II Областная олимпиада школьников по информатике
2017-2018 учебный год
9-10 классы
Отборочный тур

Разбор задач

Задача 1. Украшение (10 баллов)

Подсчитаем, какое наименьшее количество фиолетовых колечек может быть в цепочке. Можно считать, что первое колечко – жёлтое. Поскольку рядом с каждым жёлтым колечком обязательно есть фиолетовое, то три жёлтых колечка не могут идти подряд. Следовательно, среди каждых трёх последовательно идущих колечек хотя бы одно колечко должно быть фиолетовым. Тогда наименьшее количество фиолетовых можно получить как округление вверх от числа $N / 3$, и ответ будет следующим:

$$N - ((N + 2) / 3)$$

Возможны и другие верные ответы – например, $-(2 + N) / 3 + N$, поэтому проверку решения нельзя выполнять простым сравнением файлов – проверка осуществляется подстановкой тестовых наборов значения N в формулу участника.

Задача 2. Треугольники (10 баллов)

Рассмотрим несколько способов решения данной задачи.

Первое решение можно назвать "программистским". Очевидно, что можно искать только различные треугольники с общей вершиной 1 (так как для любого треугольника, не содержащего вершину 1, всегда найдётся равный ему треугольник, который содержит эту вершину).

Пройдём от вершины 1 по рёбрам многоугольника в порядке обхода, отступим a рёбер. Мы получили возможную вторую вершину. Теперь пойдём дальше от неё в порядке обхода и отступим b рёбер − мы получили третью вершину. Тогда, чтобы вернуться в вершину 1, нужно пройти ещё c рёбер, где $c = N - a - b$. Каждая допустимая пара $<a, b>$ задаёт какой-то треугольник (правда, некоторые из них будут равны). Чтобы получать только различные треугольники, можно перебирать треугольники в порядке неубывания длин их сторон (а длина стороны тем больше, чем больше длина пути по рёбрам между вершинами). То есть нужно перебрать все такие пары $<a, b>$, что $a \le b \le N-a-b$.

Правда, можно заметить, что между любыми двумя вершинами существуют не один путь по рёбрам, а два (по часовой стрелке и против), и длина стороны определяется наименьшим из них. Тем не менее, полученное выше правило перебора всё-таки остаётся верным. Действительно, a достаточно перебирать лишь до N div 3, поэтому длина кратчайшего пути по рёбрам от первой до второй вершины будет всегда a (не $N-a$). Величина b никогда не окажется больше $N/2$, (поскольку $b \le N-a-b$), поэтому длина кратчайшего пути по рёбрам от второй вершины до третьей всегда равна b (не $N-b$). Что же касается пути от вершины 1 до вершины 3, то его длина, и правда, может оказаться меньше c. Но даже в этом случае она всё равно будет больше или равна $a+b$, то есть треугольники всё же будут перебираться в порядке неубывания длин сторон, поэтому дубликатов не будет.

Приведём программу на C++, которая выполняет такой перебор:

```
#include <iostream>

int main() {
  int n;
  std::cin >> n;
  int count = 0;
  for (int a = 1; a <= n / 3; a++) {
     for (int b = a; n - a - b >= b; b++) {
       count++;
     }
  }
  std::cout << count;
}
```

Данное решение имеет квадратичную сложность, с его помощью можно получить ответы на четыре теста из пяти. Чтобы получить ответ для пятого теста, заметим, что внутренний цикл можно заменить на вычисление суммы арифметической прогрессии − получим алгоритм с линейной сложностью:

```
#include <iostream>

int main() {
  int n;
  std::cin >> n;
  long long count = 0;
  for (int a = 1; a <= n / 3; a++) {
     count += (n - a) / 2 - a + 1;
```

```
  }
  std::cout << count;
}
```

В принципе, этого уже достаточно, чтобы решить задачу на полный балл. Однако, при желании можно избавиться и от этого цикла и получить решение с константной вычислительной сложностью. Приведём пример такого решения (вероятно, его можно реализовать и гораздо короче):

```
#include <iostream>

int main() {
  long long n;
  std::cin >> n;
  long long count = n / 3 - (1 + n / 3) * (n / 3) / 2;

  int s = 1;
  int e = n / 3;

  if (n % 2 == 1 && 1 <= e) {
    count += (n - 1) / 2;
    s = 2;
  }

  if (s <= e && e % 2 == s % 2) {
    count += (n - e) / 2;
    e--;
  }

  if (s < e) {
    long long first = (n - e) / 2;
    long long last = (n - s) / 2;
    count += (first + last) * (last - first + 1);
  }

  std::cout << count;
}
```

Теперь рассмотрим чисто "математический" способ решения этой задачи. Если вручную найти ответы для первых нескольких значений N, то можно заметить, что число различных треугольников с вершинами в вершинах правильного n-угольника равно ближайшему к $N^2/12$ целому числу. Докажем это.

Пусть всего имеется k неравных треугольников с вершинами в вершинах правильного n-угольника, причем из них k_1 правильных, k_2 –

неправильных равнобедренных, k_3 − разносторонних. Каждый правильный треугольник равен одному треугольнику с фиксированной вершиной A (рис. 1), неправильный равнобедренный – трем треугольникам с вершиной A (рис. 2), а разносторонний – шести (рис. 3).

