МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра систем автоматизированного проектирования

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: «Реализация программы с условными конструкциями»

Студентка гр. 3352 _____ Калюжная М.И.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург

2023

Цель работы

Изучить ветвление и условный оператор if с использованием разных библиотек, форматированный вывод. Практическое использование вещественного типа данных в языке программирования C++.

Исходная формулировка

Составить программу, которая определяет, принадлежит ли точка с координатами (x, y) одной из заштрихованных областей рисунка, периодически повторяющегося вдоль оси OX.

Анализ задания

В исходном графике дано изображение креста, вписанного в квадрат со стороной H. В каждом случае будет происходить проверка вхождения координаты y в область; в случае ввода x, большего периода графика, будет находиться «истинное» значение и проверяться условие для него. При неотрицательных параметрах H и a фигура представляет собой крест с тремя заштрихованными областями; все треугольники в фигуре равнобедренные, имеют основание a и боковые стороны, равные a/√(2), причем (H>=a). При H = 0 и a = 0 фигура «схлопывается» в точку с координатами (0,0), которая бесконечно множится вдоль оси OX. При a = 0 и H ≠ 0 фигура принимает вид как на (рис. 1) для 1-й и 4-й четверти и (рис. 2) для 2-й и 3-й четверти. При a = H и H ≠ 0 фигура принимает вид как на (рис. 3) для 1-й и 4-й четверти и (рис. 4) для 2-й и 3-й четверти.

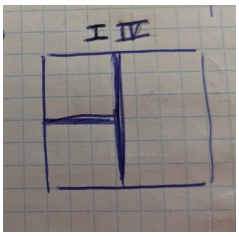


Рис. 1

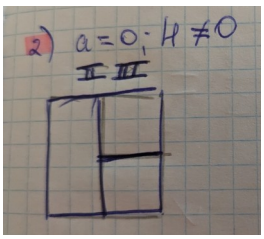


Рис.2

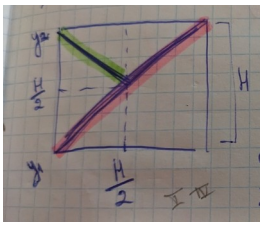


Рис.3

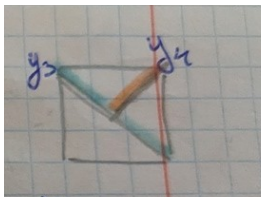


Рис.4

Формальная постановка задачи

Дано: фигура, вписанная в квадрат стороной H, все треугольники в фигуре равнобедренные, имеют основание a и боковые стороны, равные a/√(2), причем (H>=a);
Найти: принадлежит ли точка с координатами (x, y) одной из заштрихованных областей рисунка, периодически повторяющегося вдоль оси OX.

Способ решения:

1) Введём переменную t, при помощи которой задается изначальное положение фигуры в одной из четвертей.

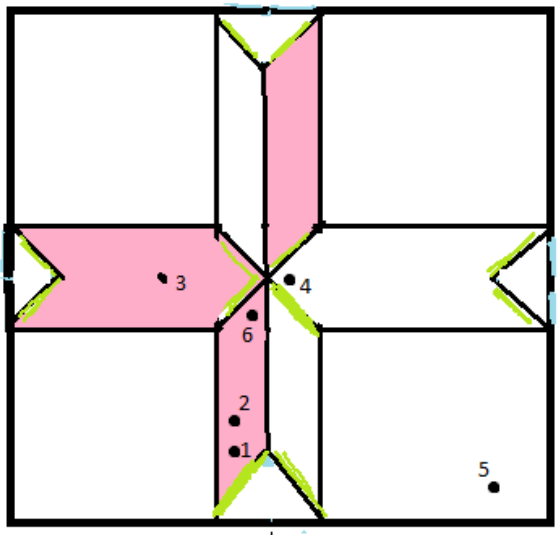
2) Проверим введенные значения H, t и a на выполнение условий: H >= a >= 0 и 1 <= t <= 4.
3) Проверим координату x на знак и нормализируем ее с помощью функции x = std::fmod(x, H) при x > 0 или с помощью x = H - std::fmod(std::fabs(x), H) при x < 0.
4) Возьмем модуль y и проверим условие H < std::fabs(y)
5) Рассмотрим случаи размещения начальной фигуры в разных четвертях:

Случай 1 (1-я четверть и 4-я четверть)

- Т.к. берется модуль y, это позволяет обработать сразу 2 четверти
- Проверим ситуацию (H = 0 && a = 0)
- Проверим ситуацию (H ≠ 0 && a = 0)
- Проверим ситуацию (H = a && a ≠ 0)
- Проверим выполнение условий попадания в нижнюю закрашенную область: ((x > (H / 2.0 - a / 2.0)) && (x < (H / 2.0)) && (y < (H / 2.0 - a / 2.0))) и (y > x + -1*(H/2.0 — a/2.0))
- Проверим выполнение условий попадания в верхнюю закрашенную область: ((x > H / 2.0) && (x < H / 2.0 + a/2.0) && ((y < H) && (y > H/2.0))) и ((y < x + (H / 2.0 - a / 2.0)) && (y > x))
- Проверим выполнение условий попадания в левую закрашенную область ((x > 0) && (x < H / 2) && (y > H/2.0 - a/2.0) && y < H / 2.0 + a / 2.0) и невыполнение условий промаха: (((x > H / 2.0 - a / 2.0) && (x < H / 2)) && (y > -1 * x + H)) и (((x > 0) && (x < a / 2)) && ((y > x + (H / 2.0 - a / 2.0)) && (y < -1*x + H / 2.0 + a / 2.0)))

Случай 2 (2-я и 3-я четверть)

- Т.к. берется модуль y, это позволяет обработать сразу 2 четверти
- Проверим ситуацию (H = 0 && a = 0)
- Проверим ситуацию (H ≠ 0 && a = 0)
- Проверим ситуацию (H = a && a ≠ 0)
- Проверим выполнение условий попадания в верхнюю закрашенную область: ((x >= H / 2 - a / 2) && (x <= H / 2) && (H / 2 + a / 2 <= y <= H)) и (y <= -1 * x + (H + H / 2 - a / 2))
- Проверим выполнение условий попадания в нижнюю закрашенную область: ((H / 2 <= x <= H / 2 + a / 2) && (y < H / 2)) и (y >= -1*x + H/2 + a/2)
- Проверим выполнение условий попадания в правую закрашенную область: ((H / 2 <= x <= H) && (H / 2 <= y <= H / 2 + a / 2)) и ((y <= x) && (y >= x - (H / 2 - a / 2))) или ((H / 2 + a / 2 <= x <= H) && (H / 2 - a / 2 <= y <= H / 2)) и (y <= -1*x + H + H/2 - a/2)



Контрольный пример

Рассмотрим случай, когда фигура находится в 1-й четверти. Пусть H = 10, a = 5, t = 1.

Точка 1 с координатами (4;2) попадает в нижнюю закрашенную область.

Точка 2 с координатами (4;3) попадает в нижнюю закрашенную область.

Точка 3 с координатами (3;5) попадает в левую закрашенную область.

Точка 4 с координатами (6;5) не попадает в закрашенную область.

Точка 5 с координатами (9;1) не попадает в закрашенную область.

Точка 6 с координатами (4,57;4,57) попадает в нижнюю закрашенную область.

