Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика". 2014 год. Отборочный этап. 8-9 классы. Билет 2.

Задача 1: Ряд Фибоначчи (10)

Последовательность чисел 1, 1, 2, 3, 5, 8, 13, 21, …, a_n, …, два первых члена которой равны 1, а каждый член, начиная с третьего, равен сумме двух предыдущих: $a_{n+2} = a_{n+1} + a_n$, называется «рядом» Фибоначчи. Вычислить $a_{n+1}a_{n+2} - a_n a_{n+3}$.

Входные данные. Во входном файле записано одно целое число n ($3 \le n \le 10^{18}$).
Выходные данные. В выходной файл вывести одно целое число – значение вычисленного выражения.

Пример входного файла	Пример выходного файла
8	1

РЕШЕНИЕ:

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned long long n;
	ifs = fopen( "1.IN", "r" );
	fscanf( ifs, "%lld", &n );
	fclose( ifs );
	ofs = fopen( "1.OUT", "w" );
	if (n % 2 == 0)   // n - четное число
		fprintf( ofs, "%d\n", 1 );
	else        // n - нечетное число
		fprintf( ofs, "%d\n", -1 );
	fclose( ofs );
	return 0;
}
```

Задача 2: Сумма чисел (15)

Разобьем ряд натуральных чисел в группы: 1, (2, 3, 4), (5, 6, 7, 8, 9), (10, 11, 12, 13, 14, 15, 16), … . Найти сумму чисел n-й группы.

Входные данные. Во входном файле записано одно целое число n – номер группы ($1 \le n \le 10^6$).
Выходные данные. В выходной файл вывести одно целое число – значение вычисленной суммы.

Примеры входного файла	Примеры выходного файла
10	1729
100	1970299

РЕШЕНИЕ:

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned long long n;
	ifs = fopen( "2.IN", "r" );
	fscanf( ifs, "%lld", &n );
	fclose( ifs );
	ofs = fopen( "2.OUT", "w" );
// Аналитическое решение
	fprintf( ofs, "%lld\n", (n-1)*(n-1)*(n-1)+n*n*n);
// Алгоритмическое решение
	unsigned long long m, s, i, j;
	m = (n-1)*(n-1);
	s = 0;
```

```
      for (i=1, j=m+1; i<2*n; i++, j++) s += j;
      fprintf( ofs, "%lld\n", s );
      fclose( ofs );
      return 0;
}
```

Задача 3: Текст (20)

Дан текст на английском языке. Предложения в тексте начинаются с заглавной буквы, а в конце предложения ставится "." (точка). Слова в предложениях и сами предложения отделяются друг от друга, по крайней мере, одним пробелом. Между словами допускаются символы пунктуации: "," (запятая), ";" (точка с запятой), ":" (двоеточие), "–" (тире). Символ апострофа считается частью слова. Определить, сколько в каждом предложении имеется слов и символов пунктуации.

Входные данные. Во входном файле записан текст на английском языке. Длина строки текста не более 60. Количество строк текста не более 1000. Длина слова не более 20. Длина предложения не более 200 символов.
Выходные данные. Каждую пару чисел (количество слов и количество символов пунктуации) печатать с новой строки.

Пример входного файла	Пример выходного файла
I'll tell you a story, a story anon, Of a noble prince, and his name was King John. For he was a prince, and a prince of great might, He held up great wrongs, he put down great right.	18 4 21 4

РЕШЕНИЕ:

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

FILE *ifs, *ofs;
void solve( char *stmt);

int _tmain(int argc, _TCHAR* argv[])
{
      char line[100];
      char stmt[300];
      char buff[300];
      int ch = '.';
      char *pdest;
      int result;
      ifs = fopen( "3.IN", "r");
      ofs = fopen( "3.OUT", "w");
      strcpy( stmt, " " );
      while (!feof( ifs ))
      {
            if (fgets( line, 100, ifs ) != NULL)
            {
                  pdest = strchr( line, ch );
                  while (pdest != NULL)
                  {
                        result = pdest - line + 1;
                        strncat( stmt, line, result);
                        solve( stmt );
                        strcpy( stmt, " " );
                        strcpy( buff, line + result );
                        strcpy( line, buff );
                        pdest = strchr( line, ch );
                  }
                  strcat( stmt, line );
            }
      }
      fclose( ifs );
      fclose( ofs );
      return 0;
}

