**Министерство образования и науки**

Белгородский государственный технологический университет

им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**КУРСОВАЯ РАБОТА**

по дисциплине «Основы программирования»

тема: «Построение многоугольника с вершинами в заданных точках»

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Браткова И.О.

(подпись) ПВ-21

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Брусенцева В.С.

(подпись)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Белгород

2017 г.

Оглавление

[Введение 3](#_Toc499846840)

[Постановка задачи 6](#_Toc499846841)

[Обоснование выбора решения задачи 6](#_Toc499846842)

[Описание структур данных 7](#_Toc499846843)

[Алгоритм в укрупнённых блоках 8](#_Toc499846844)

[Описание функций 9](#_Toc499846845)

[Тестовые данные 14](#_Toc499846846)

[Разбиение программы на модули 15](#_Toc499846847)

[Заголовочные файлы 15](#_Toc499846848)

[Заключение 16](#_Toc499846849)

[Список используемой литературы: 17](#_Toc499846850)

[Приложения 18](#_Toc499846851)

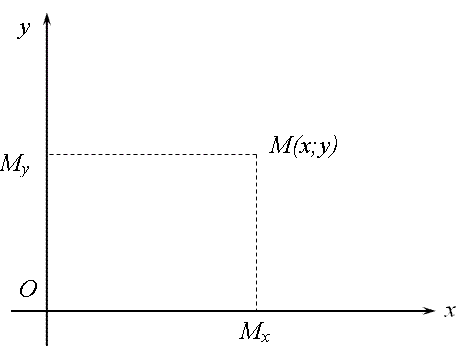
# Введение

Задача, которую требуется решить, тесно связана с геометрическими понятиями: плоскостью, пространством, уравнениями прямой и плоскости. Для того, чтобы перейти к ее решению, необходимо изучить теоретическую часть этого вопроса.

Для начала необходимо понять, что такое плоскость.

Две перпендикулярные оси на плоскости с общим началом и одинаковой масштабной единицей образуют декартову прямоугольную систему координат на плоскости.

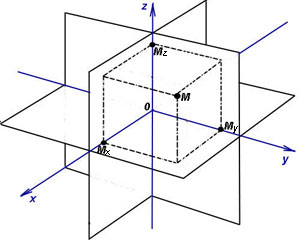
Одна из осей называется осью Ox, или осью абсцисс, другая – осью Oy, или осью ординат. Эти оси называют также координатными осями, где Mx и My соответственно проекции произвольной точки M плоскости на оси Ox и Oy. (рис 1)[4].



**Рисунок 1. Прямоугольная декартова система координат**

Декартовы координаты в пространстве вводятся в полной аналогии с декартовыми координатами на плоскости. Три взаимно перпендикулярные оси в пространстве с общим началом O и одинаковой масштабной единицей образуют декартову прямоугольную систему координат в пространстве.

Одну из указанных осей называют осью Ox, или осью абсцисс, другую - осью Oy, или осью ординат, третью - осью Oz, или осью аппликат. Пусть Mx, My, Mz - проекции произвольной точки М пространства на оси Ox, Oy и Oz соответственно(рис 2.)[5].



**Рисунок 2. Декартова система координат в пространстве**

Простейшими и основными геометрическими фигурами в трехмерном пространстве являются точка, прямая и плоскость.

Представление о плоскости в пространстве позволяет получить, к примеру, поверхность стола или стены[1]. Однако, стол или стена имеют конечные размеры, а плоскость простирается за их границы в бесконечность.

Точки и прямые в пространстве обозначаются также как и на плоскости – большими и маленькими латинскими буквами соответственно. Например, точки А и Q, прямые а и d. Если заданы две точки, лежащие на прямой, то прямую можно обозначить двумя буквами, соответствующими этим точкам. К примеру, прямая АВ или ВА проходит через точки А и В. Плоскости принято обозначать маленькими греческими буквами (рис. 3).



**Рисунок 3. Варианты изображения плоскости на чертеже**

Пусть в трехмерном пространстве зафиксирована прямоугольная система координат Oxyz и задана плоскость.

Плоскость, как и любая другая геометрическая фигура, состоит из точек. В прямоугольной системе координат Oxyz каждой точке соответствует упорядоченная тройка чисел – координаты точки.

Между координатами каждой точки плоскости можно установить зависимость с помощью уравнения, которое называют уравнением плоскости[3].

