Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 2 «Разветвляющиеся алгоритмы»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Комбаев И. Д.

студент группы 609-01

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2020 г.

Цель работы:

* закрепление знаний конструкции выбора структурного программирования;
* получение практических навыков разработки разветвляющихся алгоритмов;
* построение логических выражений и использования условных операторов

Задание: Найти общее решение вычислительной задачи в соответствии с индивидуальным заданием, выданным преподавателем, разработать алгоритм решения и реализовать программу.

Вариант 26. Заданы координаты вершит четырехугольника. Определить его тип: прямоугольник, параллелограмм, трапеция, квадрат, произвольный.

**Вариант 26**

**Формальное описание задачи.**

1. **Свойства фигур**

Все фигуры имеют уникальное множество свойств, которое и характеризует фигуру. Я выделил 4 свойства, по которым можно охарактеризовать любую фигуру:  
1) Выпуклость  
2) Количество пар параллельных линий  
3) Являются ли все стороны: равны, равны только противоположные, остальные случаи.  
4) Равны ли диагонали фигуры  
 Все эти 4 свойства можно найти зная всего лишь координаты вершин четырехугольника.

Характеристика фигур:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер свойства:** | **1** | **2** | **3** | **4** |
| **Квадрат** | Выпуклый | 2 Пары | Все равны | Да |
| **Прямоугольник** | Выпуклый | 2 Пары | Противоп. Равны | Да |
| **Параллелограмм** | Выпуклый | 2 Пары | Противоп. Равны | Нет |
| **Трапеция** | Выпуклый | 1 Пара | Произвольно | Произвольно |

Проанализировав фигуру и найдя все 4 свойства – мы можем судить о том, что это за фигура. Если найденное множество свойств не подходит ни одному множеству свойств вышеперечисленных фигур – тогда фигура произвольная.  
 В случае трапеции нам лишь достаточно знать, что фигура выпуклая и у неё всего 1 пара параллельных сторон.

1. **Нахождение свойств фигур**
   1. **Выпуклость**

Для того, чтобы найти выпуклость фигуры, мы воспользуемся умножением векторов.

Для определения выпуклости с помощью умножения векторов, мы сначала находим определители всех четырех векторов фигуры  
 Для этого воспользуемся формулой нахождения определителя вектора:

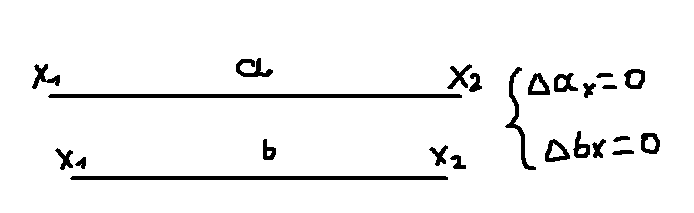
Если для всех пар ребер многольника выполняет условие:

Для определения выпуклости мы будем искать определители, обходя ребря против часовой стрелки. Если все опредилители имееют один знак – значит фигура выпуклая.

**2.2 Колличество пар параллельных сторон**

Если соотношеие разности Х и Y координаты начальной и конечной точки стороны равны аналогичному соотношению другой стороны, то стороны параллельны

Если же или , то линии параллельны



Таким образом мы можем сравнить две пары противоположных сторон на параллельность и получить количество параллельных сторон

**2.3 Равность сторон**

Длина стороны отрезка находится по формуле Пифагора для прямоугольного треугольника, где X – модуль разницы значений координат отрезка по иксу, а Y – модуль разницы значений координат отрезка по игреку.

Сравнивая все стороны с друг другом, мы можем найти одно из трех состояний:

1. Либо все стороны равны
2. Либо только противоположные стороны равны
3. Остальные случаи

**2.4 Равность диагоналей**

Находим длину диагоналей фигуры и сравниваем их

Для нахождения диагонали фигуры мы используем формулу Пифагора, где Х и Y длины смежных сторон.

