**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 многоугольник с минимальным периметром из заданного множества точек»

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Макаров Д.С.

(подпись) ВТ-22

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

(подпись)

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

Белгород

2018 г.

**Оглавление**

[Введение 3](#_Toc533422469)

[Постановка задачи 3](#_Toc533422470)

[Обоснование выбора решения задачи. 3](#_Toc533422471)

[Демонстрационная программа. 5](#_Toc533422472)

[Блок-схема алгоритма в укрупненных блоках. 6](#_Toc533422473)

[Описание функций 7](#_Toc533422474)

[Тестовые данные 12](#_Toc533422475)

[Разбиение на модули 12](#_Toc533422476)

[Заголовочные файлы 12](#_Toc533422477)

[Заключение 13](#_Toc533422478)

[Список используемой литературы: 13](#_Toc533422479)

[Приложение 14](#_Toc533422480)

Введение

Поставленная в этой работе задача требует знания основных геометрических понятий и связана с аналитической геометрией и комбинаторикой.

Прежде чем приступить к решению задачи необходимо точно сформулировать понятие многоугольник.

Многоугольник - это [геометрическая](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F) фигура, обычно определяемая как часть плоскости ограниченная замкнутой не самопересекающейся [ломаной](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BC%D0%B0%D0%BD%D0%B0%D1%8F).  
Ломанная— [геометрическая фигура](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B3%D1%83%D1%80%D0%B0_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)), состоящая из [отрезков](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D1%80%D0%B5%D0%B7%D0%BE%D0%BA), последовательно соединённых своими концами.

Поиск многоугольника с минимальным периметром будет осуществляться при помощи полного перебора всех перестановок.

Перестановка – упорядоченный набор без повторений чисел 1,2,3…, n где n длина перестановки.

Постановка задачи

Разработать алгоритм и написать программу на языке Си для определения возможности соединения n точек на плоскости так, чтобы они образовывали многоугольник. Ввод может производится как с клавиатуры так из файла.

Обоснование выбора решения задачи.

Решение поставленной задачи заключается в следующем:   
 Ввод исходных данных производится из файла указанного пользователем или же из консоли. Если количество точек меньше 3, программа завершается т.к. по определению многоугольник должен состоять из 3 или более точек , иначе начинается поиск многоугольника с минимальным периметром среди всех перестановок введенных точек.  
 Путем генерации перестановок массива индексов, перебираются все возможные варианты ломанных линий, которые можно составить из исходных точек. Точки расставляются в порядке, указанном в текущей перестановки и начинается проверка на самопересечение сгенерированной ломанной. Если ломанная не самопересекается то вычисляется ее периметр и сравнивается с минимальным (в начале минимальным значением считается максимальное значение принимаемое типом double), если вычисленный периметр меньше чем минимальный, полученная фигура сохраняется.

Точки и линии представляют собой структуры

***Точка***

**typedef struct{**

**int x;**

**int y;**

**}point;**

***Линия***

**typedef struct{**

**int a;**

**int b;**

**}line;**

Проверка пересечения двух линий состоит из логического выражения:

**((point\_rotate(a.a,a.b,b.a)\*point\_rotate(a.a,a.b,b.b))<=0)&&((point\_rotate(b.a,b.b,a.a)\*point\_rotate(b.a,b.b,a.b))<0)**

Где, функция point\_rotate проверяет при помощи которой вычисляется направление поворота, по формуле z=(b.x-a.x)\*(c.y-b.y)-(b.y-a.y)\*(c.x-b.a), если z>0 то поворот правый, если z<0 то поворот левый, если же z=0 то все 3 точки лежат на одной прямой.

Для проверки всех возможных ломанных, которые можно составить из введенных точек, используется отложенный генератор перестановок, где каждая новая перестановка генерируется по вызову функции. Так как решение задачи использует алгоритм полного перебора, программа выполняется за факториальное время (O(n!)).

График 1. График зависимости кол-ва операций проверки на пересечение от кол-ва точек

Это необходимо учитывать при подборе тестовых данных, потому что уже при n <11 поиск фигуры с минимальным периметром будет относительно долго.

Демонстрационная программа.

Представляет собой программу считывающую название файла с исходными данными указанными в формате x:y, где x и y соответствующие координаты точки в декартовой системе координат. При успешном нахождении фигуры выводит координаты точек в порядке соединения для построения фигуры, а так же сохраняет результат в файл по запросу пользователя.

Блок-схема алгоритма в укрупненных блоках.



Описание функций

**Проверка массива линий на самопересечение**

Спецификация функции poly\_check\_intersect

Заголовок char poly\_check\_intersect(Line\* arr,unsigned size)

Назначение: Возвращает истина если ломанная записанная в массиве arr самопересекается, ложь если нет.