Уравнение плоскости в прямоугольной системе координат Oxyz в трехмерном пространстве – это уравнение с тремя переменными x, y и z, которому удовлетворяют координаты любой точки заданной плоскости и не удовлетворяют координаты точек, лежащих вне данной плоскости.

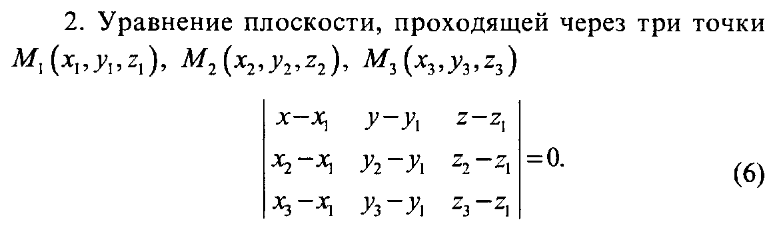
Таким образом, уравнение плоскости обращается в тождество при подстановке в него координат любой точки плоскости. Если в уравнение плоскости подставить координаты точки, не лежащей в этой плоскости, то оно обратится в неверное равенство.

Общее уравнение плоскости было введено Л. О. Гессе в 1861 году, хотя впервые упоминания об этом уравнении можно встретить в работах А. К. Клеро еще в 1731 году. Итак, выглядит это уравнение следующим образом: Ax + By + Cz + D = 0 где: A, B, C и D – это постоянные числа, причем, первые три из них одновременно не равны 0.

Случаи расположения плоскостей:

* Если A=0, то плоскость параллельна оси Ox;
* Если B=0, то плоскость параллельна оси Oy;
* Если C=0, то плоскость параллельна оси Oz;
* Если D=0, то плоскость проходит через начало координат.
* Если A=B=0, то плоскость параллельна плоскости Oxy;
* Если B=C=0, то плоскость параллельна плоскости Oyz;
* Если A=C=0, то плоскость параллельна плоскости Oxz.

Уравнение плоскости по трем точкам

****

**Рисунок 4. Уравнение плоскости по трём точкам**

Для того, чтобы проверить, принадлежат ли остальные точки плоскости, необходимо подставлять их координаты в полученное уравнение[2].

# Постановка задачи

Разработать алгоритм и написать модульную программу на языке Си для определения возможности соединения n точек в пространстве так, чтобы они образовывали многоугольник.

Ввод производится с клавиатуры или из файла, в зависимости от желания пользователя.

# Обоснование выбора решения задачи

Для решения задачи был выбран вариант решения, который заключается в следующем:

Пользователь вводит количество точек (или оно считывается из файла), если количество точек меньше трёх, то программа завершается, т.к. на двух точках невозможно построить замкнутый многоугольник.

Если количество точек больше или равно трем, то проводится их проверка на принадлежность одной прямой. В случае, если точки не лежат на одной прямой составляется уравнение плоскости. После этого проверяется принадлежность всех точек данной плоскости. Если точки не лежат в одной плоскости, то программа завершает работу, в противном случае, происходит переход к следующему шагу.

При положительном результате проверки выполняется проецирование данных точек на одну из координатных плоскостей, не перпендикулярную полученной плоскости.

На следующем этапе производится поиск двух точек: с минимальными координатами x и y, а так же точки с максимальными координатами. По этим двум точкам составляется уравнение прямой Ax+By+C=0, относительно которой будет определяться положение точек при соединении их ломанной линией.

Таким образом, процесс получения многоугольника представляет собой сортировку массива данных точек против часовой стрелки относительно прямой, заданной уравнением Ax+By+C=0.

Если координаты двух точек равны, то обмен не производится. Если обе точки находятся под прямой, то обмен происходит в случае, когда координаты первой точки больше координат второй точки. В случае, когда точки лежат над прямой, обмен будет происходить тогда, когда координаты второй точки будут меньше координат первой точки.