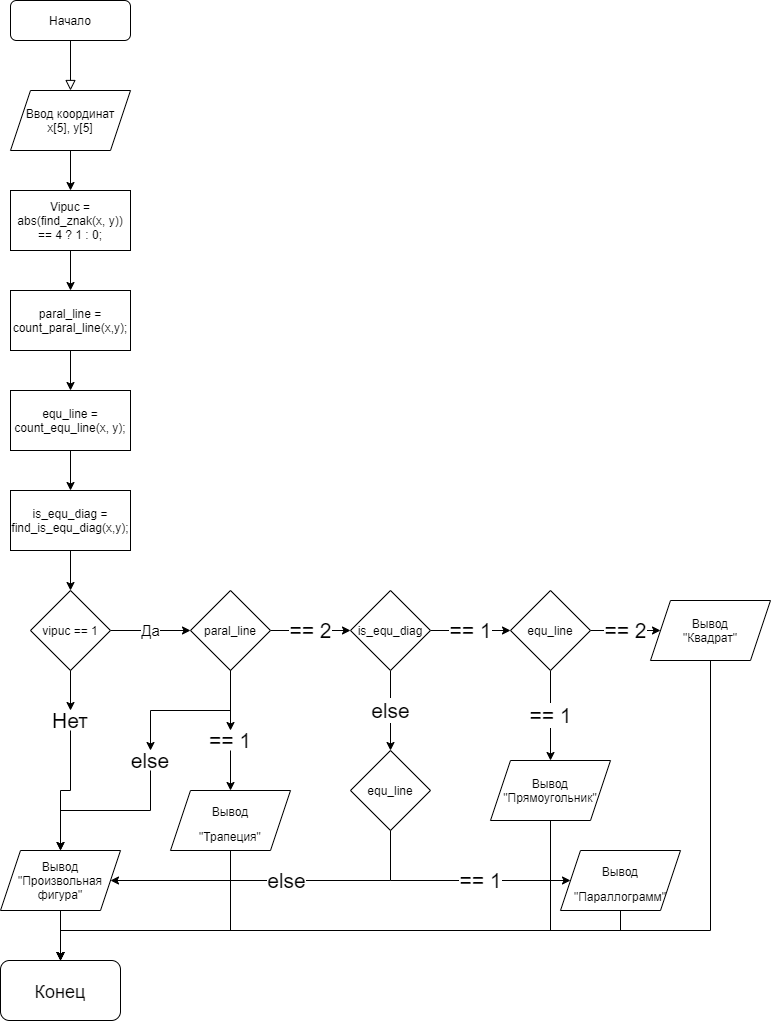
**Алгоритм программы.**

Для решения данной задачи необходимо:

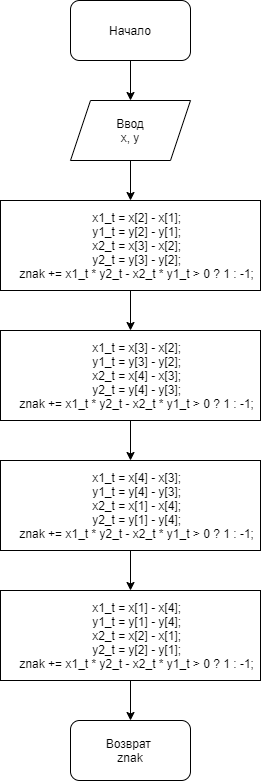
1. Ввести исходные данные – количество координаты точек
2. Узнать – выпуклая ли фигура
3. Вычислить количество параллельных сторон.
4. Определить свойство равности сторон.
5. Определить – равны ли диагонали фигуры
6. Имея все свойства фигуры – определить название фигуры
7. Вывести результаты на экран.