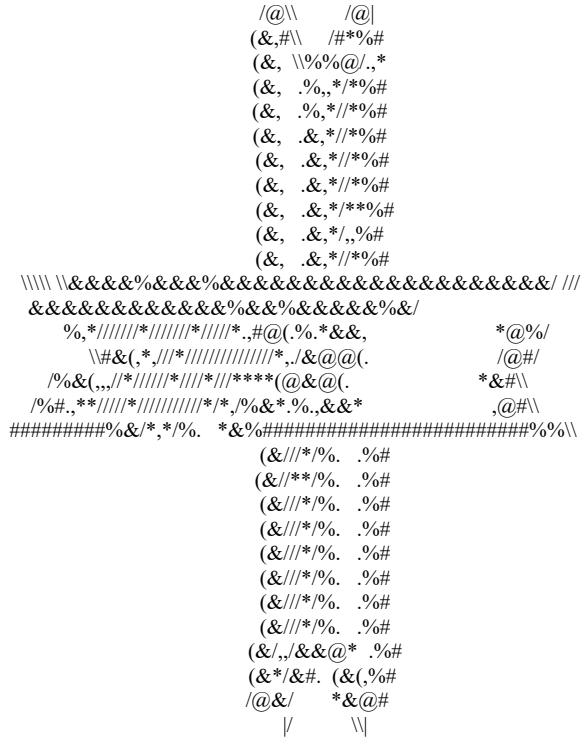
Формат хранения данных

Тип	Имя	Назначение
float	a, H, x, y	Исходное значения параметров и координат точки
	t	Номер четверти, в которой находится начальная фигура

Ограничение, условленное исполнением на компьютере

float – тип данных с плавающей точкой (вещественный). Диапазон: от 3,4E–38 до 3,4E+38

Макеты ввода/вывода

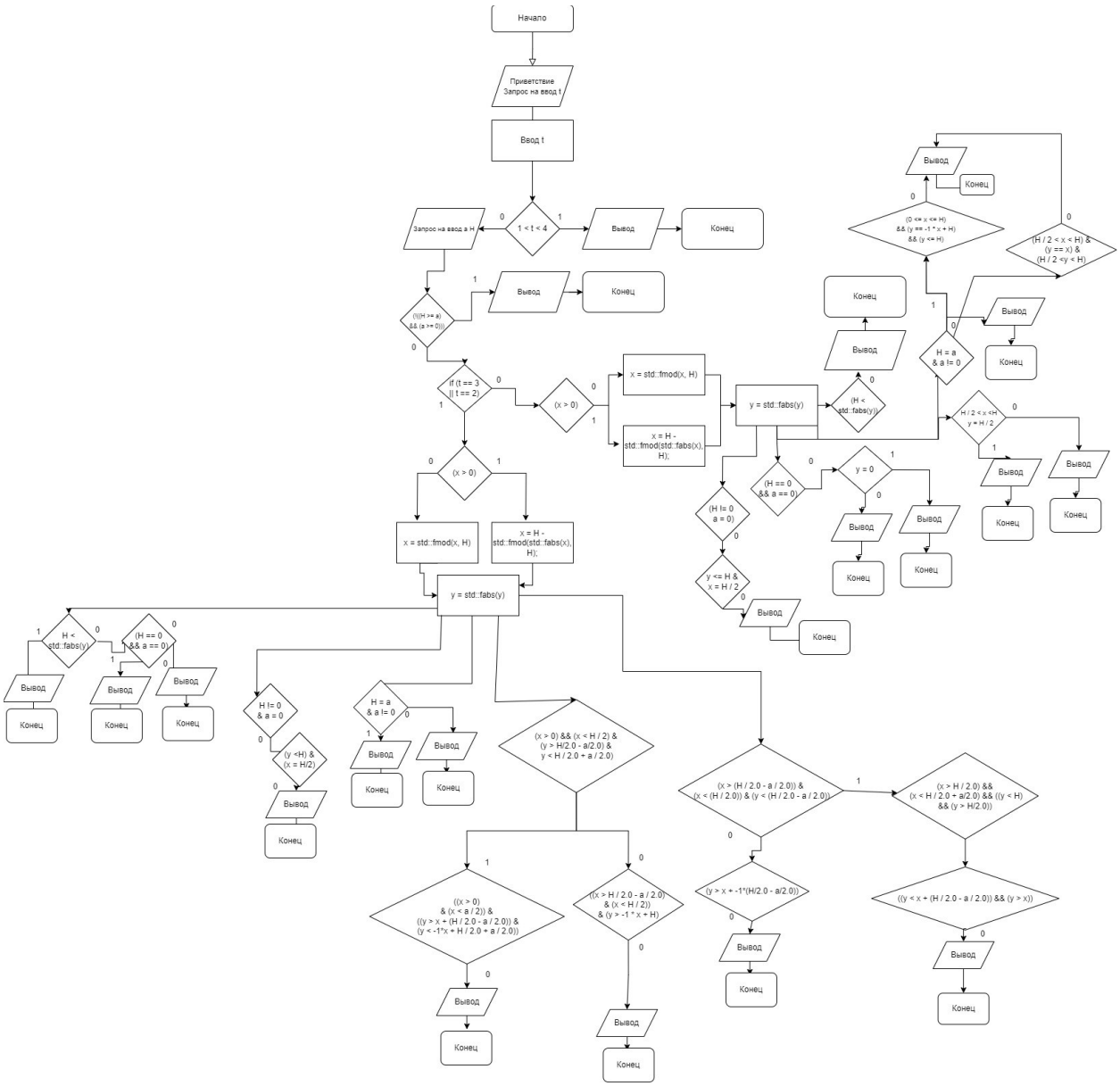
1. Приветствие	Приветствую! Это лабораторная работа №2 Калюжной М.И. группа 3352 Вариант 1_25_Б начало работы 19.09.23, конец 28.09. 
2. Запрос ввода переменной	Введите параметры a, H, t:
3. Ввод	a H t
4. Запрос ввода переменной	Введите координаты точки x, y:

5. Ввод	x y
6. Результат	• Данная точка не принадлежит ни одной из заштрихованных областей. • Данная точка принадлежит одной из заштрихованных областей :)

Средства обеспечения ввода/вывода

Библиотека	Команды
iostream	cout, cin
math.h	fabs
locale.h	setlocale

Алгоритм решения



Программа

```
#include <iostream>
#include <math.h>
#include <locale.h>
int main(int argc, char *arv[])
{
   setlocale(LC_ALL, "Russian");

   std::cout << "Приветвую! Это лабораторная работа №2 Калюжной М.И. группа 3352 Вариант 1_25_Б  начало работы 19.09.23, конец
28.09." << std::endl;
   // tut figura simwolami
   std::cout << "\n";

                                    std::cout << "                    /@\\        /@|                  " << std::endl;
                                    std::cout << "                   (&,#\\     /#*%#                   " << std::endl;
                                   std::cout << "                  (&,  \\%%@/.,*%#                     " << std::endl;
                                    std::cout << "                   (&,   .%,,*/*%#                   " << std::endl;
                                    std::cout << "                   (&,   .%,*//*%#                   " << std::endl;
                                    std::cout << "                   (&,   .&,*//*%#                   " << std::endl;
                                    std::cout << "                   (&,   .&,*//*%#                   " << std::endl;
                                    std::cout << "                   (&,   .&,*//*%#                   " << std::endl;
                                   std::cout << "                   (&,   .&,*/**%#                    " << std::endl;
                                    std::cout << "                   (&,   .&,*/,,%#                   " << std::endl;
                                    std::cout << "                   (&,   .&,*//*%#                   " << std::endl;
   std::cout << "      \\&&&&%&&&%&&&&&&&&&&&&&&&&&&&&&&&/   .&,**,/&&%&&&&&&&&&&&&&&&&%&&%&&&&&%&/
                                                                                                  " << std::endl;
                                 std::cout << "        \\#%,*///////*///////*////*.,#@(.%.*&&,              *@%/     " << std::endl;
                                   std::cout << "          \\#&(,*,///*/////////////*,./&@@(.                /@#/       " << std::endl;
                                  std::cout << "          /%&(,,,//*//////*////*///****(@&@(.                *&#\\       " << std::endl;
                                  std::cout << "        /%#.,**/////*//////////*/*,/%&*.%.,&&*                ,@#\\     " << std::endl;
            std::cout << "      /%%%##########################%&/*,*/%.   *&%##########################%%\\\   " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&//**/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                    std::cout << "                   (&///*/%.   .%#                    " << std::endl;
                                   std::cout << "                  (&/,,/&&@*  .%#                      " << std::endl;
                                   std::cout << "                   (&*/&#.  (&(,%#                     " << std::endl;
                                   std::cout << "                    /@&/       *&@#                     " << std::endl;
                                       std::cout << "                      |/          \\|                  " << std::endl;

   float t;
   std::cout << "\n Данная фигура вписана в квадрат с высотой H, все треуголики в фигуре равнобедренные, имеют основание a и
боковые стороны, равные a/sqrt(2), причем (H>=a)." << std::endl;
   std::cout << "Введенные значения a и H должны удовлетворять таким условиям: H >= a >= 0 \n";
   std::cout << "\n";
   std::cout << "Введите пожалуйста номер четверти, в которой находится фигура:" << std::endl;
   std::cin >> t;
   if (t > 4 || t < 1)
   {
      std::cout << " Введенные значения не удовлетворяют начальным условиям.";
      return 1;
   }
   std::cout << "\n";
   std::cout << "Номер четверти, в которой находится фигура: " << t << std::endl;
   std::cout << "\n";
   std::cout << "Введите H, a, x, y:" << std::endl;

   // считать параметры с пользователя: H, a, x, y (float)
   float H, a, x, y;
   std::cin >> H;
   std::cin >> a;
   std::cin >> x;
   std::cin >> y;

   // проверка введенных значений на то что H >= a >= 0
   if (!((H >= a) && (a >= 0)))
   {
      std::cout << " Введенные значения не удовлетворяют начальным условиям.";
```