void solve( char *stmt)
{
```

```
	char *token;
	char seps[] = " .,;:-\n\r";
	int i, countwords = 0, countseps = 0;
	for (i = 0; i < strlen( stmt ); i++)
		if (stmt[i] == '.' || stmt[i] == ',' || stmt[i] == ';' || stmt[i] == ':' ||
stmt[i] == '-') countseps++;
	token = strtok( stmt, seps );
	while (token != NULL)
	{
		countwords++;
		token = strtok( NULL, seps );
	}
	fprintf( ofs, "%d\t%d\n", countwords, countseps );
}
```

Задача 4: Выражение (25)

Обычный метод записи математических выражений, в которых бинарный оператор записывается между операндами, известен под названием инфиксной записи. При отсутствии скобок операции выполняются согласно правилам приоритета операторов. Для изменения порядка выполнения операций применяют скобки. Постфиксная польская запись – это форма записи математических выражений, в которой операнды расположены перед оператором. Если оператор имеет фиксированную арность, то в такой записи будут отсутствовать скобки, и она может быть интерпретирована без неоднозначности. Для заданного инфиксного арифметического выражения получить последовательность операторов в постфиксной польской записи.

Входные данные. Входной файл содержит одну строку, в которой записано инфиксное арифметическое выражение. В исходном выражении нет пробелов, в качестве операндов используются латинские буквы в верхнем или нижнем регистре, в качестве операторов используются знаки "+", "–", "*" и "/". Длина исходного выражения не превосходит 100.
Выходные данные. В выходной файл вывести одну строку, в которой записаны операторы постфиксной польской записи.

Пример входного файла	Пример выходного файла
A+B*(C+D)*(E+F)	+*+*+

РЕШЕНИЕ:

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

const int N = 100;

FILE *ifs, *ofs;

static int op_stack[N+1];
static int ex_stack[N+1];
static int op_index = 0, ex_index = 0;

void op_push( int value )
{
	op_stack[++op_index] = value;
}

int op_pop()
{
	op_index--;
	return (op_stack[op_index+1]);
}

int op_top()
{
	return (op_stack[op_index]);
}

void ex_push( int value )
{
	ex_stack[++ex_index] = value;
```

```
}

int is_operator( int c )
{
	if (c == '+' || c == '-' || c == '/' || c == '*' || c == '^') return 1;
	else return 0;
}

int op_priority( int op )
{
	if (op == '^') return 4;
	if (op == '*' || op == '/') return 3;
	if (op == '+' || op == '-') return 2;
	return 1;
}

int _tmain(int argc, _TCHAR* argv[])
{
	char line[N+1];
	int m, i;
	char ch, top;
	// Читаем строку выражения
	ifs = fopen( "4.IN", "r" );
	fgets( line, N, ifs );
	fclose( ifs );
	m = strlen( line );
	if (m == 0) return 1;
	// Реализуем алгоритм сортировочной станции
	op_stack[0] = 0;
	ex_stack[0] = 0;
	for (i = 0; i < m; i++)
	{
		ch = line[i];
		if (isalpha( ch )) ex_push( ch );
		else if (ch == '(') op_push( ch );
		else if (ch == ')')
		{
			top = op_top();
			while (top != '(' && op_index != 0)
			{
				ex_push( top );
				op_pop();
				top = op_top();
			}
			if (top == '(') op_pop();
		}
		else if (is_operator( ch ))
		{
			top = op_top();
			while (is_operator( top ) && op_priority( top ) >= op_priority( ch )
&& op_index != 0)
			{
				ex_push( top );
				op_pop();
				top = op_top();
			}
			op_push( ch );
		}
		else;
	}
	top = op_top();
	while (is_operator( top ) && op_index != 0)
	{
		ex_push( top );
		op_pop();
		top = op_top();
	}
```

```
	// Печатаем результат
	ofs = fopen( "4.OUT", "w" );
	for (i = 1; i <= ex_index; i++)
		if (!isalpha( ex_stack[i] ))
			fprintf( ofs, "%c", ex_stack[i] );
	fprintf( ofs, "\n" );
	fclose( ofs );
	return 0;
}
```

Задача 5: Видимая часть фигуры (30)

Ортогональную целочисленную решетку, состоящую из точек с целыми координатами в декартовой системе координат, будем обозначать через Z^2. В первом квадранте решетки Z^2 задано N фигур – прямоугольников и прямоугольных трапеций. Для всех прямоугольников две вершины лежат на оси X. Для всех трапеций две вершины боковой стороны, которая перпендикулярна основаниям, также лежат на оси X. Это позволяет задавать фигуру координатами левого верхнего и правого верхнего углов. Порядок задания фигур является существенным, т. к. фигура с меньшим номером может заслонять фигуру с большим номером. Для каждой фигуры вычислить ту часть ее площади, которая не заслоняется другими фигурами.

Входные данные. Первая строка входного файла содержит целое число N – количество фигур ($2 \le N \le 100$). В последующих N строках записаны четверки целых чисел x_1, y_1, x_2, y_2 ($0 \le x_1 < x_2 \le 10^4$; $0 < y_1, y_2 \le 10^4$), задающих координаты верхних углов фигур.
Выходные данные. Для каждой фигуры в выходной файл вывести вещественное число в диапазоне от 0 до 1, определяющее ту часть ее площади, которая не заслоняется другими фигурами. Вычисления проводить с точностью до 10^{-6}.

