Всероссийская олимпиада школьников по информатике
II (муниципальный) этап
2015-2016 учебный год
9 - 11 классы
Разбор задач

Задача 1. Стрелка (100 баллов)

Исходными данными гарантируется, что вершины треугольника занимают ровно один пиксель. Поэтому можно построить описанный вокруг стрелки прямоугольник и, пройдясь по его границам, определить, на каких сторонах лежат ровно по одной точке. После этого не составит труда определить, куда смотрит стрелка.

Альтернативное решение − найти основание стрелки, то есть ту сторону прямоугольника стрелки, которая лежит на одной из сторон описанного вокруг стрелки прямоугольника.

Можно решить эту задачу и другими способами.

Пример решения на Free Pascal

```
uses math;

const
  NMAX = 33;

var
  m, n, left, top, right, bottom, i, j, count: longint;
  a: array[1..NMAX] of String;

begin
  readln(m, n);
  for i := 1 to m do
    readln(a[i]);
  left := n;
  right := 1;
  top := m;
  bottom := 1;
  for i := 1 to m do
    for j := 1 to n do
      if (a[i][j] = '*') then
      begin
        left := min(left, j);
        right := max(right, j);
```

```
        top := min(top, i);
        bottom := max(bottom, i);
      end;

  // check right
  count := 0;
  for i := top to bottom do
    if a[i][left] = '*' then
      inc(count);
  if count >= 3 then begin
    writeln('Right');
    halt(0);
  end;

  // check left
  count := 0;
  for i := top to bottom do
    if a[i][right] = '*' then
      inc(count);
  if count >= 3 then begin
    writeln('Left');
    halt(0);
  end;

  // check down
  count := 0;
  for j := left to right do
    if a[top][j] = '*' then
      inc(count);
  if count >= 3 then begin
    writeln('Down');
    halt(0);
  end;

  writeln('Up');
end.
```

Задача 2. Игра на клетчатом поле (100 баллов)

Если M либо N равно 1, то, очевидно, выигрывает первый игрок. Теперь рассмотрим два более общих случая.

1). M и N имеют одинаковую чётность. При этом всегда выигрывает второй игрок. Его возможная выигрышная стратегия может быть следующей. После того как первый игрок вычеркнет

какую-то строку или столбец, чётность количества строк и количества столбцов станет разной. Если стало нечётно количество строк, то второй игрок должен вычеркнуть любую строку, иначе - любой столбец. В результате у противника будет чётное количество и строк, и столбцов, поэтому одним ходом он выиграть не сможет, а после его хода чётность станет опять разной. Дальше описанный процесс повторяется. В конце концов, второй игрок вычеркнет последнюю оставшуюся строку или столбец и победит.

2). M и N имеют разную чётность. В этом случае выигрывает первый игрок. Он может вычеркнуть любую строку и столбец, при этом задача сведётся к предыдущему варианту (только в роли первого игрока теперь будет выступать второй).

Однако, здесь есть один частный случай: если количество строк (столбцов) равно 2, то такие строки (столбцы) вычеркивать нельзя.

Пример решения на Free Pascal:

```
m, n: longint;
begin
  read(m, n);
  if (m = 1) or (n = 1) then begin
     writeln(1);
     writeln(1);
  end else
    if m mod 2 = n mod 2 then
      writeln(2)
    else begin
      writeln(1);
      if (m = 2) or (n = 2) then
        writeln(m + n - 2)
      else
        writeln(m + n);
    end;
end.
```

Задача 3. Факториал (100 баллов)

Рассмотрим сначала, от чего зависит количество нулей на конце N! в десятичной системе счисления. Каждый нуль в конце числа возникает от произведения простых множителей 2 и 5. Таким образом, количество нулей на конце числа − это максимум из количества двоек и пятёрок в разложении числа на простые множители. А поскольку

количество пятёрок в *N*! не превышает количества двоек, то количество нулей определяется лишь количеством пятёрок. Например, число 30! оканчивается 7 нулями, поскольку оно содержит по одной пятёрке в множителях 5, 10, 15, 20, 25, 30 и ещё одну дополнительную пятёрку в множителе 25. Заметим, что для произвольного N количество пятёрок в *N*! можно найти по формуле N div 5 + N div 25 + N div 125 + … N div 5^i… (вычисление заканчиваются, когда знаменатель превысит числитель).

Аналогичным образом можно определить количество нулей на конце *N*! и для других систем счисления. Например, запись числа 50! в системе с основанием 14 заканчивается 8 нулями − количество множителей 7: (50 div 7 + 50 div 7^2) = 8.

Немного более сложная ситуация возникает, когда основание системы счисления представляет собой простое число в некоторой степени. Пример − основание 16 = 2^4. В таком случае найденное количество двоек нужно еще поделить на степень множителя. Например, число 10! имеет (10 div 2 + 10 div 4 + 10 div 8) = 8 двоек. Поскольку 8 div 4 = 2, то запись числа 10! в шестнадцатеричной системе имеет два нуля на конце.

