Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика".
2015 год. Отборочный этап. 8-9 классы. Билет 1. Условия и решения.

Задача 1: Наиболее удаленные точки (10)

Ортогональную целочисленную решетку, состоящую из точек с целыми координатами в декартовой системе координат, будем обозначать через Z^2. На решетке Z^2 задано N точек. Найти расстояние между двумя наиболее удаленными точками.
Входные данные. Первая строка входного файла содержит целое число N ($1<N<10^4$) – количество точек. В последующих N строках записаны пары целых чисел x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$), задающих координаты точек.
Выходные данные. В выходной файл вывести одно вещественное число – расстояние между двумя наиболее удаленными точками с точностью до четвертого знака после запятой.

Пример входного файла	Пример выходного файла
4 0 1 1 0 0 -1 -1 0	2.0000

Решение задачи 1

```
#include "stdafx.h"
#include <math.h>

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	int n;
	long int x[10000], y[10000];
	double maximum_distance = 0;
	double ax, ay, distance;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	fscanf( ifs, "%d", &n);
	for(int i = 0; i < n; i++)
		fscanf( ifs, "%ld %ld", &x[i], &y[i] );
	for(int i = 0; i < n-1; i++)
		for(int j = i+1; j < n; j++)
		{
			ax = fabs( (double)(x[i]-x[j]) );
			ay = fabs( (double)(y[i]-y[j]) );
			distance=sqrt( ax*ax + ay*ay );
			if ( distance > maximum_distance ) maximum_distance = distance;
		}
	fprintf( ofs, "%.4f\n", maximum_distance );
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 2: Заглавная буква (25)

Дан текст, содержащий список слов на английском языке. Слова отделяются друг от друга, по крайней мере, одним пробелом. При записи слов допускаются как строчные, так и заглавные буквы. Необходимо каждое слово начать с заглавной буквы, а все остальные буквы в слове сделать строчными.
Входные данные. Входной файл содержит одну или несколько строк текста. Длина строки текста не более 100. Размер файла не превышает 1000 строк.
Выходные данные. В выходной файл вывести строки преобразованного исходного текста.

Пример входного файла	Пример выходного файла
CaPiTaLiZe the first letter number the lines Army ANTS darth Vader	Capitalize The First Letter Number The Lines Army Ants Darth Vader

Решение задачи 2

```
#include "stdafx.h"
#include <string.h>
#include <ctype.h>
```

```
FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	char istr[100];
	bool flag = true;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	while ( !feof( ifs ) )
	{
		fgets( istr, 100, ifs );
		if ( feof( ifs ) ) break;
		if ( strlen( istr ) == 0) continue;
		flag = true;
		strlwr( istr );
		for ( size_t i = 0; i < strlen( istr ); i++ )
		{
			if ( isalpha( istr[i] ) )
			{
				if ( flag )
				{
					istr[i] = toupper( istr[i] );
					flag = false;
				}
			}
			else
				flag = true;
		}
		fputs( istr, ofs );
	}
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 3: Последовательность (20)

Последовательность чисел a_1, a_2, …, a_n, … задается следующими условиями: а) a_1 – произвольное целое положительное число; б) если a_n чётно, то $a_{n+1} = a_n/2$, а если нечётно, то $a_{n+1} = 3a_n + 1$. Например, для $a_1 = 5$ получится следующая последовательность: 5, 16, 8, 4, 2, 1, 4, 2, 1, ... Интересно (но не доказано), что такая последовательность будет всегда заканчиваться повторяющимся циклом: 4, 2, 1, 4, 2, 1, 4, 2, 1, ... Полагают, что при $a_1 = 1$ последовательность закончилась. Для заданного числа a_1 определить наибольшее значение в последовательности.
Входные данные. Во входном файле записано одно целое число a_1 ($1 \le a_1 \le 10^5$).
Выходные данные. В выходной файл вывести одно целое число – наибольшее значение в полученной последовательности.