**Блок-схемы алгоритма программы.**

**int main()**

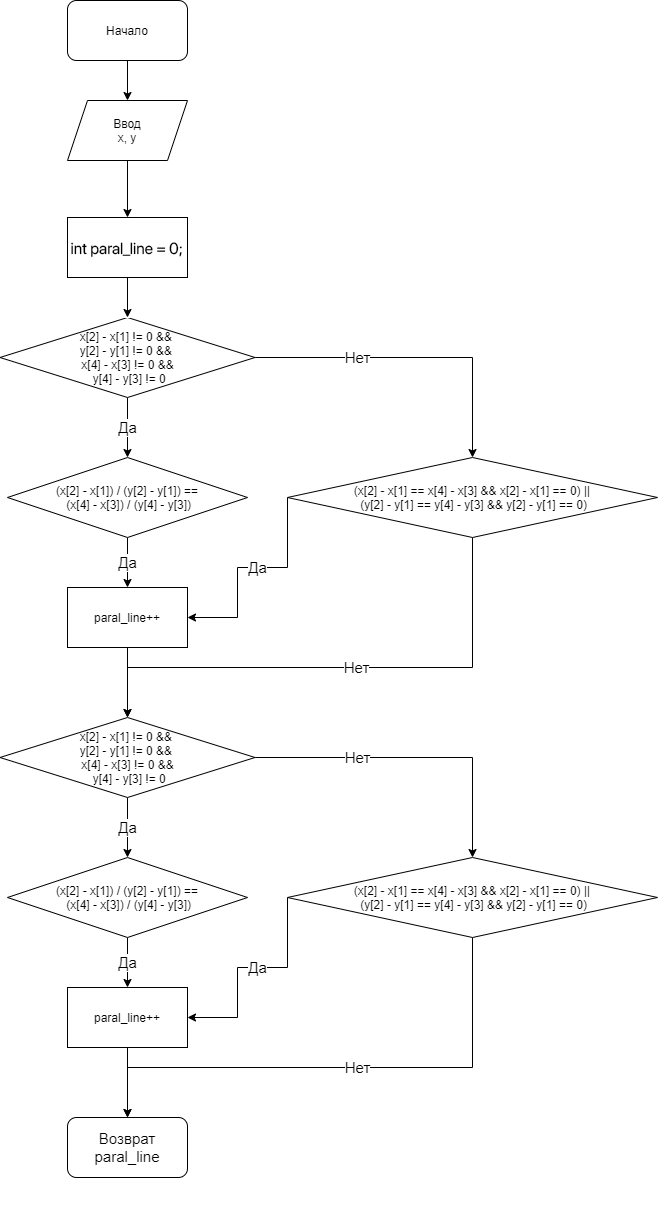
****

**int find\_znak(double x[5], double y[5])**

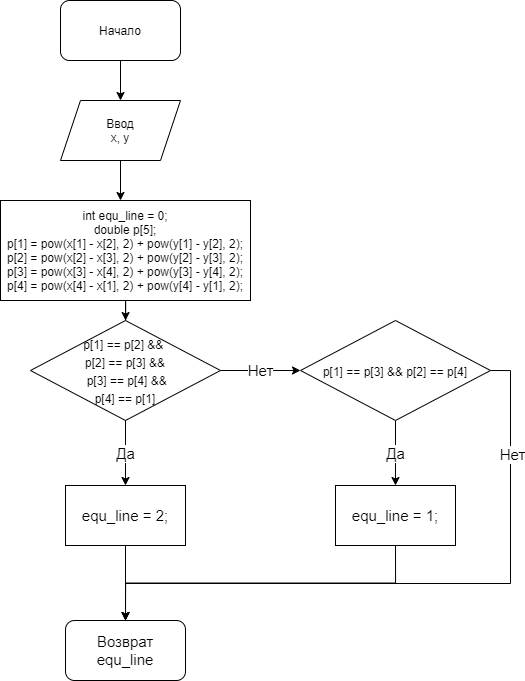
****

int znak = 0;

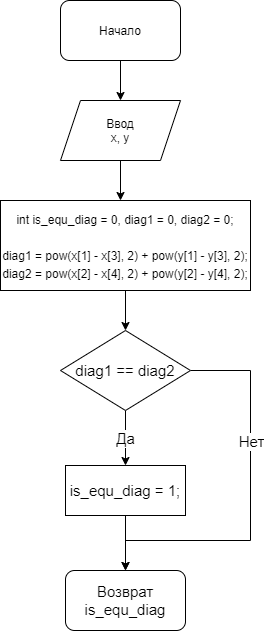
**int count\_paral\_line(double x[5], double y[5])**