Обратим ещё внимание на основание 12. Это единственное основание (от 2 до 16), которое имеет в разложении разные простые множители, среди которых есть повторяющийся: 12 = 2^2*3. Заметим, что для некоторых значений *N* количество двоек, делённое пополам, может оказаться меньше количества троек, поэтому именно оно будет определять число нулей на конце (например, для *N* = 30).

На основе данных рассуждений сравнительно несложно написать решение, которое перебирает все возможные основания, для каждого основания считает количество нулей и запоминает наилучший вариант. Однако, попробуем получить более простое и быстрое решение.

Какое основание системы счисления будет давать наименьшее количество нулей для достаточно больших *N*? Можно предположить, что это основание 13, поскольку это самое большое простое число от 2 до 16, и оно будет встречаться в *N*! наиболее редко. Действительно, данное предположение верно − для всех достаточно больших N (начиная с *N*=66) ответом задачи будет 13.

Однако, для небольших *N* может существовать основание меньше 13, дающее такое же число нулей. Например, для *N*=65 ответом будет 11, поскольку 65! содержит одинаковое количество множителей 11 и 13. Можно заметить, что для *N* в интервале от 14 до 65 ответом всегда будет или 11, или 13 − смотря каких множителей больше.

При N меньше 14 ответами могут быть и другие основания, причём даже не обязательно простые числа. Например, для N=3 ответ равен 4. Эти оставшиеся случаи можно аккуратно рассмотреть вручную.

Пример решения на Free Pascal:

```
var
   n: longint;
begin
  read(n);
  if n >= 66 then
    write(13)
  else if n >= 14 then
    if n div 13 < n div 11 then
      write(13)
    else
      write(11)
  else if n = 13 then
    write(7)
  else if (n = 11) or (n = 12) then
    write(13)
  else if (n >= 7) and (n <= 10) then
    write(11)
  else if (n = 5)  or (n = 6) then
    write(7)
  else write(n + 1);
end.
```

Задача 4. Спам-фильтр (100 баллов)

Создадим два массива *letters* и *spam*. В первом будут находиться все письма (в том числе спам), во втором − только спам.

Заметим, что в качестве возможных значений порога достаточно проверить число 1, а также оценку каждого письма, увеличенную на 1. Действительно, только при таких аргументах может происходить изменение функции $F1$. Среди всех вариантов выбирается тот, на котором значение $F1$ оказалось максимально.

Рассмотрим теперь, каким образом быстро вычислять функцию $F1$ для заданного аргумента t. Нам нужно уметь для заданного t быстро находить количество всех писем с оценкой $\geq t$ и количество спам-писем с оценкой $\geq t$. Для этого предварительно отсортируем массивы *letters* и *spam*, и тогда мы можем воспользоваться

алгоритмом двоичного поиска. Заметим, что в языке C++ можно воспользоваться уже готовой функцией *lower_bound*, в языке Java - функцией *binarySearch*. Для языка Pascal такую функцию несложно написать самим.

Вычислительная сложность алгоритма составляет Θ(N•log(N)).

Пример решения на Free Pascal

```
{$MODE objfpc}
uses classes;

type
  TLetter = class
    mark: longint;
    spam: boolean;
    constructor Create(m: longint);
  end;

constructor TLetter.Create(m: longint);
begin
  mark := m;
  spam := false;
end;

function LetterCompare(letter1, letter2 : Pointer): Integer;
begin
  if TLetter(letter1).mark = TLetter(letter2).mark then
    LetterCompare := 0
  else if TLetter(letter1).mark < TLetter(letter2).mark then
    LetterCompare := -1
  else
   LetterCompare := 1;
end;

var
  letters, spam: TList;
  n, k, mark, ind, i, t, tOpt: longint;
  f, fMax: double;

function lower_bound(list: TList; t: double): longint;
var
  left, right, mid: longint;
begin
  lower_bound := list.Count;
```

```
  left := 0;
  right := list.Count - 1;
  while (left <= right) do begin
    mid := (left + right) div 2;
    if TLetter(list[mid]).mark >= t then begin
      lower_bound := mid;
      right := mid - 1;
    end else
      left := mid + 1;
  end;
end;

function f1(t: longint): double;
var
  spamCount, allCount: longint;
  precision, recall: double;
begin
  spamCount := spam.Count - lower_bound(spam, t);
  allCount := letters.Count - lower_bound(letters, t);
  precision := spamCount / allCount;
  recall := spamCount / spam.Count;
  if (precision = 0) and (recall = 0) then
    f1 := 0
  else
    f1 := precision * recall / (precision + recall);
end;

begin
  read(n);
  letters := TList.Create();
  for i := 0 to n - 1 do begin
    read(mark);
    letters.Add(TLetter.Create(mark));
  end;

  read(k);
  spam := TList.Create();
  for i := 0 to k - 1 do begin
    read(ind);
    dec(ind);
    TLetter(letters[ind]).spam := true;
    spam.add(letters[ind]);
  end;
```

```
  letters.sort(@LetterCompare);
  spam.sort(@LetterCompare);

  fMax := -1;
  tOpt := -1;
  for i := 0 to letters.Count - 1 do begin
    if i = 0 then
      t := 1
    else
      t := TLetter(letters[i - 1]).mark + 1;
    f := f1(t);
    if f > fMax then begin
      tOpt := t;
      fMax := f;
    end;
  end;
  writeln(tOpt);
end.
```

Примеры других решений задач на языках C++ и Java можно найти в материалах олимпиады.