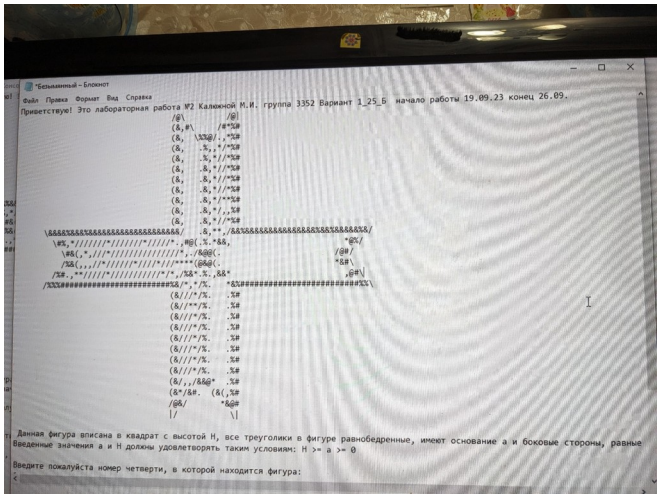
```
    return 1;
}

// начинаю обработку 2 и 3 четвертей

if (t == 3 || t == 2)
{
    if (x > 0)
    {
        x = std::fmod(x, H);
    }
    else
    {
        x = H - std::fmod(std::fabs(x), H);
    }
    y = std::fabs(y);
    if (H < std::fabs(y))
    {
        std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
        return 1;
    }
    if (H == 0 && a == 0)
    {
        if (y == 0)
        {
            std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
            return 0;
        }
        else
        {
            std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
            return 1;
        }
    }
    if (H != 0 && a == 0)
    {
        if ((y <= H) && (x == H / 2))
        {
            std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
            return 0;
        }

        if ((H / 2 <= x <= H) && (y == H / 2))
        {
            std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
            return 0;
        }
        else
        {
            std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
            return 1;
        }
    }
    if (H == a && a != 0)
    {
        if ((0 <= x <= H) && (y == -1 * x + H) && (y <= H))
        {
            std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
            return 0;
        }
        if ((H / 2 <= x <= H) && (y == x) && (H / 2 <= y <= H))
        {
            std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
            return 0;
        }
        else
        {
            std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
            return 1;
        }
    }
    if ((x > (H / 2.0 - a / 2.0)) && (x < (H / 2.0)) && (y < (H / 2.0 - a / 2.0))) // входит в зеленый прямоугольник
    {
        // надо убедиться что не входит в треугольничек y = k*x + b, где b == - (H/2 - a/2), k == 1
        if (y > x + -1 * (H / 2.0 - a / 2.0))
        {
            std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
            return 1;
        }
```