****

**int count\_equ\_line(double x[5], double y[5])**

****

**int find\_is\_equ\_diag(double x[5], double y[5])**

****

**Листинг программы.**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. **int** find\_znak(**double** x[5], **double** y[5]);
6. **int** count\_paral\_line(**double** x[5], **double** y[5]);
7. **int** count\_equ\_line(**double** x[5], **double** y[5]);
8. **int** find\_is\_equ\_diag(**double** x[5], **double** y[5]);
10. **int** main() {
11. **int** is\_equ\_diag = 0, equ\_line = 0, paral\_line = 0, vipuc = 0;
12. **double** x[5];
13. **double** y[5];
15. system("chcp 1251 > null");
17. printf("Введите координаты 4-ех точек\n");
19. printf("Первая точка: ");
20. scanf("%lf%lf", &x[1], &y[1]);
22. printf("Вторая точка: ");
23. scanf("%lf%lf", &x[2], &y[2]);
25. printf("Третья точка: ");
26. scanf("%lf%lf", &x[3], &y[3]);
28. printf("Четвертая точка: ");
29. scanf("%lf%lf", &x[4], &y[4]);
31. vipuc = abs(find\_znak(x, y)) == 4 ? 1 : 0; // Если колличество элементов с одним знаком равно 4, то фигура выпуклая
32. paral\_line = count\_paral\_line(x, y);
33. equ\_line = count\_equ\_line(x, y);
34. is\_equ\_diag = find\_is\_equ\_diag(x, y);
36. **if** (vipuc == 1) {
37. **if** (paral\_line == 2) {
38. **if**(is\_equ\_diag) {
39. **if** (equ\_line == 2) {
40. printf("Квадрат");
41. } **else** **if** (equ\_line == 1) {
42. printf("Прямоугольник");
43. }
44. } **else** {
45. **if** (equ\_line == 1) {
46. printf("Параллелограмм");
47. } **else** {
48. printf("Произвольная фигура"); // Вообще - ромб
49. }
50. }
51. } **else** **if** (paral\_line == 1) {
52. printf("Трапеция");
53. } **else** {
54. printf("Произвольная фигура");
55. }
56. } **else** {
57. printf("Проивольная фигура");
58. }
60. printf("\n");
61. system("pause");
62. **return** 0;
64. }
66. //Определение выпуклости фигуры с помощью произведения векторов (против часовой)
67. **int** find\_znak(**double** x[5], **double** y[5]) {
68. **int** znak = 0;
69. **double** x1\_t, x2\_t, y1\_t, y2\_t;
71. // [x1\_t;y1\_t] - AB
72. // [x2\_t;y2\_t] - BC
73. x1\_t = x[2] - x[1];
74. y1\_t = y[2] - y[1];
75. x2\_t = x[3] - x[2];
76. y2\_t = y[3] - y[2];
77. znak += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1; // [ABxBC]
79. x1\_t = x[3] - x[2];
80. y1\_t = y[3] - y[2];
81. x2\_t = x[4] - x[3];
82. y2\_t = y[4] - y[3];
83. znak += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
85. x1\_t = x[4] - x[3];
86. y1\_t = y[4] - y[3];
87. x2\_t = x[1] - x[4];
88. y2\_t = y[1] - y[4];
89. znak += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
91. x1\_t = x[1] - x[4];
92. y1\_t = y[1] - y[4];
93. x2\_t = x[2] - x[1];
94. y2\_t = y[2] - y[1];
95. znak += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;