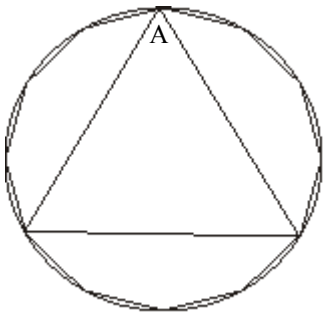


Рис. 1

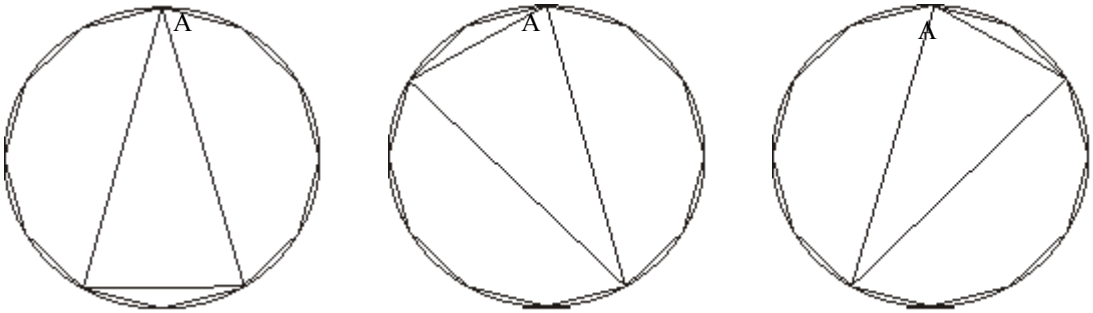


Рис. 2

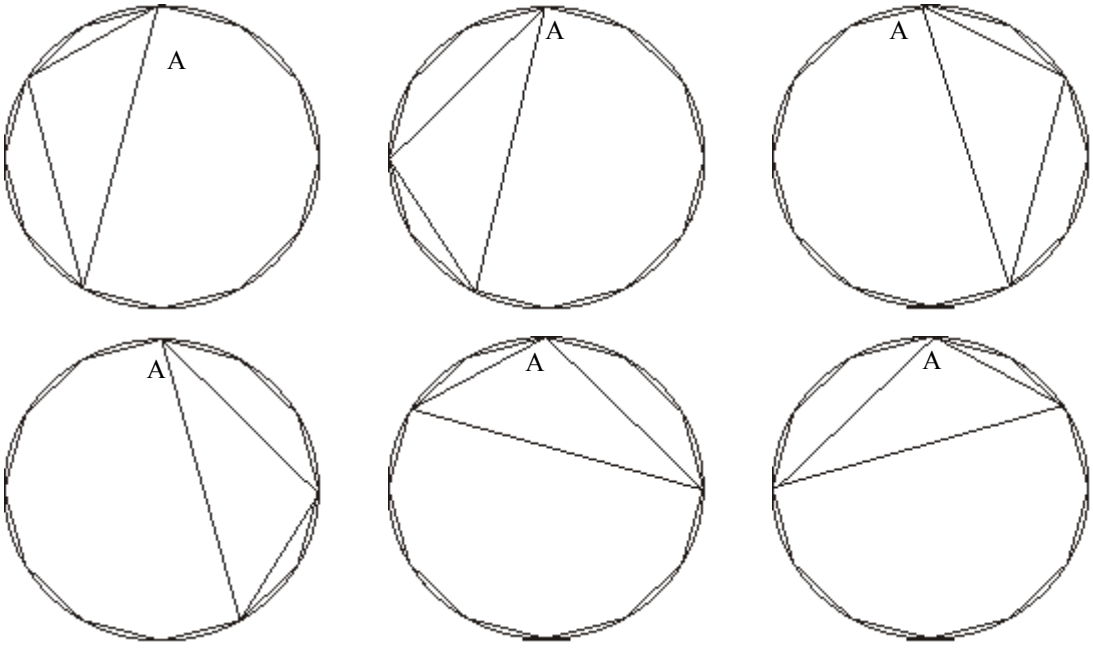


Рис. 3

Так как всего имеется $\frac{(n-1)(n-2)}{2}$ треугольников с вершиной A, то $\frac{(n-1)(n-2)}{2} = k_1 + 3k_2 + 6k_3$. Ясно, что число неравных правильных треугольников равно 0 или 1, а число неравных равнобедренных равно $\frac{n-1}{2}$ или $\frac{n}{2}-1$, т. е. $k_1 = 1 - c$, $k_1 + k_2 = \frac{n-2+d}{2}$ где c и d равны 0 или 1. Поэтому

$12k = 12(k_1 + k_2 + k_3) = 2(k_1 + 3k_2 + 6k_3) + 6(k_1 + k_2) + 4k_1 = (n-1)(n-2) + 3(n-2+ d) + 4(1 - c) = n^2 + 3d - 4c.$

Так как $|3d - 4c| < 6$, то k совпадает с ближайшим к $\frac{n^2}{12}$ целым числом. Ответы на входные числа несложно найти с помощью калькулятора или написав небольшую программу. Для примера приведём текст программы на языке Pascal:

```
var
  n: int64;

begin
  readln(n);
  writeln(round(n * n / 12));
end.
```

Задача 3. Игра (10 баллов)

Рассмотрим два случая: когда N имеет чётное число цифр, и когда нечётное.