Примеры входного файла	Примеры выходного файла
3	16
100000	100000

Решение задачи 3

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned long n=0;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	fscanf( ifs, "%lu", &n);
	unsigned long max = n;
	while ( n > 1 )
	{
		if (n%2 == 0) n /= 2;
		else
		{
			n *= 3;
			n++;
			if (n > max) max = n;
```

```
            }
        }
        fprintf( ofs, "%lu\n", max );
        fclose( ifs );
        fclose( ofs );
        return 0;
}
```

Задача 4: PIN-код (20)

Рассматривается PIN-код, состоящий из 6 (шести) десятичных цифр. PIN-код считается «слабым», если

a) содержит последовательность из 3 (трех) цифр, идущих подряд в порядке возрастания/убывания от 0 до 9 и от 9 до 0 (например, 145698 и 986541);

b) содержит последовательность из 3 (трех) одинаковых цифр, идущих подряд (например, 130001).

Во всех остальных случаях PIN-код считается «допустимым». Определить уровень безопасности коллекции PIN-кодов.

Входные данные. Первая строка входного файла содержит целое число N ($1 < N < 1000$) – количество PIN-кодов. В последующих N строках записаны PIN-коды.

Выходные данные. В выходной файл для каждого PIN-кода во входном файле вывести слово "WEAK" или "ACCEPTABLE". Слово "WEAK" означает "слабый", а слово "ACCEPTABLE" означает "допустимым".

Пример входного файла	Пример выходного файла
5 145698 986541 130001 968541 540872	WEAK WEAK WEAK ACCEPTABLE ACCEPTABLE

Решение задачи 4

```
#include "stdafx.h"
#include <string.h>

FILE *ifs, *ofs;

int f(char *string)
{
        char *str1[] = { "012", "123", "234", "345", "456", "567", "678", "789" };
        char *str2[] = { "987", "876", "765", "654", "543", "432", "321", "210" };
        char *str3[] = { "000", "111", "222", "333", "444", "555", "666", "777", "888", "999" };
        int i;
        char *pdest;
        int result = 0;
        for (i = 0; i < 8; i++)
        {
                pdest = strstr( string, str1[i] );
                if (pdest != NULL)
                        result = 1;
        }
        for (i = 0; i < 8; i++)
        {
                pdest = strstr( string, str2[i] );
                if (pdest != NULL)
                        result = 1;
        }
        for (i = 0; i < 10; i++)
        {
                pdest = strstr( string, str3[i] );
                if (pdest != NULL)
                        result = 1;
        }
        return result;
}

int _tmain(int argc, _TCHAR* argv[])
{
        int n;
        char string[7];
        if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
        if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
        fscanf( ifs, "%d", &n );
        for (int i = 0; i < n; i++)
```

```
    {
        fscanf( ifs, "%s", string );
        if ( !f( string ) )
            fprintf( ofs, "ACCEPTABLE\n" );
        else
            fprintf( ofs, "WEEK\n" );
    }
    fclose( ifs );
    fclose( ofs );
    return 0;
}
```

Задача 5: Треугольники (25)

На окружности произвольного радиуса задано N точек, пронумерованных числами натурального ряда: 1, 2, …, N. Для каждой пары смежных точек определена длина дуги окружности в виде целого числа. Определить, сколько различных равносторонних треугольников можно построить, используя заданные точки в качестве вершин.

Входные данные. Первая строка входного файла содержит целое число N ($3 \le N \le 10^5$) – количество точек на окружности. Вторая строка входного файла содержит N целых чисел X_i ($1 \le X_i \le 10^3$), представляющих длины дуг между двумя последовательными точками на окружности. Число X_i для $1 \le i \le (N - 1)$ представляет собой длину дуги между точками i и $i + 1$; число X_N представляет собой длину дуги между точками N и 1.

Выходные данные. В выходной файл вывести одно целое число – количество различных равносторонних треугольников.

Примеры входного файла	Примеры выходного файла
8 4 2 4 2 2 6 2 2	2
6 3 4 2 1 5 3	1

Решение задачи 5

```
def convert_str_to_int_arr(string, space):
    '''
    преобразование стороки в массив целых чисел
    string - строка,
    space - разделитель
    '''
    res_list = [int(el) for el in string.split(space)]
    return res_list


def read_from_file(filename):
    '''
    чтение из файла
    filename - имя файла
    '''
    try:
        with open(filename) as fin:
            # считываем количество точек
            num = fin.readline()
            # считываем расстояния и возвращаем их
            return convert_str_to_int_arr(fin.readline(), ' ')
    except Exception as error:
        pass


def write_to_file(filename, number):
    '''
    запись в файл
    filename - имя файла,
    number - число для записи
    '''
    try:
        with open(filename, "w") as fout:
            fout.write(str(number))
    except Exception as error:
        pass


def find_num_of_trs(dist_list):
    '''
    Нахождение количества разных равносторонних треугольников
```

```
    dist_list - список, содержащий расстояния точек на окружности
    '''