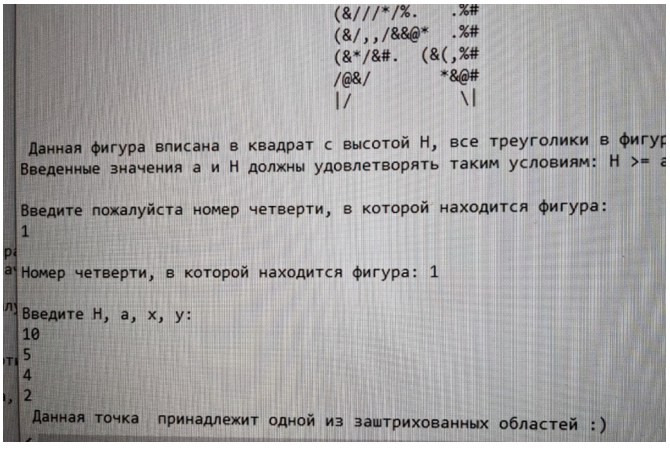
```
}
if ((x > H / 2.0) && (x < H / 2.0 + a / 2.0) && ((y < H) && (y > H / 2.0))) // входит в красный прямоугольник
{
   if ((y < x + (H / 2.0 - a / 2.0)) && (y > x))
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 0;
   }
}
if ((x > 0) && (x < H / 2) && (y > H / 2.0 - a / 2.0) && y < H / 2.0 + a / 2.0)
{
   if (((x > H / 2.0 - a / 2.0) && (x < H / 2)) && (y > -1 * x + H))
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
   else
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }

   if (((x > 0) && (x < a / 2)) && ((y > x + (H / 2.0 - a / 2.0)) && (y < -1 * x + H / 2.0 + a / 2.0)))
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
}

// попадание в верхний лепесток
if ((x >= H / 2 - a / 2) && (x <= H / 2) && (H / 2 + a / 2 <= y <= H))
{
   if (y <= -1 * x + (H + H / 2 - a / 2))
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
   else
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
}

// попадание в нижний лепесток (малиновый)
if ((H / 2 <= x <= H / 2 + a / 2) && (y < H / 2))
{
   if (y >= -1 * x + H / 2 + a / 2)
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
   else
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
}
if ((H / 2 <= x <= H) && (H / 2 <= y <= H / 2 + a / 2))
{
   if ((y <= x) && (y >= x - (H / 2 - a / 2)))
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
   else
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
}
if ((H / 2 + a / 2 <= x <= H) && (H / 2 - a / 2 <= y <= H / 2))
{
   if (y <= -1 * x + H + H / 2 - a / 2)
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
   else
```