Примеры входного файла	Примеры выходного файла
4 2 3 7 5 4 6 9 2 11 4 15 4 13 2 20 2	1.00000000 0.38083333 1.00000000 0.71428571
5 200 1200 400 700 1200 1400 1700 900 5000 300 7000 900 8200 400 8900 1300 0 1000 10000 800	1.00000000 1.00000000 1.00000000 1.00000000 0.73667852

РЕШЕНИЕ:

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

using namespace std;

const double eps = 1e-9;
const int M = 100;

struct building
{
	int x1, y1, x2, y2;
} buildings[M];

double xs[M*M];
double yleft[M*M];
double yright[M*M];
int N, numX;

FILE *ifs, *ofs;

double Crossing(const building& b1, const building& b2);
double AreaBelow(double x1, double y1, double x2, double y2);

int _tmain(int argc, _TCHAR* argv[])
{
```

```
	ifs = fopen( "5.IN", "r" );
	fscanf( ifs, "%d", &N );
	for (int i = 0; i < N; ++i)
	{
		fscanf( ifs, "%d %d %d %d", &buildings[i].x1, &buildings[i].y1,
&buildings[i].x2, &buildings[i].y2 );
		xs[numX] = buildings[i].x1;
		yleft[numX] = 0;
		yright[numX] = 0;
		++numX;
		xs[numX] = buildings[i].x2;
		yleft[numX] = 0;
		yright[numX] = 0;
		++numX;
	}
	fclose ( ifs );
	ofs = fopen( "1.OUT", "w" );
	// Пересекаем каждую линию с каждой другой, запоминаем X-значения
	for (int i = 0; i < N; ++i)
	{
		for (int j = i+1; j < N; ++j)
		{
			double x = Crossing(buildings[i], buildings[j]);
			if(x > -1)
			{
				xs[numX] = x;
				yleft[numX] = 0;
				yright[numX] = 0;
				++numX;
			}
		}
	}
	sort(xs, xs+numX);
	numX = unique(xs, xs+numX) - xs;
	for (int i = 0; i < N; ++i)
	{
		building &b = buildings[i];
		double ownArea = AreaBelow(b.x1, b.y1, b.x2, b.y2);
		double visibleArea = 0;
		int ifirst = find(xs, xs+numX, (double)b.x1) - xs;
		int ilast = find(&xs[ifirst+1], xs+numX, (double)b.x2) - xs;
		double ycur = (xs[ifirst]-b.x1)/(b.x2-b.x1) * (b.y2-b.y1) + b.y1;
		double ynext;
		for(int icur = ifirst; icur != ilast; ++icur, ycur = ynext)
		{
			int inext = icur+1;
			double xcur = xs[icur];
			double xnext = xs[icur+1];
			ynext = (xnext-b.x1)/(b.x2-b.x1) * (b.y2-b.y1) + b.y1;
			if(ycur >= yright[icur] - eps && ynext >= yleft[inext] - eps)
			{
				visibleArea += AreaBelow(xcur, ycur, xnext, ynext);
				visibleArea -= AreaBelow(xcur, yright[icur], xnext,
yleft[inext]);
				yright[icur] = ycur;
				yleft[inext] = ynext;
			}
		}
		fprintf( ofs, "%.6lf\n", visibleArea / ownArea );
	}
	fclose( ofs );
	return 0;
}

double Crossing(const building& b1, const building& b2) // Returns x, < -1 if no
crossing
{
```

```
	if(b1.x2 <= b2.x1 || b2.x2 <= b1.x1) return -2;
	// Есть ли пересечение
	int x1 = max(b1.x1, b2.x1);
	int x2 = min(b1.x2, b2.x2);
	double y11 = (x1-b1.x1)/(double)(b1.x2-b1.x1) * (b1.y2-b1.y1) + b1.y1;
	double y12 = (x2-b1.x1)/(double)(b1.x2-b1.x1) * (b1.y2-b1.y1) + b1.y1;
	double y21 = (x1-b2.x1)/(double)(b2.x2-b2.x1) * (b2.y2-b2.y1) + b2.y1;
	double y22 = (x2-b2.x1)/(double)(b2.x2-b2.x1) * (b2.y2-b2.y1) + b2.y1;
	if(y11 >= y21 - eps && y12 >= y22 - eps) return -2;
	if(y11 - eps <= y21 && y12 - eps <= y22) return -2;
	// Находим x пересечения
	int f1 = b1.y2 - b1.y1;
	int g1 = b1.x1 - b1.x2;
	double h1 = f1*b1.x1 + g1*b1.y1;
	int f2 = b2.y2 - b2.y1;
	int g2 = b2.x1 - b2.x2;
	double h2 = f2*b2.x1 + g2*b2.y1;
	double det = f1*(double)g2 - f2*(double)g1;
	double x = (g2*h1 - g1*h2)/det;
	return x;
}

double AreaBelow(double x1, double y1, double x2, double y2)
{
	double area = (y1 + y2) * (x2 - x1) * 0.5;
	return area;
}
```