    # длина окружности
    sum_of_dist = sum(dist_list)
    # длина дуги окружности, соответствующей каждой из сторон равностороннего треугольника
    middle = sum_of_dist // 3
    # число заданных точек на окружности
    num_of_points = len(dist_list)
    # список уже найденных равносторонних треугольников
    used_tr = []
    # число разных равносторонних треугольников
    count = 0
    # возможен ли равносторонний треугольник с натуральной длиной дуг, соответствующих сторонам
треугольника
    if sum_of_dist % 3 == 0:
        # проходим по всем точкам на окружности
        for i, el in enumerate(dist_list):
            # вершина a равностороннего треугольника
            a = i
            # внутренний индекс j, dist - расстояние между точками на окружности
            j, dist = a, 0
            # моделируем кольцевой список на основе списка, содержащего точки на окружности
путем изменения индекса j по
            # формуле j = (j+1) % num_of_points
            # в цикле постепенно накапливаем расстояния между точками и определяем точку на
окр., которая находится
            # от предыдущей на расстояние, которое не меньше dist
            while(dist < middle):
                dist += dist_list[j]
                j = (j + 1) % num_of_points
            # если dist > middle, то отбраковываем найденную точку и переходим на следующую
итерацию цикла for
            if dist > middle:
                continue
            # вершина b равностороннего тр-ка
            b = j
            dist = 0
            while(dist < middle):
                dist += dist_list[j]
                j = (j + 1) % num_of_points
            if dist > middle:
                continue
            # вершина c равностороннего тр-ка
            c = j
            # формируем тр-к из вершин a, b, c
            # треугольник - трехэлементное множество, состоящее из точек на окр-ти с индексами
a, b, c
            tr = {a, b, c}
            # если этот треугольник еще не был найден нами до этого, то увеличиваем число
равносторонних треугольников
            # на один и добавляем его в список обработанных равносторонних треугольников. Это
делается для подсчета
            # только различных равносторонних треугольников
            if tr not in used_tr:
                count += 1
                used_tr.append(tr)
    return count


def main():
    # имя входного файла
    input = "input1.txt"
    # имя выходного файла
    output = "output.txt"
    # получение данных из входного файла
    dist_list = read_from_file(input)
    # нахождение количества различных равносторонних треугольников
    count = find_num_of_trs(dist_list)
    # запись результата в выходной файл
    write_to_file(output, count)

if __name__ == "__main__":
    main()
```

Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика".
2015 год. Отборочный этап. 8-9 классы. Билет 2. Условия и решения.

Задача 1: Наиболее близкие точки (10)

Ортогональную целочисленную решетку, состоящую из точек с целыми координатами в декартовой системе координат, будем обозначать через Z^2. На решетке Z^2 задано N точек. Найти расстояние между двумя наиболее близкими точками.
Входные данные. Первая строка входного файла содержит целое число N ($1 < N < 10^4$) – количество точек. В последующих N строках записаны пары целых чисел x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$), задающих координаты точек.
Выходные данные. В выходной файл вывести одно вещественное число – расстояние между двумя наиболее близкими точками с точностью до четвертого знака после запятой.

Пример входного файла	Пример выходного файла
4 0 1 1 0 0 -1 -1 0	1.4142

Решение задачи 1

```
#include "stdafx.h"
#include <float.h>
#include <math.h>

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	int n;
	long int x[10000], y[10000];
	double minimum_distance = DBL_MAX; //1.7e308;
	double ax, ay, distance;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	fscanf( ifs, "%d", &n);
	for(int i = 0; i < n; i++)
		fscanf( ifs, "%ld %ld", &x[i], &y[i] );
	for(int i = 0; i < n-1; i++)
		for(int j = i+1; j < n; j++)
		{
			ax = fabs( (double)(x[i]-x[j]) );
			ay = fabs( (double)(y[i]-y[j]) );
			distance=sqrt( ax*ax + ay*ay );
			if ( distance < minimum_distance ) minimum_distance = distance;
		}
	fprintf( ofs, "%.4f\n", minimum_distance );
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 2: Лишние пробелы (25)

Дан текст, содержащий список слов на английском языке. Слова отделяются друг от друга, по крайней мере, одним пробелом. В строках допускаются лидирующие и хвостовые пробелы. Необходимо в каждой строке удалить все лидирующие и хвостовые пробелы и оставить между словами по одному пробелу.
Входные данные. Входной файл содержит одну или несколько строк текста. Длина строки текста не более 100. Размер файла не превышает 1000 строк.
Выходные данные. В выходной файл вывести строки преобразованного исходного текста.