**Генерация перестановок**

Спецификация функции gen\_transposition

Заголовок int gen\_transposition(unsigned\* index\_arr,unsigned size)

Назначение: Возвращает истина если следующая перестановка записана в массив index\_arr, ложь если перестановки закончились.



**Генерация массива содержащего фигуру**

Спецификация функции gen\_poly

Заголовок void gen\_poly(int\* index\_arr,point\* point\_arr,Line\* poly,unsigned size)

Назначение: Записывает в массив линий poly, точки из массива point\_arr в порядке указанном в массиве index\_arr



**Вычислениe периметра фигуры.**

Спецификация функции perimetr\_poly

Заголовок double perimetr\_poly(Line\* arr,unsigned size)

Назначение: Возвращает периметр многоугольника записанного в массиве arr.



**Поиск и построение многоугольника с минимальным периметром.**

Спецификация функции min\_perimetr\_poly

Заголовок Line\* min\_perimetr\_poly(point\* arr,unsigned size)

Назначение: Возвращает указатель на массив содержащий многоугольник с минимальным периметром, иначе если построить многоугольник из заданных точек невозможно возвращает 0.



**Генерация массива индексов.**

Спецификация функции gen\_index\_arr

Заголовок unsigned\* gen\_index\_arr(unsigned size)

Назначение: Возвращает указатель на массив arr размера size содержащий числа от 0 до size-1.



**Генерация многоугольника с минимальным периметром.**

Спецификация функции gen\_minimal\_poly

Заголовок void gen\_minimal\_poly(point\* arr, unsigned size)

Назначение: Выводит в стандартный поток вывода массив точек в порядке построения многоугольника с минимальным периметром. По усмотрению пользователя может произвести запись результата в файл.



Тестовые данные

В качестве тестовых данных используются случайные наборы точек с координатами от 0 до 100.

|  |  |  |
| --- | --- | --- |
| Исходные данные | Полученный результат | Визуализация |
| 61:32 83:44 12:75 90:46 81:74 82:1 68:60 20:11 | 68:60 12:75 20:11 82:1 61:32 83:44 90:46 81:74  Периметр = 303.572652 | https://pp.userapi.com/c847216/v847216048/1544cf/u3JMUOC6rps.jpg |
| 21:56 27:100 84:36 14:97 13:19 30:72 61:1 33:7 | 21:56 13:19 33:7 61:1 84:36 27:100 14:97 30:72  Периметр = 278.779063 | https://pp.userapi.com/c849536/v849536048/e6de3/Nl7PEJAJlK4.jpg |
| 70:70 92:91 87:78 45:51 38:33 53:30 58:48 51:91 38:3 43:87 | 87:78 70:70 58:48 53:30 38:3 38:33 45:51 43:87 51:91 92:91  Периметр = 242.658035 | https://pp.userapi.com/c845420/v845420048/15aa28/jl6rslElCII.jpg |

Разбиение на модули

Демонстрационная программа использует 2 модуля, модуль функций для работы с точками и линиями (point.h), модуль для решения задачи (polygon\_gen.h) и модуль для работы с файлами (file.h).

Заголовочные файлы

**Заголовочный файл point.h**

**#ifndef POINT\_H\_**

**#define POINT\_H\_**

**#include <stdio.h>**

**#include <stdlib.h>**

**typedef struct {int x; int y;}point;**

**typedef struct {point a; point b;}Line;**

**point\* point\_arr\_init(unsigned size);**

**Line\* Line\_arr\_init(unsigned size);**

**int point\_rotate(point a,point b, point c);**

**double Line\_lenght(Line a);**

**int Line\_intersect(Line a, Line b);**

**point\* point\_arr\_init(unsigned size);**

**void point\_swap(point\* a,point\* b);**

**void point\_to\_Line\_arr(point\* orig,Line\* dest,unsigned size);**

**void Line\_to\_point\_arr(Line\* orig,point\* dest,unsigned size);**

**void point\_read\_arr(point \*arr,int size);**

**void point\_write\_arr(point \*arr,int size);**

**void print\_Line(Line a);**

**void print\_point(point a);**

**void Line\_write\_arr(Line\* arr,int size);**

**#endif**

**Заголовочный файл file.h**

#ifndef FILE\_H\_

#define FILE\_H\_

#include <stdio.h>

#include <stdlib.h>

#include "point.h"

unsigned read\_point\_file(FILE\* file\_in, point\* arr\_out);

void write\_line\_file(point\* arr\_in,unsigned size);

#endif

**Заголовочный файл file.h**

#ifndef POINT\_GEN\_H\_

#define POINT\_GEN\_H\_

#include <stdio.h>

#include <stdlib.h>

#include "point.h"