Всероссийская олимпиада школьников по информатике
II (муниципальный) этап
2015-2016 учебный год
9-11 классы
Материалы для проверки

Введение

Для каждой задачи в жюри передаётся в электронном виде набор тестов и, если в задаче неоднозначный ответ, проверяющих программ.

Каждая задача оценивается в 100 баллов. За прохождение каждого теста даётся одинаковое число баллов. Если тест не пройден полностью, за него не следует давать ничего. В случае, если ответ задачи однозначен, расхождение с эталонным ответом допускается с точностью до лишних пробелов и/или символов перевода строки. В случае, если по задаче есть проверяющая программа, баллы за тест следует давать только когда ответ проверяющей программы 'ac', 'accepted' или 'ok'.

Не допускается внесение каких-либо изменений в систему оценивания без согласования с предметно-методической комиссией соответствующего этапа.

Система оценки задач для 9 - 11 классов

Задача	Тестов	Баллов за один тест	Проверяющая программа
1. Стрелка	20	5	не обязательна
2. Игра на клетчатом поле	20	5	не обязательна
3. Факториал	25	4	не обязательна
4. Спам-фильтр	20	5	не обязательна

Использование специализированных систем проведения соревнований во время тура

Жюри муниципального этапа при желании может принять решение об установке и использовании какой-либо специализированной системы проведения соревнований во время тура (например, EJudge). В этом случае каждый участник получает уникальный логин и пароль и сдаёт задачи в систему самостоятельно.

Обратите внимание! Проверка решений во время основного тура должна вестись только на тестах из примеров в условиях! Тесты из примеров лежат в папке *preliminary* для каждой задачи. Цель такой проверки − убедиться, что решение участника компилируется и соблюдает правильный формат ввода-вывода. В случае, если решение не проходит тесты из примера, участник может исправить ошибки и сдать его повторно. На окончательную проверку принимается последнее сданное решение, которое прошло все тесты из примера.

Окончательная проверка решений на основном наборе тестов производится после окончания тура.

Обращаем внимание, что в случае использования специализированной системы проведения соревнований пробный тур становится обязательным. Задачи для пробного тура имеются в прилагаемых материалах.

Автоматизация проверки решений на основном наборе тестов после окончания тура

Если во время тура использовалась специализированная система проведения соревнований, то, по всей видимости, проверка решений на основном наборе тестов после окончания тура может быть выполнена с помощью этой же системы.

Рассмотрим, каким образом можно выполнять проверку решений, если во время тура специализированных систем не применялось. Самый трудоёмкий способ заключается в последовательном запуске проверяемой программы на каждом тесте из заданного комплекта тестов для этой задачи. Для этого способа вполне достаточно иметь для каждого теста файл с входными данными, файл с соответствующими выходными данными и проверяющую программу.

Заметим, что даже при ручной проверке данные не требуется вводить вручную, а можно использовать механизм перенаправления потоков. Пусть `program.exe` — исполняемый файл программы, написанной участником олимпиады, входные данные расположены в файле `01.in`, а результат нужно получить в файле `01.res`. Тогда командная строка для запуска программы может выглядеть так:

```
program.exe <01.in >01.res
```

Поскольку ответы ко всем задачам в этом году однозначны, то нет необходимости в использовании специализированных проверяющих программ. Для сравнения файла с ответом участника и файла с правильным ответом в ОС Windows можно использовать команду `fc /W`, например:

```
fc /W 01.res 01.out
```

Хотя вышеописанный способ проверки возможен, но он весьма трудоёмок. Кроме того, способ не совсем корректен в плане проверки ограничений времени — сложно точно оценить, работала ли программа меньше одной секунды или больше. Поэтому рекомендуется использовать специальные средства для проверки решений участников:

1). Автоматическая проверяющая система, доступная по адресу http://prglab.cloudapp.net (резервный сервер доступен по адресу http://avt.vstu.edu.ru/acm). Инструкция по работе с системой приведена ниже.

2). Если по каким-то причинам отсутствует возможность использовать автоматическую проверяющую систему, можно воспользоваться программой "Тестер", она включена в архив

материалов олимпиады. Краткая инструкция по работе с ней приведена в конце данного документа.

3). Наконец, вполне возможно самостоятельное развёртывание другой проверяющей системы − например, EJudge (https://ejudge.ru).