Пример входного файла	Пример выходного файла
The input will start with a line containing a single integer value specifying the number of lines of text that will follow	The input will start with a line containing a single integer value specifying the number of lines of text that will follow

Решение задачи 2

```
#include "stdafx.h"
```

```
#include <string.h>

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	char istr[100], ostr[100];
	char seps[] = " ";
	char *token;
	bool flag;
	int n;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	while ( !feof( ifs ) )
	{
		fgets( istr, 100, ifs );
		if ( feof( ifs ) ) break;
		if ( strlen( istr ) == 0) continue;
		flag = true;
		token = strtok( istr, seps );
		while ( token != NULL )
		{
			if ( flag )
			{
				strcpy( ostr, token );
				flag = false;
			}
			else
			{
				strcat( ostr, seps );
				strcat( ostr, token );
			}
			token = strtok( NULL, seps );
		}
		fputs( ostr, ofs );
	}
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 3: Закраска (20)

Дан лист клетчатой бумаги размером N x N клеток. Сначала выбираются некоторые строки и закрашиваются каким-либо цветом. Затем выбираются некоторые столбцы и тоже закрашиваются каким-либо цветом. Никакая строка и никакой столбец не закрашиваются дважды. Подсчитать количество незакрашенных клеток.

Входные данные. Первая строка входного файла содержит три целых числа: N ($1 \le N \le 100$), R ($0 \le R \le N$) – количество закрашиваемых строк, C ($0 \le C \le N$) – количество закрашиваемых столбцов. Вторая строка входного файла содержит R различных целых чисел ($1 \le r_i \le N$) – номера закрашиваемых строк. Третья строка входного файла содержит C различных целых чисел ($1 \le c_j \le N$) – номера закрашиваемых столбцов.

Выходные данные. В выходной файл вывести одно целое число – количество незакрашенных клеток.

Пример входного файла	Пример выходного файла
4 1 2 2 1 3	6

Решение задачи 3

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	int n, r, c, ri, ci;
	int wall[100][100];
	int rest=0;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	fscanf( ifs, "%d %d %d", &n, &r, &c );
	for (int i = 0; i < n; i++)
```

```
		for (int j = 0; j < n; j++)
			wall[i][j]=1;
	for (int i = 0; i < r; i++)
	{
		fscanf( ifs, "%d", &ri );
		for (int j = 0; j < n; j++)
			wall[ri-1][j]=0;
	}
	for (int i = 0; i < c; i++)
	{
		fscanf( ifs, "%d", &ci );
		for (int j = 0; j < n; j++)
			wall[j][ci-1]=0;
	}
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			rest += wall[i][j];
	fprintf( ofs, "%d\n", rest );
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 4: ID студента (20)

В новом университете с преподаванием на английском языке ID студентов решили задавать в следующем формате: SSSSFFNNN, где SSSS – первые четыре согласные буквы фамилии студента, FF – первые две гласные буквы имени студента, и NNN – трехзначный порядковый номер студента, чтобы отличать ID студентов, имеющих одинаковую часть SSSSFF. Если не хватает согласных букв в фамилии или гласных букв в имени, то заполнять недостающие буквы символом Z. При назначении порядкового номера ID студента, начинать надо всегда с 000. Гласными буквами следует считать следующие 5 (пять) букв: A, E, I, O, U. Все остальные буквы следует считать согласными. Для заданного списка имен и фамилий студентов определить соответствующие ID студентов.
Входные данные. Входной файл содержит одну или несколько строк, в каждой из которых записано два слова – имя и фамилия студента заглавными буквами английского алфавита. Длина каждого слова не более 20. Размер файла не превышает 1000 строк.
Выходные данные. В выходной файл для каждой строки во входном файле вывести ID студента.

Пример входного файла	Пример выходного файла
JOHN SMITH	SMTHOZ000
MICHAEL LEE	LZZZIA000
BJORN SMITHERS	SMTHOZ001
JANE SMITH	SMTHAE000
JOHN SMITH	SMTHOZ002

Решение задачи 4