# Описание структур данных

Структура tochka, полями которой являются координаты x, y, z:

**typedef struct**

**{**

**float x, y, z;**

**} tochka;**

Массив точек в динамике:

**tochka\*t;**

В ходе решения задачи предполагается проецирование точек на какую-либо плоскость с учётом отказа от одной из координат, для чего была создана структура tochka\_xy с полями xn, yn:

**typedef struct**

**{**

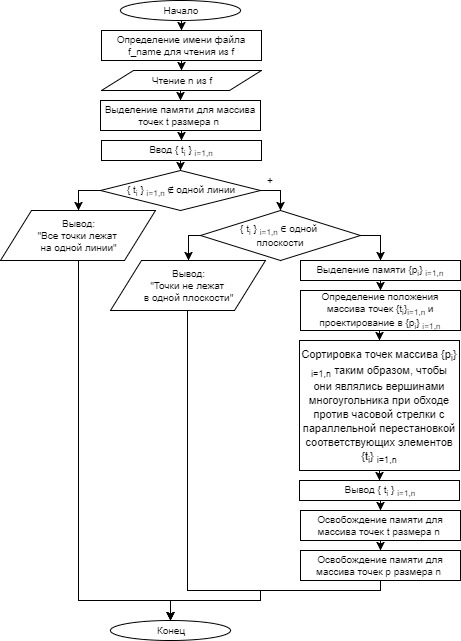
**float xn, yn;**

**} tochka\_xy;**

Массив точек данного типа в динамике:

**tochka\_xy\* p;**

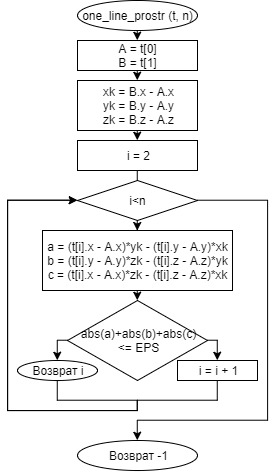
# Алгоритм в укрупнённых блоках



# Описание функций

1. Заголовок: int one\_line\_prostr (tochka \*t, size\_t n);

Назначение: Возвращает номер первой точки, которая не лежит со всеми точками массива t размера n на одной линии. Ложь в противном случае.



1. Заголовок: void toch\_new (int or, tochka \*t, tochka\_xy \*p, size\_t n);

Назначение: Производит перезапись массива точек t размера n в массив точек p размера n в зависимости от значение or.

1. Заголовок: float \*urav (tochka \*t, int nom, size\_t n);

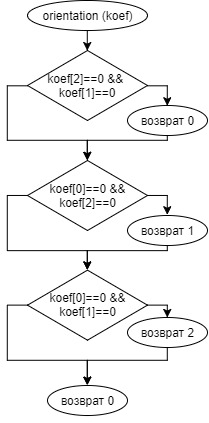
Назначение: Возвращает массив коэффициентов уравнения плоскости Ax+By+Cz+D.

1. Заголовок: int check (tochka \*t, size\_t n, float \*koeff);

Назначение: Возвращает значение истина, если точки лежат в одной плоскости и ложь в противном случае.

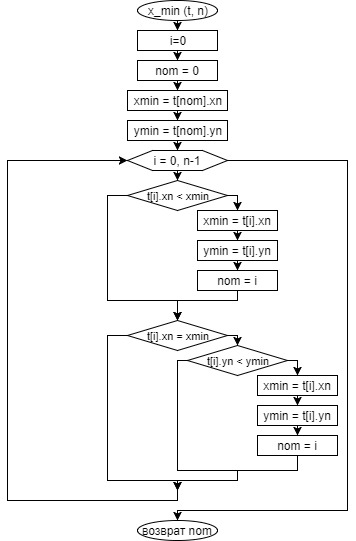
1. Заголовок: int orientation (float \*koeff, size\_t n);

Назначение: Возвращает 1, если плоскость перпендикулярна Oy. 2, если перпендикулярна Oz. 0, если перпендикулярна Oх или не перпендикулярна ни одной из плоскостей.



1. Заголовок: int x\_min (tochka\_xy \*t, size\_t n);

Назначение: Возвращает индекс минимальной точки массива t размера n.



1. Заголовок: int x\_max (tochka\_xy \*t, size\_t n);

Назначение: Возвращает индекс максимальной точки массива t размера n.

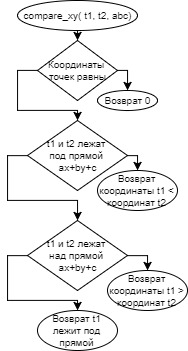
(Блок-схема аналогична функции x\_min).

1. Заголовок: float\* abc\_xy(tochka\_xy tmin, tochka\_xy tmax);

Назначение: Возвращает массив коэффициентов уравнения прямой Ax+By+C.