98. // Если все значения разности имели один знак, значит фигура выпуклая, а znak по модулю имеет число 4 - кол-во элементов с одним элементом
99. **return** znak;
100. }
102. // Поиск количества пар параллельных сторон
103. **int** count\_paral\_line(**double** x[5], **double** y[5]) {
105. **int** paral\_line = 0;
107. // Если соотношеие разности Х и Y координаты начальной и конечной точки стороны равны аналогичному соотношению другой стороны
108. // то стороны параллельны
109. // Если разность Икса\Игрека точки начала отрезка и Икса\Игрека конечной точки отрезка равна нулю
110. // И если у второго отрезка аналогично разность равна нулю - то эти две стороны параллельны
111. **if** (x[2] - x[1] != 0 && y[2] - y[1] != 0 && x[4] - x[3] != 0 && y[4] - y[3] != 0) {
112. **if** ((x[2] - x[1]) / (y[2] - y[1]) == (x[4] - x[3]) / (y[4] - y[3])) {
113. paral\_line++;
114. }
115. } **else** **if** ((x[2] - x[1] == x[4] - x[3] && x[2] - x[1] == 0) ||
116. (y[2] - y[1] == y[4] - y[3] && y[2] - y[1] == 0)) {
117. paral\_line++;
118. }
120. **if** (x[3] - x[2] != 0 && y[3] - y[2] != 0 && x[1] - x[4] != 0 && y[1] - y[4] != 0) {
121. **if** ((x[3] - x[2]) / (y[3] - y[2]) == (x[1] - x[4]) / (y[1] - y[4])) {
122. paral\_line++;
123. }
124. } **else** **if** ((x[3] - x[2] == x[1] - x[4] && x[3] - x[2] == 0) ||
125. (y[3] - y[2] == y[1] - y[4] && y[3] - y[2] == 0)) {
126. paral\_line++;
127. }
129. **return** paral\_line;
130. }
132. // Определения свойства равности строк
133. // 2 - все строки равны
134. // 1 - противолежащие строки равны
135. // 0 - Остальные случаи
136. **int** count\_equ\_line(**double** x[5], **double** y[5]) {
137. **int** equ\_line = 0;
138. **double** p[5];
140. //длины сторон
141. p[1] = pow(x[1] - x[2], 2) + pow(y[1] - y[2], 2);
142. p[2] = pow(x[2] - x[3], 2) + pow(y[2] - y[3], 2);
143. p[3] = pow(x[3] - x[4], 2) + pow(y[3] - y[4], 2);
144. p[4] = pow(x[4] - x[1], 2) + pow(y[4] - y[1], 2);
146. **if** (p[1] == p[2] && p[2] == p[3] && p[3] == p[4] && p[4] == p[1]) {
147. equ\_line = 2;
148. } **else** **if** (p[1] == p[3] && p[2] == p[4]) {
149. equ\_line = 1;
150. }
152. **return** equ\_line;
153. }
155. //Являются ли диагонали равными
156. **int** find\_is\_equ\_diag(**double** x[5], **double** y[5]) {
157. **int** is\_equ\_diag = 0, diag1 = 0, diag2 = 0;
159. //Длины диагоналей
160. diag1 = pow(x[1] - x[3], 2) + pow(y[1] - y[3], 2);
161. diag2 = pow(x[2] - x[4], 2) + pow(y[2] - y[4], 2);
163. **if** (diag1 == diag2) {
164. is\_equ\_diag = 1;
165. }
167. **return** is\_equ\_diag;
168. }

**Пояснения к программе.**

Мною были использованы массивы для записи 4ех координат фигуры. Это было сделано с целью более удобного хранения переменных. Вместо ввода восемь переменных (x1, x2, x3, x4, y1, y2, y3, y4) я ввёл два массива (x[5], y[5]). Я ввёл массивы на 5 элементов при наличии всего 4ех ради удобства написания кода. Так при объявление массивов на 5 элементов индексы массивов будут соответствовать номеру порядка координаты. (x[1] – Первая точка, x[2] – Вторая и тд.)

Мною также были использованы 4 функции. Я их ввёл в программу ради дальнейшего удобства чтения кода. Поиск всех нужных свойств фигуры осуществляются вне главной функции main, в которой реализован алгоритм определения фигуры по найденным свойствам. По сути, эти 4 функции и функция main – отдельные алгоритмы. В каждой функции есть свои локальные переменные, которые удаляются после их использования. Это оптимизирует программу по памяти.

Ниже представлен код без введения функций (и массивов). Как можно заметить – ориентироваться в таком коде очень проблематично, в начале кода объявлено очень много переменных, а логика отдельных блоков кода не понятна без комментарий, что противоположно в случае объявления функций.

