Решения олимпиадных задач.
2014 год. Заключительный этап. 8-9 классы. Билет 2.

Задача 1: Сумма (10)

Последовательность чисел Фибоначчи $\{F_0, F_1, F_2, \ldots\} = \{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, \ldots\}$ задается условиями $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ ($n \geq 0$). Для заданного числа n вычислить сумму

$$\frac{1}{2^1}+\frac{1}{2^2}+\frac{2}{2^3}+\frac{3}{2^4}+\frac{5}{2^5}+\frac{8}{2^6}+\frac{13}{2^7}+\ldots+\frac{F_n}{2^n}.$$

Входные данные. Во входном файле записано одно целое число n ($1 \leq n \leq 10^9$).
Выходные данные. В выходной файл вывести одно число – значение вычисленной суммы с точностью до трех знаков после запятой.

Пример входного файла	Пример выходного файла
10	1.772

РЕШЕНИЕ:

Надо избегать прямого вычисление 2^n, иначе при больших n возникнет переполнение.

```
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
	__int64 f_one, f_two, f_three;
	int n;
	double q, s, t;
	scanf("%d", &n);
	f_one = 0;
	f_two = 1;
	q = 1.0;
	s = 0;
	for (int i=1; i<=n; i++)
	{
		q = q/2.0;
		t = f_two*q;
		s = s+t;
		f_three = f_one + f_two;
		f_one = f_two;
		f_two = f_three;
	}
	printf("%.3f\n", s);
	return 0;
}
```

Задача 2: НОК и НОД (15)

Будем обозначать через $M(a, b, c, \ldots, k)$ наименьшее общее кратное, а через $D(a, b, c, \ldots, k)$ – наибольший общий делитель целых чисел $a, b, c, \ldots, k$. Вычислить

$$M(a,b,c)\cdot D(a,b)\cdot D\left(\frac{ab}{D(a,b)},c\right).$$

Входные данные. Входной файл содержит одну строку, в которой записаны три целых числа a, b и c ($1 \leq a, b, c \leq 10^3$).
Выходные данные. В выходной файл вывести одно целое число – значение вычисленного произведения.

Пример входного файла	Пример выходного файла
9 18 45	7290

РЕШЕНИЕ:

Можно показать, что НОК(a, b, c) = НОК(НОК(a, b), c) и a·b/НОД(a, b) = НОК(a, b).

```
#include "stdafx.h"

// Наибольший общий делитель (рекурсивная функция)
__int64 gcd(__int64 a, __int64 b)
{
	return (!b) ? a : gcd(b, a % b);
}

// Наименьшее общее кратное (рекурсивная функция)
__int64 lcm(__int64 a, __int64 b)
{
	return a / gcd(a, b) * b;
}

int _tmain(int argc, _TCHAR* argv[])
{
	__int64 a, b, c;
	scanf("%lld %lld %lld", &a, &b, &c);
	printf("%lld\n", lcm(lcm(a, b), c)*gcd(a, b)*gcd(lcm(a, b), c));
	return 0;
}
```

Задача 3: Площадь n-угольника (20)

Ортогональную целочисленную решетку, состоящую из точек с целыми координатами в декартовой системе координат, будем обозначать через $\mathbf{Z}^2$. На решетке $\mathbf{Z}^2$ задан простой n-угольник (т.е. без самопересечений, но не обязательно выпуклый). Вершины n–угольника находятся в узлах решетки $\mathbf{Z}^2$. Вычислить площадь n-угольника.

Входные данные. Первая строка входного файла содержит целое число n ($3 \le n \le 1000$) – количество вершин n-угольника. Далее следует n строк, задающих вершины n-угольника. Каждая такая строка содержит два целых числа x_i и y_i ($-1000 \le x_i, y_i \le 1000$) – координаты вершины n-угольника. Если соединить точки в данном порядке, а также первую и последнюю точки, получится заданный n-угольник.
Выходные данные. В выходной файл вывести одно число – площадь n-угольника с точностью до двух знаков после запятой.