Обратите внимание! Если при проверке решения произошла ошибка компиляции, то жюри после изучения протокола тестирования может попытаться устранить причины ошибки компиляции, выбрав другой компилятор с этого языка или даже внеся исправления в исходный код программы (но не модифицируя при этом алгоритм решения задачи).

Если при автоматической проверке программа не прошла ни одного теста, то, возможно, причина заключается в неправильном формате ввода-вывода (например, лишний вывод наподобие "*Введите число*" или "*Ответ равен*", лишний *readln* в конце программы на Pascal и тому подобное). В этом случае допустимо, если член жюри внесёт изменения в текст программы так, чтобы ввод/вывод удовлетворял требованиям автоматической тестирующей системы. ***При этом не допускается модифицировать алгоритм решения задачи***, то есть все исправления должны касаться только ввода-вывода или каких-либо частей программы, не связанных с алгоритмом решения (подключение модулей, библиотек и т. п.)

Внеся необходимые изменения, жюри может повторно отправить программу в автоматическую тестирующую систему, повторяя этот процесс при необходимости.

Рекомендации участникам по оформлению решений для автоматизированной проверки

Желательно заранее ознакомить участников с данными рекомендациями.

а). Программа не должна выводить ничего лишнего — наподобие ”Введите число N” или ”Ответ равен ”. Проверяющая система посчитает этот вывод за ответ и начислит ноль баллов.
б). Программа не должна ждать реакции от пользователя после вывода ответа. Например, для программ на языке Pascal не следует ставить пустой `readln` в конце программы.
в). Необходимо строго соблюдать регистр символов. Например, если в задаче требуется вывести слово ‘YES’, то ответ ‘Yes’ будет признан неверным.
г). Участникам, программирующим на языке Pascal, рекомендуется использовать среды Free Pascal или Pascal ABC. Компилятор Turbo Pascal на проверяющем сервере не поддерживается.
д). В Pascal не следует использовать модуль crt, а в C++ - conio.h
е). Функция int main() на языке C++ должна обязательно содержать `return 0;`
ж). Программа на языке Java должна содержать только один public класс с произвольным именем, который должен содержать метод `public static void main(String[] args)`. Также программа может содержать любое число вложенных и глобальных не public классов.
Ниже приведены примеры правильных программ на разных языках, решающие задачу о нахождении суммы двух целых чисел.

Pascal:

```
var
  a, b: longint;
begin
   read(a); { readln в этом случае не годится! }
   read(b);
   writeln(a + b);
end.
```

C++:

```
#include <iostream>
using namespace std;

int main(void) {
  long a, b;
  cin >> a >> b;
```

```
  cout << a+b;
  return 0;
}
```

Java:

```
import java.io.*;
import java.util.*;

public class a_plus_b
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);
      PrintWriter out = new PrintWriter(System.out);

      int a = in.nextInt();
      int b = in.nextInt();
      out.println(a + b);
      out.close();
   }
}
```

Python

```
str = input ()
elem = str.split (' ')
a = int (elem [0])
b = int (elem [1])
print (a + b)
```

C#

```
using System;
public class Sum {
    private static void Main() {
        string[] tokens = Console.ReadLine().Split(' ');
        Console.WriteLine(int.Parse(tokens[0]) +
int.Parse(tokens[1]));
    }
}
```

Visual Basic .NET

```
Module module1
Sub Main()
  Dim v() As String = Console.ReadLine().Split(" ")
  dim a as integer = integer.Parse(v(0))
  dim b as integer = integer.Parse(v(1))
  dim c as integer = a + b
  Console.WriteLine(c)
End Sub
End Module
```

Инструкция по проверке решений после окончания тура с помощью автоматической проверяющей системы

1). Откройте сайт http://prglab.cloudapp.net и нажмите ссылку «Добро пожаловать!». *В случае недоступности данного сайта можно воспользоваться резервным вариантом по адресу* http://avt.vstu.edu.ru/acm *(потребуется повторная регистрация).*

2). Нажмите ссылку «Регистрация». Заполните регистрационную форму в соответствии с нижеприведённой таблицей (если вы уже зарегистрированы, то войдите в систему и введите кодовое слово для получения доступа к задачам - порядок действий описан в пункте 3).

Поле на форме	Комментарий
Логин	Может содержать только английские буквы, цифры и знак подчеркивания. Например, **mo_vologda**
Отображаемое имя	Например, **Муниципальный этап Вологда** *(не более 50 символов)*
Ваш e-mail	E-mail для связи
Пароль	В английской раскладке клавиатуры! Если у вас на компьютере стоит программа наподобие Punto Switcher, временно отключите её.
И ешё раз пароль	
Дополнительная информация	Можно не указывать
Страна	Оставьте по умолчанию
Кодовое слово	**`infolymp15`** Все буквы в слове маленькие. Кодовое слово позволит вам получить доступ к проверке олимпиадных задач. Не сообщайте его посторонним.