1. Заголовок: int compare\_xy(tochka\_xy t1, tochka\_xy t2, float \*abc);

Назначение: Возвращает значение истина, когда координаты точки t2 меньше координат точки t1. Ложь – в противном случае.



1. Заголовок: void sortt (tochka\_xy \*pn, tochka \*tn, size\_t n, float \*abc);

Назначение: Сортировка массивов точек pn и tn размера n в порядке против часовой стрелки.

1. Заголовок: void output (tochka \*t, size\_t n);

Назначение: Вывод массива точек t размера n на экран.

1. Заголовок: void input (tochka \*t, size\_t n);

Назначение: Ввод с клавиатуры массива точек t размера n.

1. Заголовок: void output\_xy (tochka\_xy \*t, size\_t n);

Назначение: Вывод массива точек t размера n на экран.

1. Заголовок: int kol\_zap\_file (char \*f\_name);

Назначение: Возвращает количество записей файла с именем f\_name.

1. Заголовок: void input\_file (tochka \*\*t, size\_t n, char\* f\_name);

Назначение: Ввод массива точек t размера n из файла с именем f\_name.

1. Заголовок: void output\_file (tochka \*t, size\_t n);

Назначение: Запись данных массива t типа tochka размера n в файл.

# Тестовые данные

Выбор тестовых данных происходил таким образом, чтобы проверить все возможные результаты работы программы: невозможность построение многоугольника, положение всех точек на одной прямой, нахождение точек в разных плоскостях, а так же успешную перестановку точек в порядке соединения ломаной линией.

|  |  |
| --- | --- |
| **Исходные данные** | **Ожидаемый результат** |
| Файл K1.txt | 1: -8,90; 5,00; 1,00;  2: -1,10; 2,00; 1,00;  3: 3,00; 4,20; 1,00;  4: 9,40; 3,00; 1,00;  5: 2,30; 6,00; 1,00;  6: -5,00; 7,40; 1,00; |
| 0,1 4,5 3  8 0,3 0,6  -9,9 1 4,2  15,2 99 1,09 | Точки не лежат в одной плоскости. |
| N = 2 | Многоугольник не может быть построен.  N < 3. |
| Файл K11.txt | Все точки лежат на одной прямой.  Построение невозможно. |
| Файл K9.txt | 1: 5,50; -1,00; 5,10;  2: -1,10; 1,00; 2,20;  3: 2,20; 1,00; -1,20;  4: 3,30; 1,00; -3,50;  5: 3,40; 4,00; 1,20; |

# Разбиение программы на модули

Программа состоит из двух модулей и сохранена.

Модуль outinp.h предназначен для ввода данных с клавиатуры и вывода на экран данных, а так же считывания данных из файла и записи в новый файл с результатом.

Модуль tpoint.h предназначен для работы с геометрической частью программы.

# Заголовочные файлы

***Заголовочный файл outinp.h***

#ifndef OUTINP\_H\_INCLUDED #define OUTINP\_H\_INCLUDED

typedef struct { float x, y, z;

} tochka;

typedef struct { float xn, yn;

} tochka\_xy;

void output (tochka \*t, size\_t n); //Вывод массива точек t размера n на экран.

void input (tochka \*t, size\_t n); //Ввод с клавиатуры массива точек t размера n.

void output\_xy (tochka\_xy \*t, size\_t n); //Вывод массива t размера n на экран

int kol\_zap\_file (char \*f\_name); //Возвращает кол-во записей файла f\_name.

void input\_file (tochka \*\*t, size\_t n, char\* f\_name); //Ввод массива t размера n.

void output\_file (tochka \*t, size\_t n); //Запись данных в файл

#endif // OUTINP\_H\_INCLUDED

***Заголовочный файл tpoint.h***

#ifndef TPOINT\_H\_INCLUDED

#define TPOINT\_H\_INCLUDED

#include "outinp.h"

int one\_line\_prostr (tochka \*t, size\_t n);

float \*urav (tochka \*t, int nom, size\_t n);

int check (tochka \*t, size\_t n, float \*koeff);

int orientation (float \*koeff);

void toch\_new (int or, tochka \*t, tochka\_xy \*p, size\_t n);

int x\_min (tochka\_xy \*t, size\_t n);

int x\_max (tochka\_xy \*t, size\_t n);

float\* abc\_xy(tochka\_xy tmin, tochka\_xy tmax);

int compare\_xy(tochka\_xy t1, tochka\_xy t2, float \*abc);

void sortt (tochka\_xy \*pn, tochka \*tn, size\_t n, float \*abc);

int orientation2 (float \*koef);

#endif // TPOINT\_H\_INCLUDED

# Заключение

В ходе выполнения курсовой работы были получены новые знания и умения в сфере разработки алгоритмов, написания и отладки программ для решения прикладных задач с использованием языка С, а так же создания программ, состоящих из нескольких модулей, работой с потоками ввода и вывода.