```
#include "stdafx.h"
#include <stdlib.h>
#include <string.h>

typedef char ID[10];
ID ids[1000];
FILE *ifs, *ofs;

int isVowel(char c)
{
	return (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');
}

int numberOf( ID id, int n )
{
	char *pdest;
	int k = 0;
	for (int i = 0; i < n; i++)
	{
		pdest = strstr( ids[i], id );
			if (pdest != NULL)
				k++;
	}
	return k;
}
```

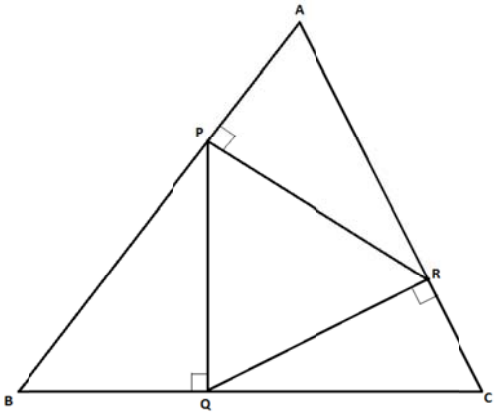
```
int _tmain(int argc, _TCHAR* argv[])
{
	char firstName[21];
	char lastName[21];
	char ssss[5];
	char ff[3];
	char nnn[4];
	char buff[4];
	ID id;
	int k;
	int n = 0;
	// Открытие файлов
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	// Чтение исходных данных и формирование массива ID студентов
	while ( !feof( ifs ) )
	{
		fscanf( ifs, "%s %s", firstName, lastName);
		if ( feof( ifs ) )
			break;
		// Шифрование фамилии
		k = 0;
		for(size_t i = 0; i < strlen(lastName); i++)
		{
			if (!isVowel( lastName[i] ))
				ssss[k++] = lastName[i];
			if (k == 4)
				break;
		}
		if (k < 4)
			for(int i = k; i < 4; i++)
				ssss[k++] = 'Z';
		ssss[k] = '\0';
		strcpy( id, ssss );
		// Шифрование имени
		k = 0;
		for (size_t i = 0; i < strlen(firstName); i++)
		{
			if (isVowel( firstName[i] ))
				ff[k++] = firstName[i];
			if (k == 2)
				break;
		}
		if (k < 2)
			for(int i = k; i < 2; i++)
				ff[k++] = 'Z';
		ff[k] = '\0';
		strcat( id, ff );
		// Шифрование номера
		int m = numberOf( id, n );
		itoa( m, buff, 10 );
		if (m < 10)
			strcpy( nnn, "00" );
		else if (m < 100)
			strcpy( nnn, "0" );
		else
			strcpy( nnn, "" );
		strcat( nnn, buff );
		strcat( id, nnn );
		strcpy( ids[n], id );
		n++;
	}
	// Печать результатов, т. е. массива ID студентов
	for (int i = 0; i < n; i++)
		fprintf( ofs, "%s\n", ids[i] );
	// Закрыти файлов
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```

Задача 5: Треугольник в треугольнике (25)

Дан остроугольный треугольник *ABC***.** Пусть *P* – такая точка на стороне *AB***,** что а) точка *Q* является проекцией точки *P* на сторону *BC*, б) точка *R* является проекцией точки *Q* на сторону *AC* и в) точка *P* является проекцией точки *R* на сторону *AB*. Вычислить длину отрезка *PB***.**

Входные данные. Входной файл содержит одну строку, в которой записаны три целых положительных числа – длины сторон *BC*, *CA* и *AB* соответственно (*BC*, *CA*, *AB* ≤ 10^4**)**.

Выходные данные. В выходной файл вывести одно вещественное число – длину отрезка *PB* с точностью до 5-го знака после запятой.



Примеры входного файла	Примеры выходного файла
10 10 10	6.66667
11 10 10	7.53894
5 6 7	3.18182
6 7 5	3.27273
7 5 6	5.34545

Решение задачи 5

```
#include "stdafx.h"

FILE *ifs, *ofs;

void readSidesLengthsFromFile( unsigned long int *bc, unsigned long int *ca, unsigned long int
*ab ) {
	fscanf( ifs, "%lu %lu %lu", bc, ca, ab );
}

float calcPbSideLength( unsigned long int a, unsigned long int b, unsigned long int c )
{
	float fa = (float)a;
	float fb = (float)b;
	float fc = (float)c;
	return (2 * (fa * fa) * fc) / (fa * fa + fb * fb + fc * fc);
}

int main(int argc, const char * argv[])
{
	unsigned long int bc, ca, ab;
	if ( (ifs = fopen( "in1.txt", "r" )) == NULL ) return 1;
	if ( (ofs = fopen( "out1.txt", "w" )) == NULL ) return 1;
	readSidesLengthsFromFile( &bc, &ca, &ab );
	fprintf( ofs, "%.5f\n", calcPbSideLength( bc, ca, ab ) );
	fclose( ifs );
	fclose( ofs );
	return 0;
}
```