координатите и (*tarx, tary*) и казал:

“Искам, започвайки от множество, съдържащо две точки с координати (0,0) и (1,0), да добавяте нови точки към него, докато някоя от новодобавените точки не се окаже достатъчно близо до точката-цел. Достатъчно близо означава, че евклидовото разстояние между нея и точката-цел е не по-голямо от 0.001. Всяка стъпка на добавяне на нови точки се изразява в следното:

- Построявате с линийка и пергел два геометрични обекта, които могат да бъдат две прави, две окръжности или права и окръжност. Правата се определя от две вече съществуващи в множеството точки, през които минава. Окръжността се определя от три (не непременно различни) съществуващи в множеството точки – едната е център на окръжността, а разстоянието между другите две задава радиуса на окръжността.
- Намирате пресечните точки на построените геометрични обекти (ако има такива) и ги добавяте към множеството.
- Проверявате дали някоя от новодобавените точки е достатъчно близо до точката цел и, ако има такава, ми казвате коя е и приключвате; ако няма, започвате нова стъпка на добавяне на точки.“

Поставената задача предизвикала силен смут и доста тъпи погледи в класа и г-н Линийков, познавайки мощните геометрични способности на учениците си, се смилил и продължил:

„Добре де, построяването с линийка и пергел и намирането на пресечните точки ще извършвам аз, а вие на всяка стъпка само трябва да ми казвате какви два геометрични обекта искате да бъдат построени и кои са точките, които ги определят.“

И така вие сте в ролята на учениците, а програмата на журито в ролята на г-н Линийков. Да формализираме задачата:

Трябва да напишете и предадете към проверяващата система файл **ruler_compass.cpp**, съдържащ функция **void go(double tarx, double tary)**, която се извиква от програмата на журито веднаж, получава координатите на точката-цел и трябва да изпълни описаните по-горе стъпки, докато не се получи точка, намираща се на разстояние не по-голямо от 0.001 от точката-цел. Файлът **ruler_compass.cpp** може да съдържа и други функции и глобални структури от данни, които са необходими за решаване на задачата. Той не трябва да съдържа функция *main()*. Трябва да се строи множество *S* от точки с идентификатори в равнината. Ще отбелязваме точка с координати (*x*,*y*) и идентификатор *id* като *id:(x,y)*.

Първоначално *S = {0:(0, 0), 1:(1, 0)}*. За добавянето на точки към S ви се предоставят следните функции:

void define_line(int ida, int idb); - дефинира права, минаваща през точки с идентификатори *ida* и *idb*. Двете точки трябва непременно да бъдат различни (разстоянието между тях да е по-голямо от 0)

void define_circle(int idc, int idr1, int idr2); - дефинира окръжност с център точка с идентификатор *idc* и радиус, разстоянието между точките с идентификатори *idr1* и *idr2*. Разстоянието между точките с идентификатори *idr1* и *idr2* трябва непременно да бъде по-голямо от 0.

XXXI НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Национален кръг
Кърджали, 24-26 април 2015 г.
Групи А и В, 9-12 клас, Ден 2

int intersect(int &id1, double &x1, double &y1, int &id2, double &x2, double &y2) - изчислява пресечните точки на последните два дефинирани обекта и връща броя на намерените пресечни точки. Броят на пресечните точки ще бъде 0, 1 или 2. Ако двете геометрични фигури имат повече от 2 пресечни точки, то те съвпадат и викането на *intersect* е било невалидно. Новоизчислените идентификатори и координати на точки ще бъдат записани в променливите *id1, x1, y1, id2, x2, y2*. Функцията *intersect()* винаги ще дава идентификатори, които са последователни, неизползвани за други точки цели числа. След извикването на *intersect*, ново-генерираните точки трябва да се добавят в множеството S, и могат да се проверяват за близост до точката-цел и да се използват при дефиниране на следващите фигури. Ако имаме само една пресечна точка, само тройката *id1, x1, y1* ще се използва за изходните данни. Ако нямаме пресечни точки, стойностите на *id1, x1, y1, id2, x2, y2* са недефинирани.

Важно: Преди всяко извикване на функция *intersect()*, трябва да дефинирате, чрез две извиквания на функции *define…..()*, и двата обекта, пресечните точки, на които тя трябва да изчисли.

При получаване на точка, която е достатъчно близо до точката-цел, трябва да се извика функция (която също ви се предоставя)

void done(int id);

която да съобщи на програмата на журито идентификатора на получената точка.

Нека обобщим накратко: Трябва да напишете функция *void go(double tarx, double tary),* която започва работа от множеството с точки *S = {0:(0, 0), 1:(1, 0)}* и последователно генерира още точки в *S* чрез викания на *define_line()*, *define_circle()* и *intersect()*, докато не се получи точка *k:(myx, myy)*, която е достатъчно близка до *(tarx, tary)*. Тогава, функцията *go()* трябва да извика *done(k)* и да приключи изпълнение.

Ограничения: $0 \le tarx, tary < 10000$

Оценяване:

Ако вашата програма докладва точка (*myx, myy*), на разстояние от (*tarx, tary*), по-голямо от 0.001, тя получава 0 точки за теста.

Нека вашата програма за даден тест е викала функцията *intersect() p* пъти. Тогава за този тест тя ще получи от 0 до 10 точки, които се смятат по формулата:

*score = 10 – (p – 45) * 0.085*

Ако *score* е по-малко от 0, резултатът за теста е 0. Ако е по-голямо от 10, резултатът за теста е 10 точки.

В 10% от тестовете, *tary* = 0, *tarx* ≤ 40, *tarx* е цяло число

В други 10% от тестовете, *tary* = 0, *tarx* е цяло число

В други 10% от тестовете, *tary* = 0, *tarx* ≤ 10

В други 10% от тестовете, *tary* = 0

В други 10% от тестовете, *tarx* и *tary* са цели числа

Всеки тест се оценява отделно.

Примерна интеракция:

```
go(1.5, 0.866025404);                    // S = {0:(0, 0), 1:(1, 0)}
define_line(0, 1);
define_circle(1, 0, 1);
intersect(id1, x1, y1, id2, x2, y2);
    // intersect() връща 2.
```

```
        // id1 = 2, x1 = 0, y1 = 0 ; id2 = 3, x1 = 2, y2 = 0
        // S = {0:(0, 0), 1:(1, 0), 2:(0, 0), 3:(2, 0)}
define_circle(1, 1, 3);
define_circle(3, 1, 3);
intersect(id1, x1, y1, id2, x2, y2);
        // intersect() връща 2.
        // id1 = 4, x1 = 1.5, y1 =  0.866025404 ; id2 = 5, x2 = 1.5, y2 = -0.866025404
        // S = {(0:(0, 0), 1:(1, 0), 2:(0, 0), 3:(2, 0), 4:(1.5, 0.866025404), 5:(1.5, -0.866025404)}
done(4);
        // OK. 2 стъпки -> 10 точки
```

Указания:

На състезателите се предоставя файл **contestant_grader.cpp** и един тест във файл **test.00.in** . Целта е да могат да си тестват написаната функция *go*. Примерът в test.00.in е за точката-цел, с която се вика функцията *go* в примера по-горе. Не е задължително contestant_grader и grader на журито да съвпадат. Гарантира се, че двата файла ще имат същото поведение на взаимодействие с програмата на състезателя – ще изчисляват координатите на точките по един и същи начин.

За да компилирате *contestant_grader.cpp* с вашата функция *go*, прочетете указанията във файлове README, предоставени от системата.