Пример заполнения формы:

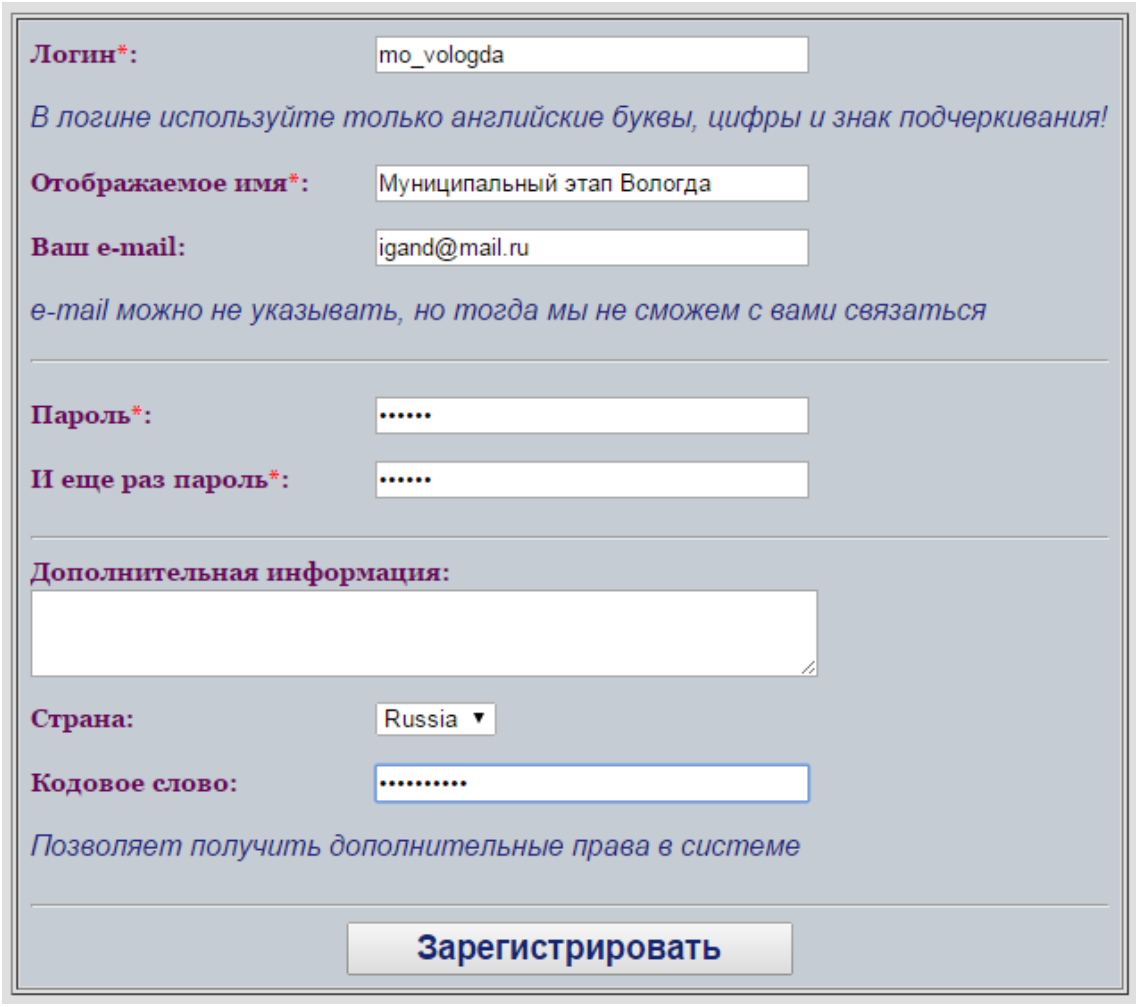


Рис.1. Регистрация нового пользователя

3). Нажмите на кнопку «Задачи». Слева будут показаны темы (разделы). Раскройте тему «**Олимпиада по информатике 2015 - муниципальный этап**», затем перейдите во вложенную тему «**Пробные задачи**». Если всё получилось, то переходите к пункту 4.

Если у вас не отображается тема «Олимпиада по информатике 2015 - муниципальный этап», то, вероятно, вы не ввели или неверно ввели кодовое слово при регистрации. Попробуйте ввести его повторно. Для этого найдите на web-странице строчку «**Вы вошли в систему как**», сразу после неё написано ваше имя в виде гиперссылки. Нажмите на эту ссылку, а затем на ссылку «**Редактировать профиль**». В открывшейся форме введите правильное кодовое слово, другие поля при этом не меняйте. Нажмите кнопку «**Сохранить**», затем кнопку «**Задачи**».

