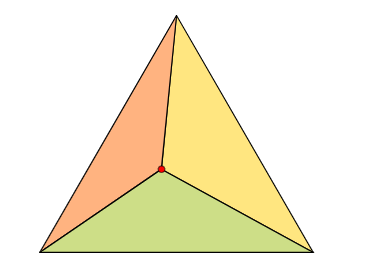
**Zadanie**

Dany jest trójkąt *T*, który został opisany przez współrzędnej wszystkich trzech wierzchołków: *(x1, y1)*, *(x1, y1)*, *(x1, y1)*. Dany jest punkt *P = (xp, yp)*. Zadanie polega na stwierdzeniu czy punkt *P* leży wewnątrz trójkąta *T*.

## Implementacja

### Prosta

Jednym z najprotszych sposobów na sprawdzenie czy punkt leży wewnątrz trójkąta jest podzielenie go na trzy podobszary tak jak zostało to przedstawione poniżej:



Sumując wszystkie te trzy obszary suma powinna się równać polu trójkąta wyjściowego. Jeśli tak nie jest to znaczy, że punkt leży poza trójkątem. Potrzebna jest tutaj formuła na wyliczanie pola trójkąta na podstawie jego współrzędnych. Wzór wygląda następująco: **T∆ = 0.5·|x1(y2 - y3) + x2(y3 - y1 + x3(y1 - y2))|** co zapisane w postaci funkcji **Obszar()** przedstawia się następująco:

1. double Obszar(int x1, int y1, int x2, int y2, int x3, int y3) {
2. return 0.5\*abs((x1\*(y2 - y3) + x2\*(y3 - y1) + x3\*(y1 - y2)));
3. }

Wtedy funkcja sprawdzająca czy punkt jest w środku będzie polegać na wyliczeniu wszystkich kolejnych trójkątów, a następnie porównanie z polem trójkąta niepodzielonego na obszary. Przykładowa implementacja wygląda następująco:

1. bool CzyWewnatrz(int x1, int y1, int x2, int y2, int x3, int y3, int xp, int yp) {
2. double T = Obszar(x1, y1, x2, y2, x3, y3);
3. double T1 = Obszar(xp, yp, x2, y2, x3, y3);
4. double T2 = Obszar(x1, y1, xp, yp, x3, y3);
5. double T3 = Obszar(x1, y1, x2, y2, xp, yp);
6. return (T == T1 + T2 + T3);
7. }

Wyliczany jest najpierw (2.) obszar całego trójkąta, a potem (3. - 5.) podobszarów i (6.) wynikiem jest porównanie sumy podobszarów do obszaru głównego. Warto zauważyć, że problemem może się tutaj okazać, że dane w komputerze są przechowywane w postaci przybliżonej.

### Metoda efektywna

Sprawdzanie obszaru za każdym razem jest metodą prostą, ale nieefektywną. Proces ten można przyśpieszyć używając wpółrzędnych barycentrycznych. Przepis na znalezienie czy punkt jest w środku trójkąta jest wtedy następujący:

Na początek przekształć punkty względem punktu A w następujący sposób: *B = B - A*, *C = C - A*, *P = P - A*. Następnie wylicz *d = x2y3-x3y2*. Teraz pozostaje sprawdzić czy następujące wartości znajdują się w przedziale **[0, d]**:

1. xp(y2 - y3) + yp(x3 - x2) + x2y3 - x3y2
2. xpy3 - ypx3
3. ypx2 - xpy2

Program realizujący to zadanie przedstawia się następująco:

1. bool CzyWewnatrz(int x1, int y1, int x2, int y2, int x3, int y3, int xp, int yp) {
2. x2 -= x1; y2 -= y1;
3. x3 -= x1; y3 -= y1;
4. xp -= x1; yp -= y1;
5. double d = x2\*y3 - x3\*y2;
6. double w1 = xp\*(y2 - y3) + yp\*(x3 - x2) + x2\*y3 - x3\*y2;
7. double w2 = xp\*y3 - yp\*x3;
8. double w3 = yp\*x2 - xp\*y2;
9. return SprawdzZakres(w1, 0, d) && SprawdzZakres(w2, 0, d) && SprawdzZakres(w3, 0, d);
10. }

W programie kolejno są wyliczane: (2. - 4.) kolejne przesunięcia poszczególnych punktów, (5.) skalar *d* oraz (6. - 8.) wartości barycentrycznych wag. Na koniec (9.) wynikiem jest koniunkcja wszystkich warunków.

Pomocnicza funkcja zakres zwraca wartość logiczną prawda wtedy i tylko wtedy, gdy podana wartość *val* leży w zakresie **[min, max]**.

1. bool SprawdzZakres(double val, double min, double max) {
2. return val >= min && val <= max;
3. }

## Testowanie funkcji

W celu przetestowania funkcji można skorzystać z poniższej funkcji **main()**, która wczyta od użytkownika potrzebne dane, a następnie wypisze na ekran informację czy punkt leży wewnątrz trójkąta:

1. int main() {
2. int x1, y1, x2, y2, x3, y3, xp, yp;
3. cout << "Podaj wspolrzedne punkt P1: ";
4. cin >> x1 >> y1;
5. cout << "Podaj wspolrzedne punkt P2: ";
6. cin >> x2 >> y2;
7. cout << "Podaj wspolrzedne punkt P3: ";
8. cin >> x3 >> y3;
9. cout << "Podaj punkt P: ";
10. cin >> xp >> yp;
11. cout << "Punkt P";
12. cout << (CzyWewnatrz(x1, y1, x2, y2, x3, y3, xp, yp) ? "" : "nie ");
13. cout << " lezy wewnatrz T o wierzcholkach P1, P2 i P3";
14. system("pause");
15. return 0;
16. }