#include <float.h>

#include "file.h"

extern int count;

void copy\_poly(point\* dest, point\* origin,unsigned size);

int gen\_transposition(unsigned\* index\_arr,unsigned size);

char poly\_check\_intersect(Line\* arr,unsigned size);

void gen\_poly(int\* index\_arr,point\* point\_arr,Line\* poly,unsigned size);

void swap(int \*a, int i, int j);

void Line\_arr\_copy(Line\* orig,Line\* dest,unsigned size);

Line\* min\_perimetr\_poly(point\* arr,unsigned size);

unsigned\* gen\_index\_arr(unsigned size);

double perimetr\_poly(Line\* arr,unsigned size);

void gen\_minimal\_poly(point\* arr,unsigned size);

#endif

Заключение

В ходе выполнения курсовой работы были повторен материал курса “Алгебра и геометрия”, а также получены новые знания в сфере аналитической геометрии и комбинаторики.

Получены навыки реализации отложенного генератора перестановок, а также новые умения в сфере разработки алгоритмов по поставленным прикладным задачам и создание программ по созданным алгоритмам на языке программирования C, состоящих из нескольких модулей.

Список используемой литературы:

1. Стратегия вычисления [Электронный ресурс] <URL:https://ru.wikipedia.org/wiki/Стратегия_вычисления>
2. Локализация точки в выпуклом многоугольнике [Электронный ресурс] URL:<https://habr.com/post/144571/>
3. Ломанная [Электронный ресурс]

<URL:https://ru.wikipedia.org/wiki/Ломанная>

Приложение

**Исходный код file.с**

#include "file.h"

unsigned read\_point\_file(FILE\* file\_in, point\* arr\_out){

int size=10;

int i=0;

arr\_out=point\_arr\_init(size);

char current\_char;

while((fscanf(file\_in,"%d:%d",&arr\_out[i].x,&arr\_out[size].y))!=EOF){

i++;

if(size<=i){

size\*=2;

arr\_out=(point\*)realloc(arr\_out,size);

}

}

arr\_out=(point\*)realloc(arr\_out,i);

return size;

}

void write\_point\_file(point\* arr\_in,unsigned size){

FILE\* file\_out=fopen("./result.txt","w");

for (int i=0;i<size;i++){

fprintf(file\_out,"%d:%d ",arr\_in[i].x,arr\_in[i].y);

}

fclose(file\_out);

}

**Исходный код point.с**

#include <stdio.h>

#include <stdlib.h>

#include "point.h"

#include <math.h>

point\* point\_arr\_init(unsigned size){

point\* a =(point\*)malloc(sizeof(point)\*size);

return a;

}

Line\* Line\_arr\_init(unsigned size){

Line\* a =(Line\*)malloc(sizeof(Line)\*size);

return a;

}

int point\_rotate(point a,point b, point c){

return ((b.x-a.x)\*(c.y-b.y)-(b.y-a.y)\*(c.x-b.x));

}

double Line\_lenght(Line a){

return sqrt((a.b.x-a.a.x)\*(a.b.x-a.a.x)+(a.b.y-a.a.y)\*(a.b.y-a.a.y));

}

int Line\_intersect(Line a,Line b){

return ((point\_rotate(a.a,a.b,b.a)\*point\_rotate(a.a,a.b,b.b))<=0)&&((point\_rotate(b.a,b.b,a.a)\*point\_rotate(b.a,b.b,a.b))<0);

}

void point\_to\_Line\_arr(point\* orig,Line\* dest,unsigned size){

Line temp\_Line;

int i;

for(i=0;i<size-1;i++){

temp\_Line.a=orig[i];

temp\_Line.b=orig[i+1];

dest[i]=temp\_Line;

}

temp\_Line.a=orig[0];

temp\_Line.b=orig[i];

dest[i]=temp\_Line;

}

void Line\_to\_point\_arr(Line\* orig,point\* dest,unsigned size){

dest[0]=orig[0].a;

for(int i=0;i<size-1;i++){

dest[i+1]=orig[i].b;

}

}

void point\_read\_arr(point \*arr,int size){

printf("Input point coordinates - ");

for(int i=0;i<size;i++){

scanf("%d:%d",&arr[i].x,&arr[i].y);

}

}

void point\_write\_arr(point \*arr,int size){

for (int i=0;i<size;i++){

print\_point(arr[i]);

}

printf("\n");

}

void print\_point(point a){

printf("%d:%d ",a.x,a.y);

}

void print\_Line(Line a){

printf("\n");

print\_point(a.a);

print\_point(a.b);

printf("\n");

}

void Line\_write\_arr(Line\* arr,int size){

for (int i=0;i<size;i++){

print\_Line(arr[i]);

}

}

**Исходный код polygon\_gen.с**

#include <stdio.h>

#include <stdlib.h>

#include "point.h"

#include "polygon\_gen.h"