Кроме того, был более тщательно изучен материал курса «Алгебра и геометрия», в частности расположение точек в пространстве и на плоскости, а так же составление уравнений плоскостей и прямых.

# Список используемой литературы:

1. Атанасян Л.С., Бутузов В.Ф., Кадомцев С.Б., Киселева Л.С., Позняк Э.Г. Геометрия. Учебник для 10-11 классов средней школы.
2. Бугров Я.С., Никольский С.М. Высшая математика. Том первый: элементы линейной алгебры и аналитической геометрии.
3. Ильин В.А., Позняк Э.Г. Аналитическая геометрия.
4. Сайт: <https://studfiles.net/preview/3796469/>
5. Сайт: <https://function-x.ru/geometry_coordinates_cartesian.html>

# Приложения

Содержание файла tpoint.h

#ifndef TPOINT\_H\_INCLUDED

#define TPOINT\_H\_INCLUDED

#include "outinp.h"

int one\_line\_prostr (tochka \*t, size\_t n);

float \*urav (tochka \*t, int nom, size\_t n);

int check (tochka \*t, size\_t n, float \*koeff);

int orientation (float \*koeff);

void toch\_new (int or, tochka \*t, tochka\_xy \*p, size\_t n);

int x\_min (tochka\_xy \*t, size\_t n);

int x\_max (tochka\_xy \*t, size\_t n);

float\* abc\_xy(tochka\_xy tmin, tochka\_xy tmax);

int compare\_xy(tochka\_xy t1, tochka\_xy t2, float \*abc);

void sortt (tochka\_xy \*pn, tochka \*tn, size\_t n, float \*abc);

int orientation2 (float \*koef);

#endif // TPOINT\_H\_INCLUDED

Содержание файла tpoint.c

#include <stdio.h> #include <stdlib.h> #include <math.h> #include "tpoint.h"

#define EPS 0.00001 #include "outinp.h"

float \*urav (tochka \*t, int nom, size\_t n)

{ float \*koef = (float\*)calloc(4, sizeof(float));

float yz = (t[nom+1].y-t[nom].y)\*(t[nom+2].z-t[nom].z);

float zy = (t[nom+1].z-t[nom].z)\*(t[nom+2].y-t[nom].y);

float zx = (t[nom+1].z-t[nom].z)\*(t[nom+2].x-t[nom].x);

float xz = (t[nom+1].x-t[nom].x)\*(t[nom+2].z-t[nom].z);

float xy = (t[nom+1].x-t[nom].x)\*(t[nom+2].y-t[nom].y);

float yx = (t[nom+1].y-t[nom].y)\*(t[nom+2].x-t[nom].x);

koef[0] = yz - zy; //A

koef[1] = zx - xz; //B

koef[2] = xy - yx; //C

koef[3] = - (t[nom].x) \* yz - (t[nom].y) \* zx - (t[nom].z) \* xy + (t[nom].z) \* yx + (t[nom].y) \* xz + (t[nom].x) \* zy; //D

return koef;

}

int check (tochka \*t, size\_t n, float \*koef)

{

int i=0;

while (i<n)

{if((((koef[0]\*t[i].x)+(koef[1]\*t[i].y)+(koef[2]\*t[i].z)+koef[3]))<=EPS) i++;

else return 0; }

return 1;

}

int orientation (float \*koef)

{

if (koef[2]==0 && koef[1]==0) //yz

return 0;

if (koef[0]==0 && koef[2]==0) //xz

return 1;

if (koef[0]==0 && koef[1]==0) //xy

return 2;

return 0;

}

void toch\_new (int or, tochka \*t, tochka\_xy \*p, size\_t n)

{/ int i;

switch (or)

{

case 0:

{

for (i=0; i<n; i++) { p[i].xn = t[i].y; p[i].yn = t[i].z; }

break;