4). В теме «Пробные задачи» имеется одна несложная задача с именем «Точные квадраты». Откройте её условие (щелкнув по названию). Попробуйте написать и послать решение на каком-нибудь языке программирования. Для отправки решения сделайте следующие действия:

- нажмите ссылку «**Послать на проверку**».
- выберите компилятор из выпадающего списка.
- вставьте текст решения в поле «Исходный текст» **либо** нажмите кнопку «Выберите файл» и выберите файл с исходным текстом. *Сразу и то, и другое делать не нужно.*
- напишите что-нибудь в поле "Комментарий" (при проверке задач основного тура здесь будет писаться шифр участника, чьё решение проверяется).
- нажмите кнопку «**Отправить**». Пример показан на следующем рисунке.

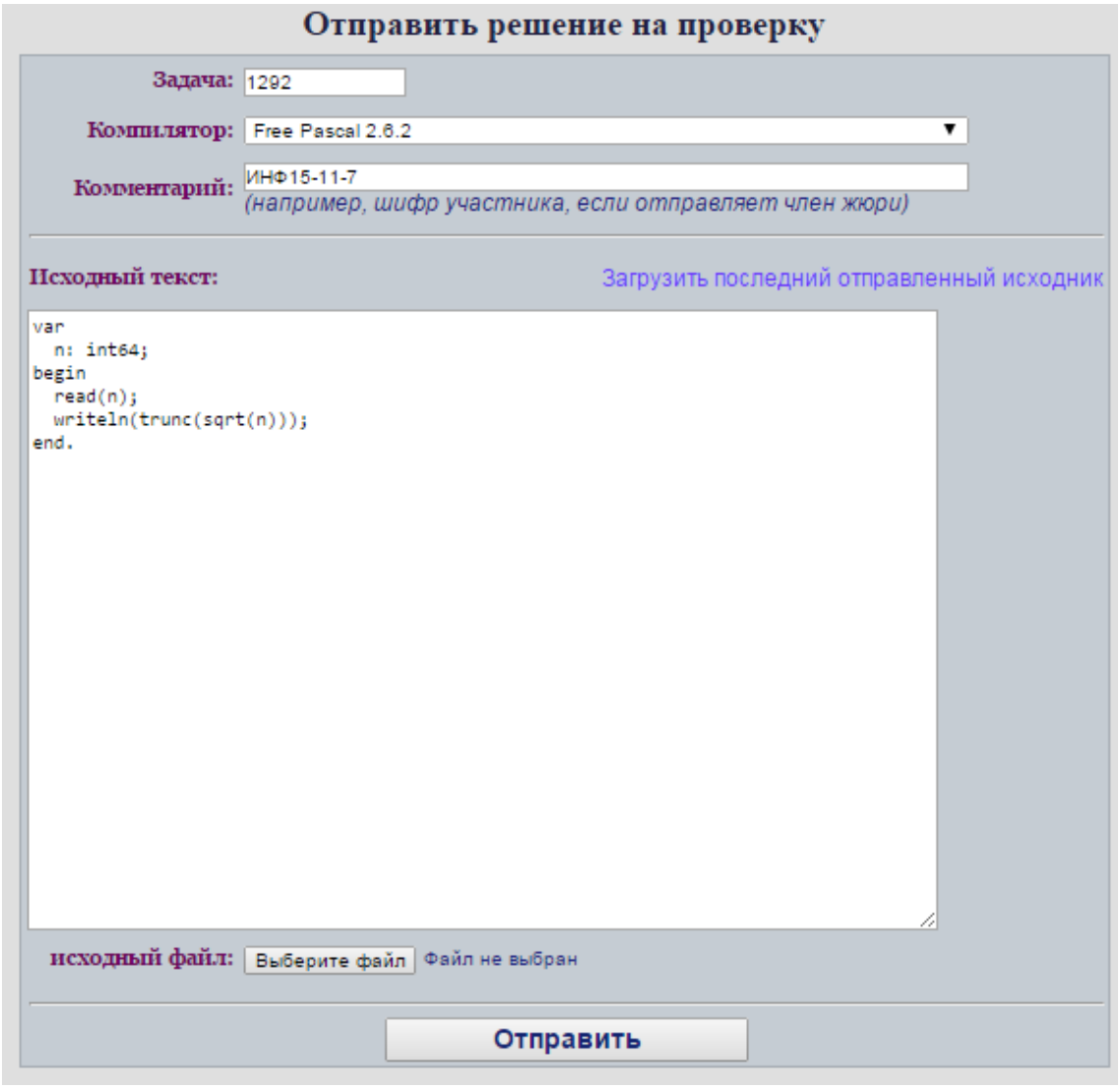


Рис.2. Отправка решения на проверку

Откроется окно «**Online статус**». В выпадающем списке «**Показать решения**» выберите вариант «**Свои**» и нажмите кнопку ”**OK**”. В результате Вы будете видеть результаты проверки только тех решений, которые посылали сами.

Результаты проверки последнего отправленного решения будут показаны **в верхней строке** таблицы. Ниже приведены возможные результаты проверки и комментарии.

