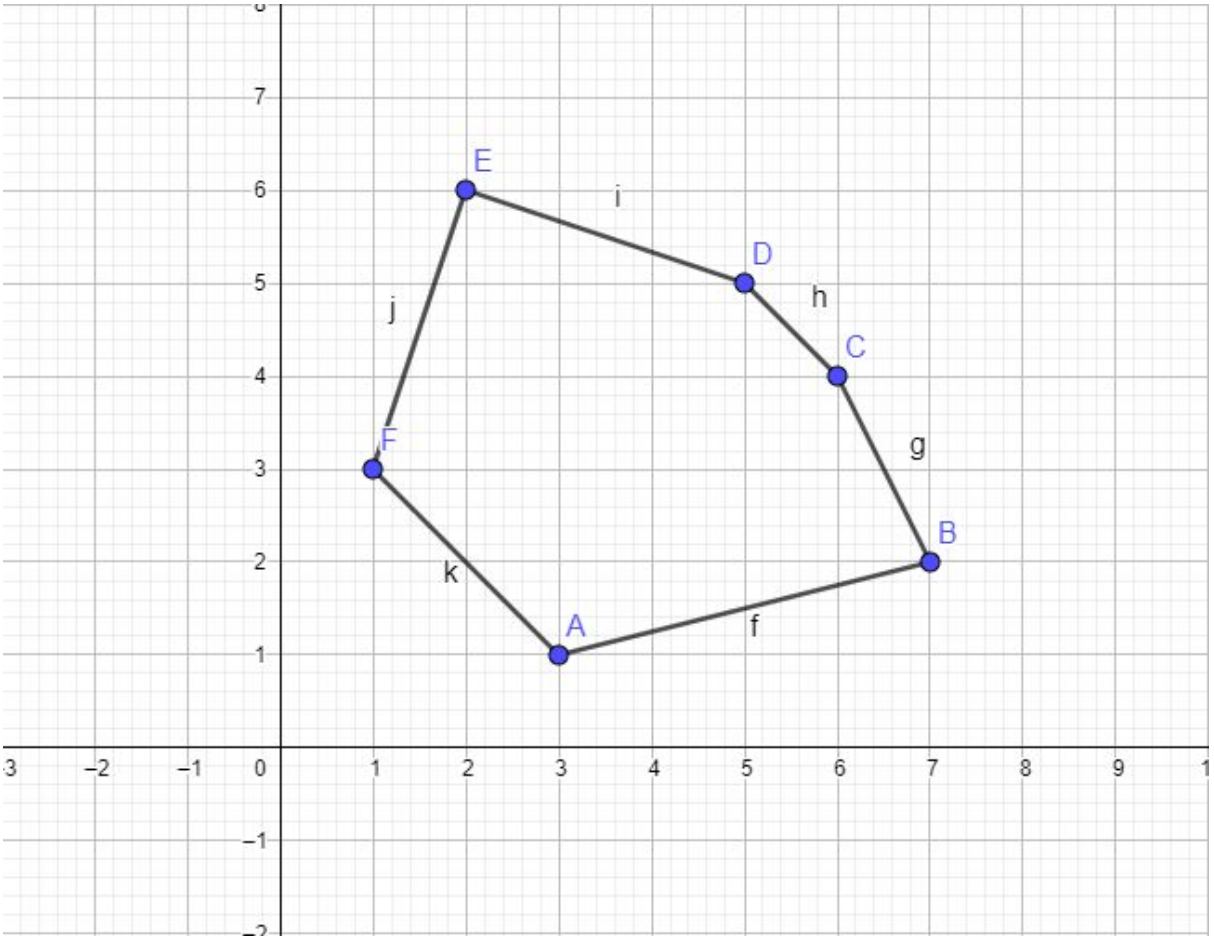
Ориентирани лица

Някога мечтали ли сте си да пресметнете лицето на случаен многоъгълник зададен спрямо броя на ъглите и техните координати?? На пръв поглед ни изскачат методи като:

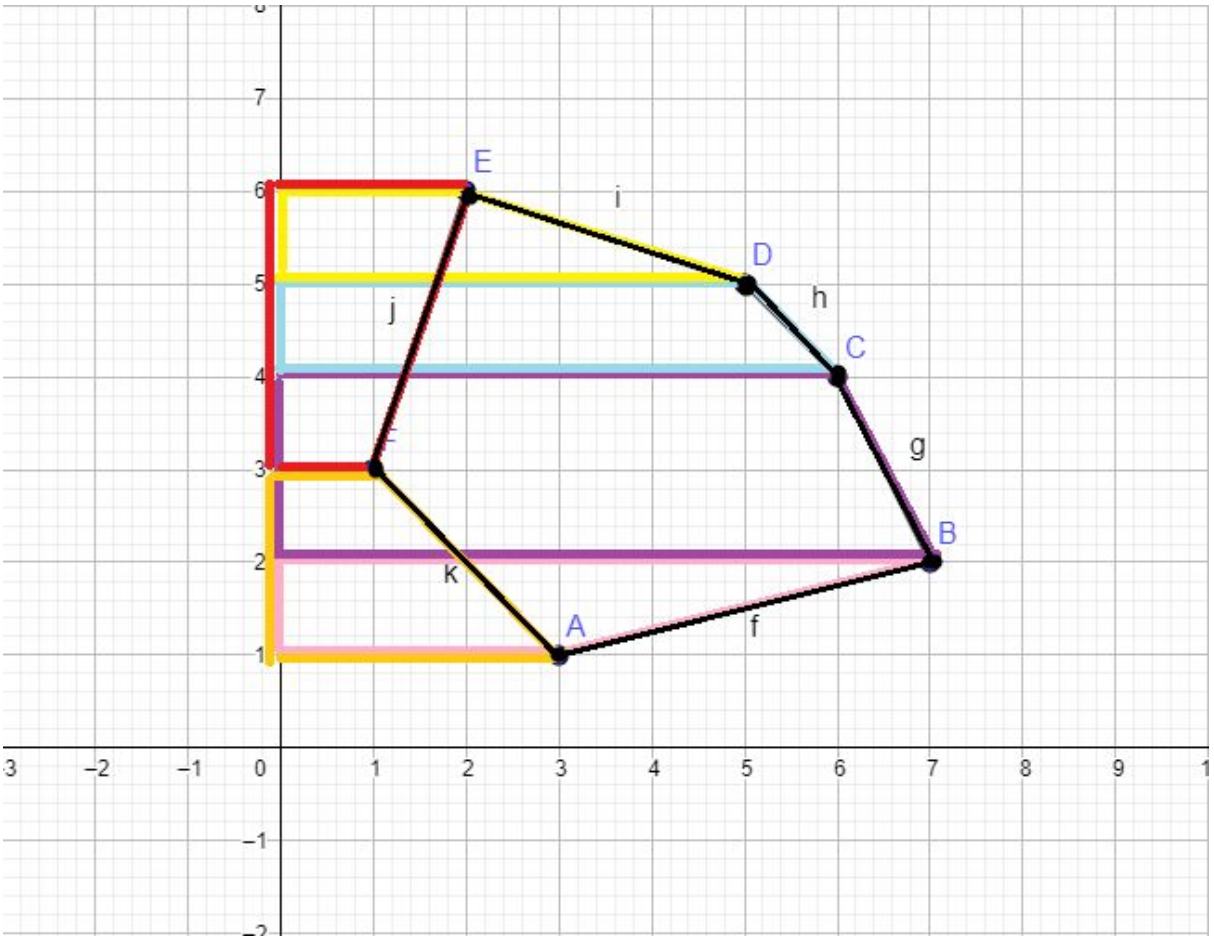
1. Разделяне на многоъгълника на много триъгълничета (правоъгълни) и след това събиране на лицата на всички триъгълничета
2. Опаковане на многоъгълника: допълване до правилна фигура (правоъгълник/квадрат) и след това това пресмятане на сбора на лицата на излишните фигури и изваждане от лицето на правилната фигура.

Наясно сме, че тези методи са полезни при пресмятане на лица на такива фигури в математиката, но в повечето случаи в програмирането са или трудно-приложими, или невъзможни за използване.
Съществува интересен, лесен и сравнително неинтуитивен начин за пресмятане на лицето на многоъгълник, който наподобява метода за разделяне на фигури, на които можем лесно да пресметнем лицето.
Ще се опитам да ви го представя и да ви предложа за какво може да бъде приложим.