}

case 1:

{

for (i=0; i<n; i++) { p[i].xn = t[i].x; p[i].yn = t[i].z; }

break;

}

case 2:

{

for (i=0; i<n; i++) { p[i].xn = t[i].x; p[i].yn = t[i].y; }

break;

}

}

}

int x\_min (tochka\_xy \*t, size\_t n)

{ int i=0, nom = 0;

float xmin = t[nom].xn; float ymin = t[nom].yn;

for (i=0; i < n; i++)

{

if (t[i].xn < xmin)

{

xmin = t[i].xn; ymin = t[i].yn; nom = i;

}

if (t[i].xn == xmin)

{

if (t[i].yn < ymin)

{

xmin = t[i].xn; ymin = t[i].yn; nom = i;

}

}

}

return nom;

}

int x\_max (tochka\_xy \*t, size\_t n)

{ int i=0, nom = 0;

float xmax = t[nom].xn; float ymax = t[nom].yn;

for (i=0; i < n; i++)

{

if (t[i].xn > xmax)

{

xmax = t[i].xn; ymax = t[i].yn; nom = i;

}

if (t[i].xn == xmax)

{

if (t[i].yn > ymax)

{

xmax = t[i].xn; ymax = t[i].yn; nom = i;

}

}

}

return nom;

}

float\* abc\_xy(tochka\_xy tmin, tochka\_xy tmax)

{

float \*abc = (float\*)calloc(3, sizeof(float));

abc[0] = tmax.yn - tmin.yn;// A

abc[1] = tmax.xn - tmin.xn;// B

abc[2] = tmin.xn \* abc[0] + tmin.yn \* abc[1];

return abc;

}

int compare\_xy(tochka\_xy t1, tochka\_xy t2, float \*abc)

{

if ((t1.xn == t2.xn) \* (t1.yn == t2.yn)) //Если точки одинаковы

return 0;

//Если обе точки лежат под прямой

if ((t1.xn\*(abc[0]/abc[1]) + (abc[2]/abc[1]) <= t1.yn)

&& (t2.xn\*(abc[0]/abc[1]) + (abc[2]/abc[1]) <= t2.yn))

return !((t1.xn >= t2.xn ) || (t1.xn == t2.xn )&& (t1.yn < t2.yn));

//если первая и вторая точка лежат над прямой

if ((t1.xn\*(abc[0]/abc[1]) + (abc[2]/abc[1]) >= t1.yn)

&& (t2.xn\*(abc[0]/abc[1]) + (abc[2]/abc[1]) >= t2.yn))

return !((t1.xn <= t2.xn ) || (t1.xn == t2.xn )&& (t1.yn < t2.yn));

return t1.xn\*(abc[0]/abc[1])+(abc[2]/abc[1]) <= t1.yn;

}

void sortt (tochka\_xy \*pn, tochka \*tn, size\_t n, float \*abc)

{

int i, j, k; tochka t; tochka\_xy p;

for (i=0; i<n-1; i++)

{

t = tn[i]; p = pn[i]; k = i;

for (j=i+1; j<n; j++)

if (!compare\_xy(pn[j], p, abc))

{ k = j; t = tn[k]; p = pn[k]; }

tn[k] = tn[i];

tn[i] = t; pn[k] = pn[i];

pn[i] = p; }

}

int one\_line\_prostr (tochka \*t, size\_t n)

{

int i=2;

float a=0, b=0, c=0;

tochka A = t[0], B = t[1];

float xk = B.x - A.x;

float yk = B.y - A.y;

float zk = B.z - A.z;

while(i<n)

{

a = (t[i].x - A.x)\*yk - (t[i].y - A.y)\*xk;

b = (t[i].y - A.y)\*zk - (t[i].z - A.z)\*yk;

c = (t[i].x - A.x)\*zk - (t[i].z - A.z)\*xk;

if (abs(a)+abs(b)+abs(c) <= EPS)

i++;

else

return i;

}

return -1;

}

Содержание файла outinp.h

#ifndef OUTINP\_H\_INCLUDED

#define OUTINP\_H\_INCLUDED

typedef struct { float x, y, z; } tochka;

typedef struct{ float xn, yn; } tochka\_xy;

void output (tochka \*t, size\_t n);

void input (tochka \*t, size\_t n);

void output\_xy (tochka\_xy \*t, size\_t n);

int kol\_zap\_file (char \*f\_name);

void input\_file (tochka \*\*t, size\_t n, char\* f\_name);

void output\_file (tochka \*t, size\_t n);