#include "file.h"

char poly\_check\_intersect(Line\* arr,unsigned size){

int i,j,f;

for(i=0;i<size;i++){

for(j=0;j<size;j++){

if((j!=i-1)&&(j!=i)&&(j!=i+1)){

if(Line\_intersect(arr[i],arr[j])){

count+=i\*j;

return 1;

}

}

}

}

count+=size\*size;

return 0;

}

int gen\_transposition(unsigned\* index\_arr,unsigned size){

int j = size - 2;

while ((j!= -1) && (index\_arr[j] >= index\_arr[j+1]))

j--;

if (j == -1) return 0;

int k = size - 1;

while (index\_arr[j] >= index\_arr[k]) k--;

swap(index\_arr,j,k);

int l = j + 1, r = size - 1;

while (l<r) swap(index\_arr,l++,r--);

return 1;

}

void swap(int \*a, int i, int j){

int s = a[i];

a[i] = a[j];

a[j] = s;

}

void gen\_poly(int\* index\_arr,point\* point\_arr,Line\* poly,unsigned size){

point\* temp=point\_arr\_init(size);

for(int i=0;i<size;i++){

temp[i]=point\_arr[index\_arr[i]-1];

}

point\_to\_Line\_arr(temp,poly,size);

free(temp);

}

double perimetr\_poly(Line\* arr,unsigned size){

double sum=0;

for(int i=0;i<size;i++){

sum+=Line\_lenght(arr[i]);

}

return sum;

}

unsigned\* gen\_index\_arr(unsigned size){

unsigned\* arr=(unsigned\*)malloc(sizeof(unsigned)\*size);

for(int i=0;i<size;i++){

arr[i]=i+1;

}

return arr;

}

Line\* min\_perimetr\_poly(point\* arr,unsigned size){

unsigned\* index\_arr=gen\_index\_arr(size);

Line\* poly\_result=Line\_arr\_init(size);

Line\* poly\_current=Line\_arr\_init(size);

double min=DBL\_MAX;

double temp;

char flag=0;

gen\_poly(index\_arr,arr,poly\_current,size);

if(!(poly\_check\_intersect(poly\_current,size))){

temp=perimetr\_poly(poly\_current,size);

if(temp<min){

flag=1;

min=temp;

Line\_arr\_copy(poly\_current,poly\_result,size);

}

}

while(gen\_transposition(index\_arr,size)){

gen\_poly(index\_arr,arr,poly\_current,size);

if(!(poly\_check\_intersect(poly\_current,size))){

temp=perimetr\_poly(poly\_current,size);

if(temp<min){

flag=1;

min=temp;

Line\_arr\_copy(poly\_current,poly\_result,size);

}

}

}

free(index\_arr);

free(poly\_current);

if (flag){

return poly\_result;

}

else return 0;

}

void Line\_arr\_copy(Line\* orig,Line\* dest,unsigned size){

int i;

for(i=0;i<size;i++){

dest[i]=orig[i];

}

}

void gen\_minimal\_poly(point\* arr, unsigned size){

Line\* l\_arr = Line\_arr\_init(size);

if((l\_arr=min\_perimetr\_poly(arr,size))!=NULL){

Line\_to\_point\_arr(l\_arr,arr,size);

printf("Result polygon with min perimetr\n");

point\_write\_arr(arr,size);

printf("Perimetr = %f\n",perimetr\_poly(l\_arr,size));

}

else(printf("Can't build polygon"));

fflush(stdin);

printf("Save result in file? (y/N) ");

char m=0;

scanf("%c",&m);

if((m=='y')||(m=='Y')){

printf("flag %c Write result to file\n",m);

write\_point\_file(l\_arr,size);

}

}

**Исходный код main.с**

#include <stdio.h>

#include <stdlib.h>

#include "point.h"

#include "polygon\_gen.h"

#include <time.h>

#include <string.h>

int count=0;

void console\_mode(){

printf("Write count of points - ");

int size;

scanf("%d",&size);

point\* arr=point\_arr\_init(size);

point\_read\_arr(arr,size);

clock\_t start = clock();

gen\_minimal\_poly(arr,size);

clock\_t end = clock();

printf("Count of intersect check - %d\n",count);

printf("Runtime - %f sec\n",(double)(end-start)/CLOCKS\_PER\_SEC );

free(arr);

}

void file\_mode(){

FILE \*file\_in;

file\_in=fopen(name,"r");

if(file\_in!=NULL){

point\* arr;

unsigned size=read\_point\_file(file\_in,arr);

fclose(file\_in);

gen\_minimal\_poly(arr,size);}

else{

printf("Wrong file name or file don't exist\n");

}}}

int main(){

char\* str\_name[50];

scanf("%s\n",str\_name);

if(strcomp(str\_name,"console")){

file\_mode();}

else{

console\_mode();

}

}