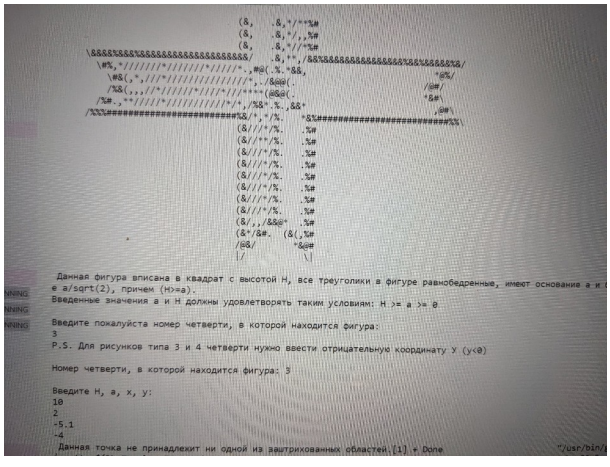
```
        {
            std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
            return 1;
        }
    }

    std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
    return 1;
}

// обработка 2 и 3 четыертей завершена, далее код предназначен для 1 и 4 четвертей

if (x > 0)
{
    x = std::fmod(x, H);
}
else
{
    x = H - std::fmod(std::fabs(x), H);
}
// обрабатываем отрицательные значения y (вариант с отражением фигуры относительно оси X)
y = std::fabs(y);

if (H < std::fabs(y))
{
    std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
    return 1;
}
if (H == 0 && a == 0)
{
    if (y == 0)
    {
        std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
        return 0;
    }
    else
    {
        std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
        return 1;
    }
}
if (H != 0 && a == 0)
{
    if ((y <= H) && (x == H / 2))
    {
        std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
        return 0;
    }

    if ((0 <= x <= H / 2) && (y == H / 2))
    {
        std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
        return 0;
    }
    else
    {
        std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
        return 1;
    }
}

if (H == a && a != 0)
{
    if ((x <= H) && (y == x))
    {
        std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
        return 0;
    }
    if ((0 <= x <= H / 2) && (y == H - x) && (H / 2 <= y <= H))
    {
        std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
        return 0;
    }
    else
    {
        std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
        return 1;
    }
}
```

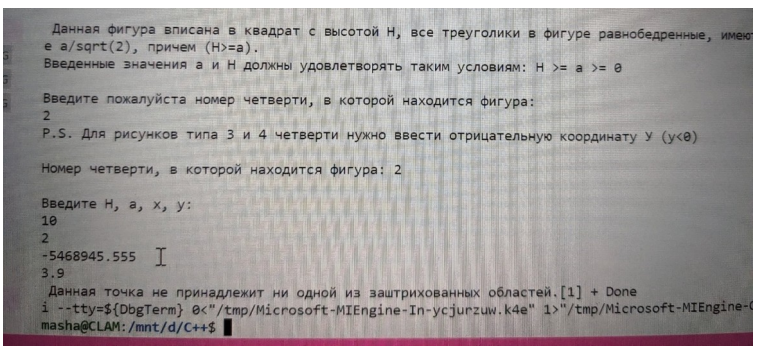
```
if ((x > (H / 2.0 - a / 2.0)) && (x < (H / 2.0)) && (y < (H / 2.0 - a / 2.0))) // входит в зеленый прямоугольник
{
   // надо убедиться что не входит в треугольничек y = k*x + b, где b == - (H/2 - a/2), k == 1
   if (y > x + -1 * (H / 2.0 - a / 2.0))
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
}
//     попасть в красную или
if ((x > H / 2.0) && (x < H / 2.0 + a / 2.0) && ((y < H) && (y > H / 2.0))) // входит в красный прямоугольник
{

   if ((y < x + (H / 2.0 - a / 2.0)) && (y > x))
   {
      std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
      return 0;
   }
}

if ((x > 0) && (x < H / 2) && (y > H / 2.0 - a / 2.0) && y < H / 2.0 + a / 2.0)
{
   if (((x > H / 2.0 - a / 2.0) && (x < H / 2)) && (y > -1 * x + H))
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
   if (((x > 0) && (x < a / 2)) && ((y > x + (H / 2.0 - a / 2.0)) && (y < -1 * x + H / 2.0 + a / 2.0)))
   {
      std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
      return 1;
   }
   std::cout << " Данная точка  принадлежит одной из заштрихованных областей :)";
   return 0;
}

std::cout << " Данная точка не принадлежит ни одной из заштрихованных областей.";
return 1;
}
```