Результаты проверки решений на странице «Онлайн статус» и в детальном отчёте

Результат проверки	**Комментарий**
Ждёт	Вероятно, сервер занят проверкой других решений. Подождите несколько секунд и обновите web-страницу
Компилируется	На сервере происходит компиляция вашего решения. Подождите несколько секунд и обновите web-страницу
Выполняется	Ваше решение выполняется на наборе тестов. Подождите несколько секунд и обновите web-страницу
Верно	Решение полностью верно. Баллы за решение показываются в столбце «Баллы»
Частично верно	Решение работает верно не на всех тестах. Набранные баллы за решение показываются в столбце «Баллы». В поле «Тест» показан номер первого не пройденного теста.
Неправильный ответ	
Ошибка представления	Формат ответа не соответствует ожидаемому. Например, программа выдала строку вместо числа или отрицательное число, когда в ответе ожидаются только положительные. Данная ошибка также возникает, когда программа завершилась с ненулевым кодом возврата — например, если функция main() в программе на языке C++ не содержит return 0;
Ошибка выполнения	Программа аварийно завершилась. Например, произошёл выход за границу массива, деление на ноль, обращение к памяти по неверному указателю, переполнение стека при глубокой рекурсии и др.
Предел времени	Программа работает слишком долго, «зависла» или ожидает ввода от пользователя (проверьте, что в конце программы на Pascal не стоит пустой readln)
Предел памяти	Программа использовала слишком много памяти. Вероятно, это произошло вследствие ошибки в программе, поскольку для задач муниципального этапа ограничения на объём памяти будут выставлены с большим запасом
Нарушение безопасности	Зафиксирована попытка обращения из программы к файлам операционной системы, к сети и т.п.
Ошибка компиляции	Вероятно, программа содержит синтаксические ошибки. Проверьте, та ли версия компилятора выбрана (например, Pascal ABC.NET и Free Pascal во многом несовместимы)
Обнаружено бездействие	Программа не использует процессорное время, но и не завершается (например, ожидает нажатия клавиши). ***Данный результат возможен в связи с кратковременной перегрузкой сервера - попробуйте послать задачу повторно.***

Для многих из перечисленных результатов проверки можно посмотреть дополнительную информацию. Для этого нужно нажать на маленькое изображение лампочки в столбце «Результат» (см. рисунок 3).

Online статус

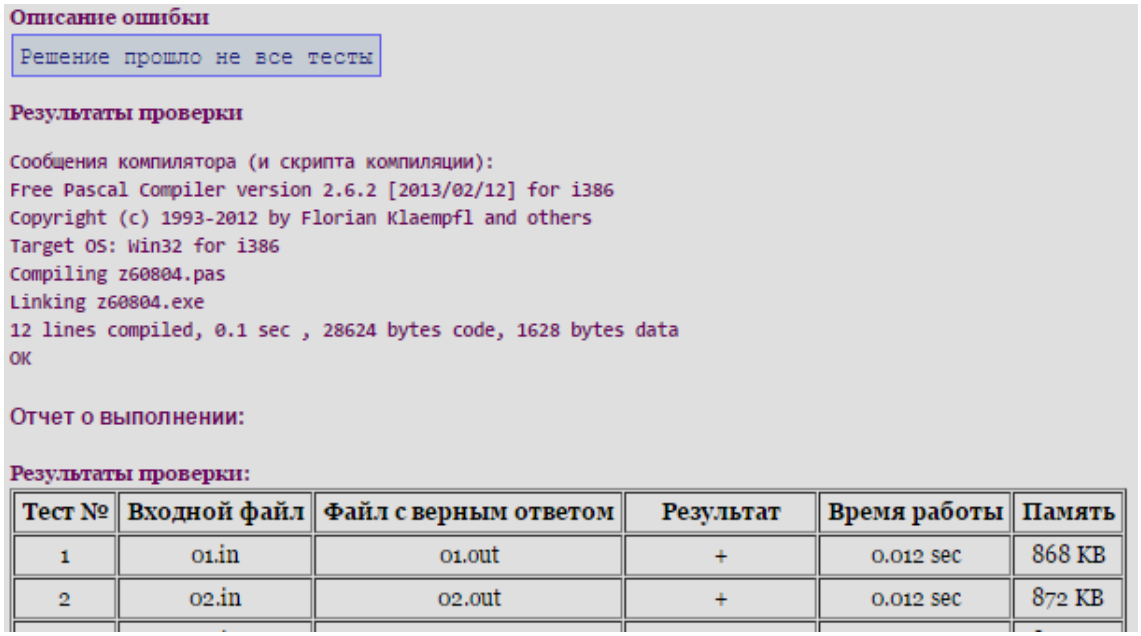
Показать решения: Свои ▾ OK

Номер	Дата	Задача	Компилятор	Результат	Комментарий	Тест	Баллы	Время работы (сек)	Исп-но памяти (KB)
395268	10/18/2015 11:47:32 PM	1292. Точные квадраты	Free Pascal 2.6.2	Частично верно	*ИНФ15-11-7*	16	75	2.9844	602
395267	10/18/2015 11:36:56 PM	1292. Точные квадраты	Free Pascal 2.6.2	**Верно**		исх. код	**100**	**0.0136**	**602**
395259	10/16/2015 1:02:54 AM	1. A + B	Free Pascal 2.6.2	Неправильный ответ		-1		0.0132	602

Рис. 3. Результаты проверки (последние решения сверху, включена опция "показывать только свои решения")

Например, для статуса «Частично верно» при нажатии на лампочку будет показан отчёт о прохождении всех тестов (см. рисунок 4). Это может быть полезно, например, при проведении апелляции — можно посмотреть номер первого не пройденного теста и запустить программу участника на этом тесте на своём компьютере.