Пример входного файла	Пример выходного файла
10 0 0 4 0 3 1 6 0 6 3 7 5 3 2 2 2 0 5 1 1	15.50

РЕШЕНИЕ:

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

int x[1000];
int y[1000];

double area(int n)
{
	double t, s = 0;
	for (int i = 0; i < n; i++)
	{
		if (i == 0) //если i == 0, то y[i-1] заменяем на y[n-1]
		{
			t = x[i]*(y[n-1] - y[i+1]);
			s += t;
		}
```

```
            else if (i == (n-1)) // если i == n-1, то y[i+1] заменяем на y[0]
            {
                  t = x[i]*(y[i-1] - y[0]);
                  s += t;
            }
            else
            {
                  t = x[i]*(y[i-1] - y[i+1]);
                  s += t;
            }
      }
      return (fabs(s/2));
}

int _tmain(int argc, _TCHAR* argv[])
{
      int n;
      scanf("%d", &n);
      for (int i = 0; i < n; i++)
            scanf("%d %d", &x[i], &y[i]);
      printf("%.2lf\n", area(n));
      return 0;
}
```

Задача 4: Польская запись (30)

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 арифметического выражения в постфиксной польской записи вычислить его значение.

Входные данные. Входной файл содержит одну строку, в которой записано арифметическое выражение в постфиксной польской записи. В исходном выражении операнды и операции разделены пробелами, в качестве операндов используются целые числа, в качестве операторов используются знаки "+", "–", "*", "/" и "%". Длина исходного выражения не превосходит 100.
Выходные данные. В выходной файл вывести одну строку, в которой записано одно число — значение арифметического выражения.

Примеры входного файла	Примеры выходного файла
1 2 + 4 * 5 + 3 -	14
12 13 7 - + 5 /	3
4 7 3 * 14 + - 8 9 10 * * /	0

РЕШЕНИЕ:

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

int stack[1000];
int index = 0;

void push(int a)
{
      stack[index++] = a;
}

int pop()
{
      return stack[--index];
}

int eval( char * s)
{
      char seps[] = " ";
```

```
	char *token;
	int a, b;
	token = strtok( s, seps );
	while (token != NULL)
	{
		if ((strlen( token ) == 1) && (	(token[0] == '+') ||
						(token[0] == '-') ||
						(token[0] == '*') ||
						(token[0] == '/') ||
						(token[0] == '%')))
			switch (token[0])
			{
			case '+':
				b = pop();
				a = pop();
				push( a + b );
				break;
			case '-':
				b = pop();
				a = pop();
				push( a - b );
				break;
			case '*':
				b = pop();
				a = pop();
				push( a * b );
				break;
			case '/':
				b = pop();
				a = pop();
				push( a / b );
				break;
			case '%':
				b = pop();
				a = pop();
				push( a % b );
				break;
			default:
				;
			}
		else
		{
			a = atoi( token );
			push( a );
		}
		token = strtok( NULL, seps );
	}
	return stack[0];
}

int _tmain(int argc, _TCHAR* argv[])
{
	char line[1000];
	gets( line );
	printf( "%d\n", eval( line ) );
	return 0;
}
```

Задача 5: Логика мышления (25)

В финальном шахматном турнире воинского соединения встретились 8 шахматистов, имевших следующие звания: полковник, майор, капитан, лейтенант, старшина, сержант, ефрейтор и рядовой. Среди них были пехотинец, летчик, танкист, артиллерист, минометчик, сапер, связист и военврач. Известно, что:

a) В первом туре полковник играл с летчиком, майор – с танкистом, сержант – с пехотинцем и старшина – с военврачом.
b) После первого тура рядовой выбыл из турнира. Из-за этого выходными оказались:
во II туре – минометчик, в III туре – капитан, в IV туре – сапер, в V туре – лейтенант, в VI туре – танкист, в VII туре – связист.

c) Во II туре полковник играл с артиллеристом, пехотинец – с танкистом, лейтенант – с сержантом.
d) В III туре ефрейтор выиграл у военврача, а партии полковника с сапером и майора с минометчиком окончились вничью.

Какую воинскую специальность имел каждый?

РЕШЕНИЕ:

полковник	связист
майор	артиллерист
капитан	пехотинец
лейтенант	военврач
старшина	минометчик
сержант	сапер
ефрейтор	танкист
рядовой	летчик