АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
LINES

Тъй като ограничението не е голямо, бихме могли да използваме масив, в който да съхраняваме редица от представителите на даден клас. Всеки елемент на масива ще съхранява правите от различен клас. След въвеждане на входните данни и генериране на масива, неговата дължина ще показва точно броя на различните класове прави.
От друга страна, редиците, съхраняващи различните представители на даден клас няма нужда да се съхраняват в масив, тъй като ще се резервира многократно повече памет от необходимата (ограничението е за общия брой прави, но за всеки клас неговите представители могат да са от 1 до 1000). За това е по-удачно да се използва линейна динамична структура, позволяваща обхождане. Това съображение води до избора на линеен списък.
И не на последно място – достатъчно е да се съхраняват коефициентите на един представител на съвпадащи прави.

Таблицата показва структурата от данни, която се получава накрая (защрихованите клетки реално няма да се включат в нея, тъй като представят вече съществуващи прави):

№	Клас			Дължина на списъка
	a	***b***	***c***	
0.	1	-1	2	1
	-3	3	-6	
1.	2	3	-4	1
2.	-4	10	8	2
	6	-15	-20	
	-2	5	4	

Сборът от дължините на всички списъци представя броя на различните прави.

Забележка:

Нека $l_1: a_1x+b_1y+c_1=0$
$l_2: a_2x+b_2y+c_2=0$

Двете прави са от един клас, ако $a_1b_2 = a_2b_1$.
Двете прави съвпадат, когато $(a_1b_2=a_2b_1)\wedge(a_1c_2=a_2c_1)\wedge(b_1c_2=b_2c_1)$.

Автор: Николина Николова

```
//lines.cpp
#include <iostream>
#include <list>

using namespace std;

struct line{
     int a,b,c;
};
typedef list<line> lineList;    //list of lines
lineList lineClasses[1000];     //array of different classes of lines
int n = 0;                      //number of elements in the array

bool sameClass(line, line);     //checks if two lines are in the same class
bool eqLines(line, line);       //checks if two lines of the same class
                                // are equal
bool isInList(list<line>, line);          //check if the line is in the list
void read();                    //reads and stores the input data
int calcDiff();                 //calculates the number of different lines

int main(){
     read();
     cout << calcDiff() << endl;
     cout << n << endl;
     return 0;
}

bool sameClass(line l1, line l2){
     return l1.a*l2.b == l2.a*l1.b;
}

bool eqLines(line l1, line l2){
     return (l1.a*l2.c == l2.a*l1.c)  && (l1.b*l2.c == l2.b*l1.c);
}


bool isInList(list<line> lst, line l){
     lineList::iterator lst_iter;
     for (lst_iter = lst.begin(); lst_iter != lst.end(); lst_iter++)
          if (eqLines(*lst_iter, l))
               return true;
     return false;
}

void read(){

     line l;
     int N;

     cin >> N;
     cin >> l.a >> l.b >> l.c;
     lineClasses[0].push_back(l); n++;

     for (int i=1; i<N; i++){
          cin >> l.a >> l.b >> l.c;
          int j;
          for (j = 0; (j < n) && !sameClass(l, lineClasses[j].front()); j++);
          if (!isInList(lineClasses[j],l))
               lineClasses[j].push_back(l);
          if (j==n) n++;                 //new class
     }

}

int calcDiff(){
     int cnt = 0;
     for (int i=0; i<n; i++)
          cnt += lineClasses[i].size();
     return cnt;
}
```