#endif // OUTINP\_H\_INCLUDED

Содержание файла outinp.c

#include <stdio.h> #include <stdlib.h> #include "outinp.h"

void output (tochka \*t, size\_t n)

{ int i, x;

for (i=0; i<n; i++)

{ x=i+1;

printf("%d: %.2f; %.2f; %.2f;\n", x, t[i].x, t[i].y, t[i].z);

}

}

void input (tochka \*t, size\_t n)

{ int i, x;

for (i=0; i<n; i++)

{ x=i+1;

printf("%d: ", x);

scanf("%f", &t[i].x);

scanf("%f", &t[i].y);

scanf("%f", &t[i].z);

}

}

void output\_xy (tochka\_xy \*t, size\_t n)

{ int i, x;

for (i=0; i<n; i++)

{ x=i+1;

printf("%d: %.2f, %.2f\n", x, t[i].xn, t[i].yn);

}

}

int kol\_zap\_file (char \*f\_name)

{ char ch;

int i, kol=0;

FILE \*F = fopen(f\_name, "r");

while (!feof(F))

{ ch=fgetc(F); if (ch=='\n') kol++; }

fseek(F, 0, SEEK\_SET);

fclose(F);

return kol;

}

void input\_file (tochka \*\*t, size\_t n, char\* f\_name)

{

FILE \*F = fopen(f\_name, "r");

int i;

for (i=0; i<n; i++)

{

fscanf(F, "%f", &(\*t)[i].x);

fscanf(F, "%f", &(\*t)[i].y);

fscanf(F, "%f", &(\*t)[i].z);

}

fclose(F);

}

void output\_file (tochka \*t, size\_t n)

{

printf("\nВведите имя файла для сохранения: ");

char f\_name [10];

scanf("%s", &f\_name);

int i;

FILE \*F = fopen(f\_name, "w");

for (i=0; i<n; i++)

{

fprintf(F, "%.2f", t[i].x);

fprintf(F, " %.2f", t[i].y);

fprintf(F, " %.2f", t[i].z);

fprintf(F, "\n");

}

fclose(F);

}

Содержание файла main.c

#include <stdio.h> #include <stdlib.h> #include <locale.h>

#include "outinp.h" #include "tpoint.h"

int main()

{

setlocale(LC\_ALL, "Rus");

int v, n, i; tochka \*t;

printf("1 - Ввод из файла;\n2 - Ввод с клавиатуры;\nВаш выбор: ");

scanf("%d", &v);

switch (v)

{

case 1:

{

printf("\nВведите имя файла: "); char f\_name [10];

scanf("%s", &f\_name);

n=kol\_zap\_file(f\_name);

if (n<3)

{ printf("\nМногоугольник не может быть построен.");

return 0;

}

t = (tochka\*)calloc(n, sizeof(tochka));

input\_file(&t, n, f\_name);

printf("\nИз файла считаны точки: "); output(t, n); printf("\n");

break;

}

case 2:

{

printf("\nВведите количество точек: "); scanf("%d", &n);

if (n<3)

{

printf("\nМногоугольник не может быть построен.");

return 0;

}

t = (tochka\*)calloc(n, sizeof(tochka));

input(t, n); printf("\n");

break;

}

default:

return 0;

}

int nom = one\_line\_prostr(t, n);

if (nom==-1)

{

printf("\nВсе точки лежат на одной прямой.\n");

printf("Построение невозможно.\n");

return 0;

}

float \*ur = (float\*)calloc(4, sizeof(float));

ur = urav(t, nom, n);

if (!check(t, n, ur))

{

printf("\nТочки не лежат в одной плоскости.\n");

return 0;

}

int or = orientation(ur);

tochka\_xy \*p = (tochka\_xy\*)calloc(n, sizeof(tochka\_xy));

toch\_new(or, t, p, n);

int imin = x\_min(p, n), imax = x\_max(p, n);

float \*abc;

abc = abc\_xy(p[imin], p[imax]);

sortt(p, t, n, abc);

printf("\nРезультат:");

output(t, n);

printf("\nДля записи данных в файл нажмите 1.\n");

scanf("%d", &v);

if (v==1)

output\_file(t, n);

free(t);

free(p);

return 0;

}