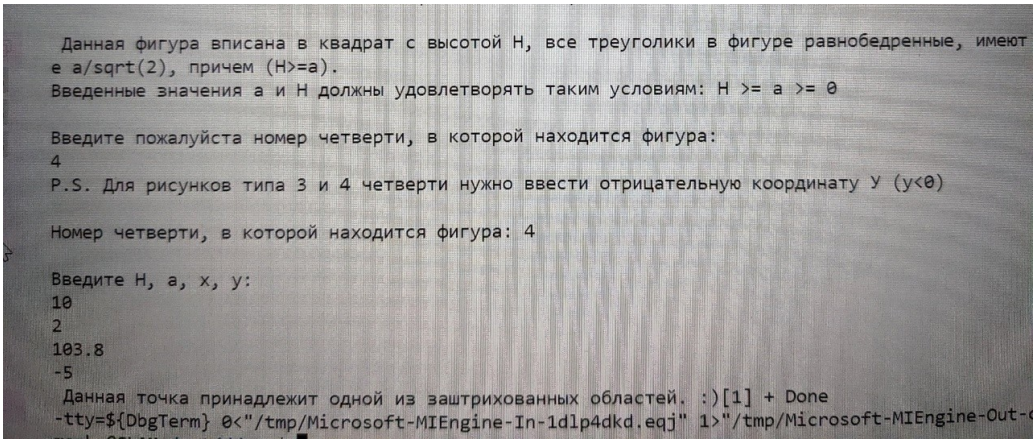
Результаты работы программы

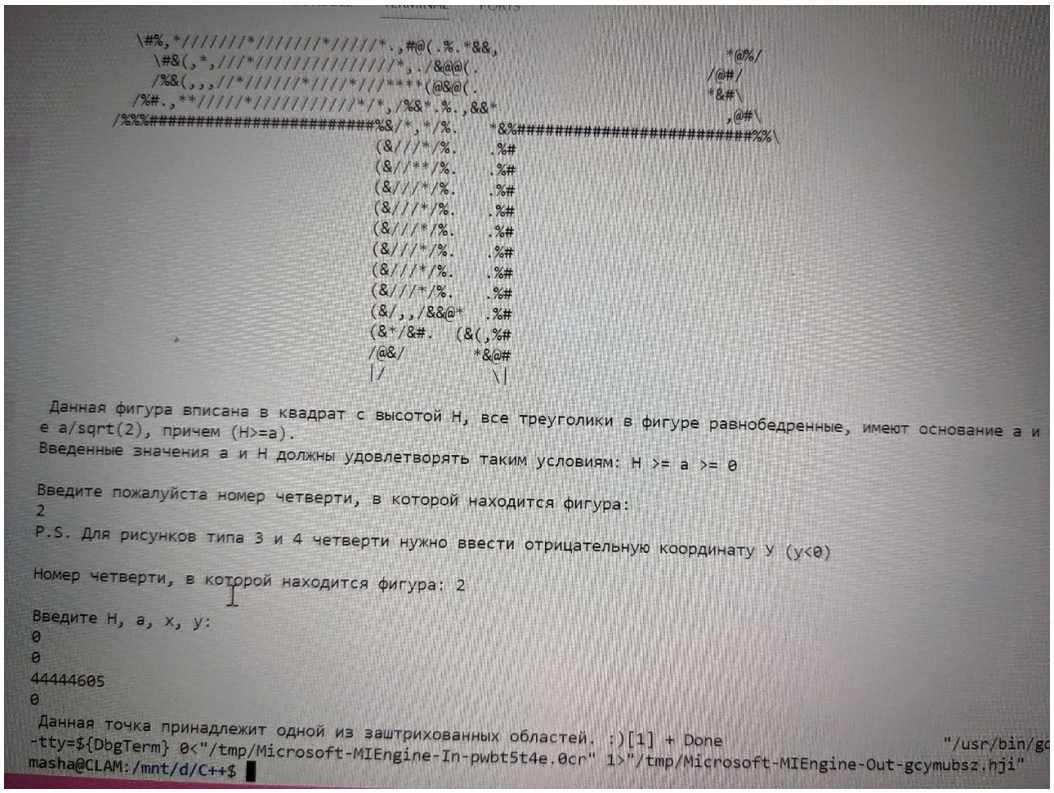












Вывод о проделанной работе

Программа работает исправно. В процессе работы я изучила новые библиотеки: math.h. Изучила команду fabs и познакомилась с инструкцией организации ветвления программы: if, else, else if.