Нека имаме следния многоъгълник:



Както се разбрахме от условието на задачата - в програмата ни той ще представлява координатите на точките A, B, C, D, E, F. И това което ще направим е да разделим фигурата ни трапци по следния начин:



Тоест на всеки 2 точки ще им намерим проекцията на правата OY и по този начин ще образуваме трапец. Хайде да сметнем лицето на розовия трапец:
(a+b)*h/2 = (3+7)*1/ 2 = 5
Забелязваме, че дължините на основите представляват всъщност X координатите на 2те поредни точки които разглеждаме в момента (а именно разстоянието от 0 до самата точка по правата X). Ами ако искаме да пресметнем височината: това ще бъде разликата в Y-ите на 2те точки, а за да се получи положителна височина в случая ще трябва да извадим от у координатата на B, у координатата на A. По този начин лицето на розовия трапец изразено чрез координати ще изглежда по този начин: (Xa+Xb)*(Yb-Ya)/2. Забелязваме, че в лицето на този правоъгълник се включват части, които не са част от многоъгълника, но нека игнорираме това за сега. Нека да изразим лицето на лилавия трапец: (Xb+Xc)*(Yc-Yb)/2. Синия: (Xc+Xd)*(Yd-Yc)/2. Жълтия: (Xd+Xe)*(Ye-Yd)/2. Става по интересно, когато започнем да пресмятаме лицето на червения трапец. Ако запазим същата последователност, която сме използвали досега би трябвало да се получи следното: (Xe+Xf)*(Yf-Ye)/2 = (2+1)*(3-6)/2 = 3*(-3)/2 = -4,5 което е странен резултат за лице на фигура. Ако се замислим какво би станало ако съберем лицата на всички трапци: сбора на розовия, лилавия, синия и жълтия ще представлява лицето на многоъгълника + излишната част вляво от многоъгълника, която се простира до правата OY. Прибавяйки стойността на лицето на червения трапец, което по нашата формула е отрицателно число, ние всъщност изваждаме от досегашната сума площта на червения трапец. Но ако обърнем внимание този червен трапец представлява някаква част от картинката, която не принадлежи на

многоъгълника. Това всъщност е фигура обхващаща всички излишни части от лицата на трапците събрани досега и би ни било полезно да я извадим. Същото ще направим и с оранжевия трапец и успешно ще махнем останалото от ненужно добавената част. В крайна сметка сбора на всички трапци е: розовия трапец + лилавия трапец + синия трапец + жълтия трапец + (-червения трапец) + (-оранжевия трапец) = лицето на многоъгълника!

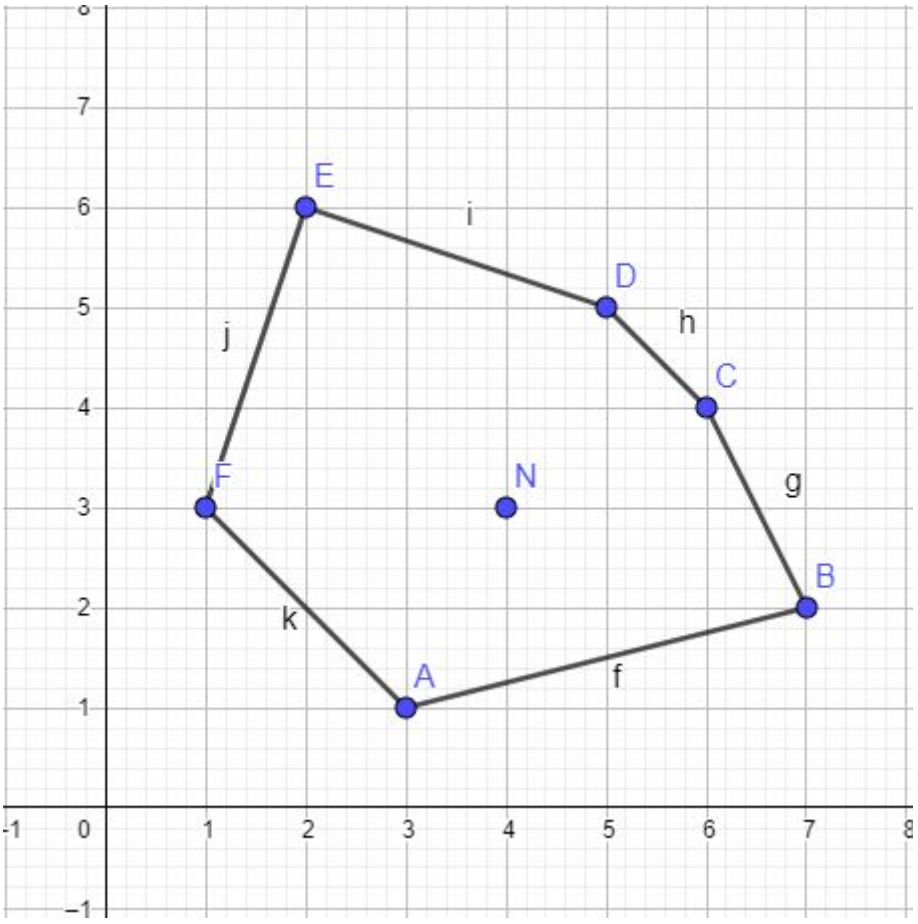
Кода за намирането на лице на многоъгълник по този метод ще изглежда така:

```
#include<iostream>
using namespace std;
double s(double x1, double y1, double x2, double y2){
    return (x1+x2)*(y2-y1)/2;
}
int main (){
int n;
double x[100], y[100];
cin >> n;
for(int i = 0; i < n; i ++){
    cin >> x[i] >> y[i];
}
double area = 0;
for(int i = 0; i < n - 1; i ++){
    area += s(x[i], y[i], x[i+1], y[i+1]);
}
area += s(x[n-1], y[n-1], x[0], y[0]);
cout << area;
return 0;
}
```