Для просмотра исходного текста решения нужно нажать на маленькое изображение в виде страницы с текстом в столбце «Номер».



Описание ошибки

Решение прошло не все тесты

Результаты проверки

```
Сообщения компилятора (и скрипта компиляции):
Free Pascal Compiler version 2.6.2 [2013/02/12] for i386
Copyright (c) 1993-2012 by Florian Klaempfl and others
Target OS: Win32 for i386
Compiling z60804.pas
Linking z60804.exe
12 lines compiled, 0.1 sec , 28624 bytes code, 1628 bytes data
OK
```

Отчет о выполнении:

Результаты проверки:

Тест №	Входной файл	Файл с верным ответом	Результат	Время работы	Память
1	01.in	01.out	+	0.012 sec	868 KB
2	02.in	02.out	+	0.012 sec	872 KB

Рис. 4. Отчёт о проверке для статуса «Частично верно»

Заметим, что в случае статуса «Верно» возможно срабатывание подсистемы контроля плагиата, если решение случайно окажется похоже на чьё-то другое. В большинстве случаев совпадение случайно, и его можно игнорировать.

5). Рекомендации по выполнению проверки решений

Проверка решений выполняется жюри после того, как участники завершили решение задач и сдали жюри свои решения.

Для проверки решений зайдите в тему «Олимпиада по информатике 2015 - муниципальный этап» и выберите нужный класс. После отправки первого решения на web-странице «Онлайн статус» желательно ещё раз проверить, что в списке «Показать решения» выбран вариант «Свои».

Допустимо отправлять на проверку сразу несколько решений друг за другом — они ставятся в очередь и будут проверяться по мере освобождения сервера (при этом результат «Ждёт» будет меняться на результат проверки).

При отправке решения в поле "Комментарий" рекомендуется указывать шифр участника − это упрощает отслеживание, чьё именно решение проверилось.

Суммарный балл по всем задачам для каждого участника подсчитывается жюри самостоятельно.

Инструкция по проверке решений с помощью программы "Тестер"

Если по каким-то причинам отсутствует возможность использования автоматической проверяющей системы через Интернет, то можно воспользоваться свободно распространяемой программой "Тестер" (http://acm.timus.ru/tester). Данная программа имеется в архиве материалов олимпиады.

Рассмотрим порядок действий для проверки решения. Пусть участник сдал на проверку решение, которое называется program.pas. Скомпилируем решение, чтобы получился файл program.exe.

Скопируйте файлы program.exe и !test.exe в каталог с материалами по проверяемой задаче.

В командной строке перейдите в каталог с материалами по задаче и выполните команду `"!test.exe program.exe"`. На экран выведется протокол проверки:

```
C:\>y:

Y:\>cd squares

Y:\squares>!test.exe program.exe
Test 01: 0.015 0.016 3000K ok Ok
Test 02: 0.013 0.016 3040K ok Ok
...
Test 20: 0.014 0.000 3000K ok Ok
Score: 100
Press any key to continue...
```

Примечание 1. Программа "Тестер" не поддерживает тестирование 16-битных приложений с перенаправлением ввода/вывода. Поэтому не используйте компиляторы, создающие 16-битные программы. Например, вместо Turbo Pascal используйте Free Pascal или Delphi.

Примечание 2. Программа "Тестер" может не запускать .NET-программы (в том числе скомпилированные Pascal ABC.NET) на 64-битных операционных системах. В этом случае следует использовать 32-битную операционную систему, запущенную на другом компьютере или в виртуальной машине.

Рассмотрим кратко особенности проверки решений на разных языках. В случае, если нужно проверить решение на Java, то необходимо выполнить компиляцию и скопировать все полученные файлы с расширением ".class" в каталог с материалами задач. Пусть файл Main.class - класс, содержащий функцию main. Тогда запуск проверки выполняется командой:

```
!test.exe "C:\Program Files\Java\jdk1.8.0_31\bin\java.exe" Main
```

Разумеется, путь к файлу java.exe нужно заменить на свой. Используйте 32-битную версию Java, поскольку запуск 64-битной версии Java программой "Тестер" не поддерживается.

Аналогичным образом выполняется проверка решений на интерпретируемых языках. Например, для Python команда выглядит так:

```
!test.exe C:\Python33\python.exe program.py
```