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. **int** main() {
6. **int** is\_equ\_diag = 0, equ\_line = 0, paral\_line = 0, vipuc = 0, result;
7. **double** x1, x2, x3, x4, y1, y2, y3, y4, p1, p2, p3, p4, x1\_t, x2\_t, y1\_t, y2\_t;
9. system("chcp 1251 > null");
11. printf("Введите координаты 4-ех точек\n");
13. printf("Первая точка: ");
14. scanf("%lf%lf", &x1, &y1);
16. printf("Вторая точка: ");
17. scanf("%lf%lf", &x2, &y2);
19. printf("Третья точка: ");
20. scanf("%lf%lf", &x3, &y3);
22. printf("Четвертая точка: ");
23. scanf("%lf%lf", &x4, &y4);
25. result = 0;
27. x1\_t = x2 - x1;
28. y1\_t = y2 - y1;
29. x2\_t = x3 - x2;
30. y2\_t = y3 - y2;
31. result += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
33. x1\_t = x3 - x2;
34. y1\_t = y3 - y2;
35. x2\_t = x4 - x3;
36. y2\_t = y4 - y3;
37. result += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
39. x1\_t = x4 - x3;
40. y1\_t = y4 - y3;
41. x2\_t = x1 - x4;
42. y2\_t = y1 - y4;
43. result += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
45. x1\_t = x1 - x4;
46. y1\_t = y1 - y4;
47. x2\_t = x2 - x1;
48. y2\_t = y2 - y1;
49. result += x1\_t \* y2\_t - x2\_t \* y1\_t > 0 ? 1 : -1;
51. vipuc = abs(result) == 4 ? 1 : 0;
53. result = 0;
55. **if** (x2 - x1 != 0 && y2 - y1 != 0 && x4 - x3 != 0 && y4 - y3 != 0) {
56. **if** ((x2 - x1) / (y2 - y1) == (x4 - x3) / (y4 - y3)) {
57. result++;
58. }
59. } **else** **if** ((x2 - x1 == x4 - x3 && x2 - x1 == 0) || (y2 - y1 == y4 - y3 && y2 - y1 == 0)) {
60. result++;
61. }
63. **if** (x3 - x2 != 0 && y3 - y2 != 0 && x1 - x4 != 0 && y1 - y4 != 0) {
64. **if** ((x3 - x2) / (y3 - y2) == (x1 - x4) / (y1 - y4)) {
65. result++;
66. }
67. } **else** **if** ((x3 - x2 == x1 - x4 && x3 - x2 == 0) || (y3 - y2 == y1 - y4 && y3 - y2 == 0)) {
68. result++;
69. }
71. paral\_line = result;

74. p1 = pow(x1 - x2, 2) + pow(y1 - y2, 2);
75. p2 = pow(x2 - x3, 2) + pow(y2 - y3, 2);
76. p3 = pow(x3 - x4, 2) + pow(y3 - y4, 2);
77. p4 = pow(x4 - x1, 2) + pow(y4 - y1, 2);
79. **if** (p1 == p2 && p2 == p3 && p3 == p4 && p4 == p1) {
80. result = 2;
81. } **else** **if** (p1 == p3 && p2 == p4) {
82. result = 1;
83. }
85. equ\_line = result;
87. result = 0;
89. **if** (pow(x1 - x3, 2) + pow(y1 - y3, 2) == pow(x2 - x4, 2) + pow(y2 - y4, 2)) {
90. result = 1;
91. }
93. is\_equ\_diag = result;
95. **if** (vipuc == 1) {
96. **if** (paral\_line == 2) {
97. **if**(is\_equ\_diag) {
98. **if** (equ\_line == 2) {
99. printf("Квадрат");
100. } **else** **if** (equ\_line == 1) {
101. printf("Прямоугольник");
102. }
103. } **else** {
104. **if** (equ\_line == 1) {
105. printf("Параллелограмм");
106. } **else** {
107. printf("Произвольная фигура");
108. }
109. }
110. } **else** **if** (paral\_line == 1) {
111. printf("Трапеция");
112. } **else** {
113. printf("Произвольная фигура");
114. }
115. } **else** {
116. printf("Проивольная фигура");
117. }
119. printf("\n");
120. system("pause");
121. **return** 0;
123. }

Для организации дружественного интерфейса пользователя перед вводом данных выводится сообщение для пользователя. Ввод и вывод организован при помощи стандартных функций scanf() и printf(), для этого подключен заголовочный файл stdio.h.

Заголовочный файл stdlib.h необходим для использования функции system(), чтобы приостановить выполнение программы перед завершением (выполняется команда «pause» операционной системы).

Заголовочный файл math.h необходим для использования функции pow().

**Выводы:**

Для решения этой задачи я использовал конструкция условий типа

1. **if** (condition) {
2. code;
3. } **else** **if** (condidion2) {
4. code;
5. } **else** {
6. code;
7. }

В ходе решения я использовал такие операторы как «И» (&&),«ИЛИ» (||) и оператор тождества (==). Используя эти два оператора, я составлял сложные условия, состоящие из более простых условий.