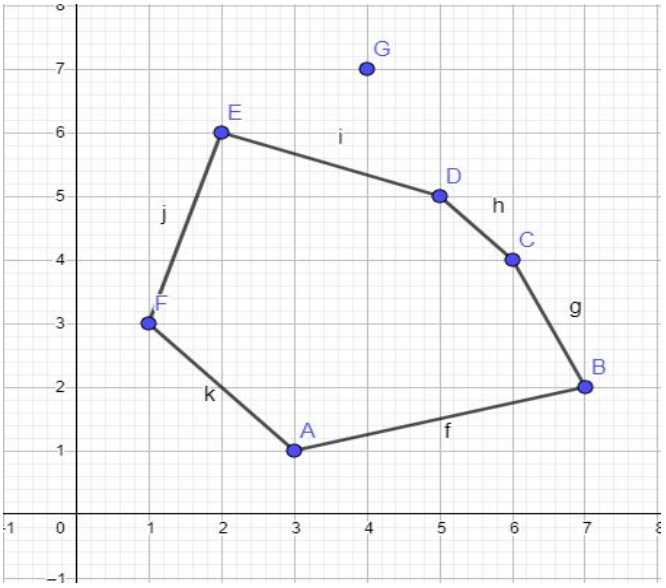
Най напред би ни било полезно да имаме функция, която пресмята лицето на всеки отделен трапец (това е double функцията s). Имайки въведените данни от потребителя за конкретен многоъгълник можем да обходим всички точки и да сметнем лицата на трапците спрямо всеки 2 последователни точки. Сумирайки всички лица ще получим търсеното на многоъгълника.

Приложение

Освен че е доста полезно за намиране на лица на случайни фигури (даже и вдлъбнати многоъгълници да са) можем да използваме този метод и за проверка дали дадена точка попада вътре в случаен многоъгълник. Та ако имаме същата фигура обаче ни е дадена и точката с координати (a,b) например лесно можем да проверим дали тя попада в многоъгълника:



По формулата за която си говорихме по горе може да изчислим точно колко е лицето на многоъгълника ABCDEF. Но ние можем да го изчислим и по още един начин използвайки точката дадена ни като вход от потребителя: Sabcdef = Sabn + Sbcn + Scdn + Sden + Sefn + Sfan. Ако пресметнем лицата и по двата начина и се окаже, че те са равни, то тогава очевидно точката принадлежи на многоъгълника. Един случай в който ще се окаже че лицата са различни и което ще доведе до извода че точката лежи извън многоъгълника е ето този:



Очевидно лицето на ABCDEF ще е различно от сбора на: Sabg + Sbcg + Scdg + Sdeg + Sefg + Sfag.