Пусть количество цифр $len(N)$ чётно. Заметим, что в результате игры может получиться число либо с таким же числом цифр, что и в числе N, либо на одну цифру больше. В первом варианте победит второй игрок, во втором − первый. Поэтому первый игрок стремится ходить так, чтобы в ответе получилась $len(N)+1$ цифра, а второй − чтобы получилось $len(N)$ цифр. Рассмотрим подробнее стратегии игроков.

Если первая цифра не равна 1, то выиграет первый. Для этого он на первом ходу напишет цифру 1, и далее уже не важно, какие цифры пишут игроки − число на доске станет больше N ровно в тот момент, когда в нём станет $len(N)+1$ цифра.

Если первая цифра числа N равна 1, то первому игроку невыгодно писать цифру больше её (поскольку тогда выиграет второй

игрок из аналогичных соображений). То есть первый игрок напишет цифру 1. Теперь второму игроку невыгодно писать цифру меньше, чем вторая цифра числа *N*. Если вторая цифра числа *N* меньше 9, то второй игрок напишет девятку и гарантированно выиграет. Если же вторая цифра равна 9, то он напишет 9, и задача свелась почти к исходной (только вместо единицы первый игрок теперь будет писать нули).

Таким образом, при чётной длине *N* стратегии игроков будут таковы. Первый игрок на первом ходу пишет единицу. На каждом следующем ходу второй игрок пишет 9, а первый − 0. Получаем число 19090... Критерий выигрыша: если полученное число из *len(N)* цифр больше *N*, то выиграл второй игрок, иначе − первый.

Рассуждая аналогичным образом для нечётной длины *N*, получаем следующую стратегию: первый игрок на каждом ходу пишет цифру 9, а второй − цифру 0.

Пример решения на C++:

```
#include <stdio.h>
#include <cstring>

char s[100001];

void solve() {
  gets(s);
  int n = strlen(s);
  if (n % 2 == 0) {
    if (s[0] != '1') {
      puts("1\n"); return;
    }
    for (int i = 1; i < n; i += 2) {
      if (s[i] < '9') {
        puts("2\n"); return;
      }
      if (s[i + 1] > '0') {
        puts("1\n"); return;
      }
    }
    puts("2\n"); return;
  } else {
    for (int i = 0; i < n; i += 2) {
      if (s[i] < '9') {
        puts("1\n"); return;
      }
      if (s[i + 1] > '0') {
        puts("2\n"); return;
      }
```

```
    }
    puts("1\n"); return;
  }
}

int main() {
  int k;
  scanf("%d\n", &k);
  for (int i = 0; i < k; i++) {
    solve();
  }
}
```

Задача 4. Отчёт (10 баллов)

Создадим массив $p[0..N]$, где элементы $p[1]$, $p[2]$, …, $p[N]$ будут содержать входные данные. Элементу $p[0]$ присвоим значение минус единица − это гарантирует, что для каждого входного числа найдётся число слева меньше его.

Будем идти по массиву слева направо, последовательно получая ответ для каждого элемента. При этом в отдельной структуре данных будем хранить индексы тех элементов, которые, возможно, будут являться ответами для каких-то следующих элементов массива.

Заметим следующий факт. Пусть мы дошли до элемента $p[i]$. Тогда все элементы, которые больше или равны $p[i]$ и стоят левее i, уже никогда не смогут оказаться в ответе. Поэтому, если мы хранили индексы таких элементов, то их теперь можно смело удалить, после чего индекс i запомнить (поскольку он может стать ответом для каких-то последующих элементов).

Заметим, что элементы, индексы которых мы храним, в любой момент времени упорядочены по возрастанию (если рассматривать их в порядке добавления в структуру данных). Действительно, перед сохранением каждого очередного элемента мы вначале удаляем все, которые больше или равны ему. Получается, что в качестве структуры данных удобно использовать стек. На верхушке стека будет лежать индекс самого большого элемента, под ним − индекс элемента меньше его, и так далее.

Итак, алгоритм получается следующий. В стеке хранятся индексы элементов, которые, возможно, когда-то пойдут в ответ. Взяв очередное число $p[i]$, мы будем удалять значения из стека до тех пор, пока очередной элемент не окажется меньше, чем $p[i]$. Тогда этот индекс выводится в ответ, после чего число i заталкивается в стек.

Сложность алгоритма линейная, поскольку каждый элемент будет один раз помещён в стек, и не более чем один раз удалён из стека.

Пример решения на C++

```
#include <stdio.h>
#include <vector>
#include <stack>

int main() {
  int n;
  scanf("%d", &n);
  std::vector<int> p(n + 1);
  p[0] = -1;
  std::stack<int> st;
  st.push(0);
  for (int i = 1; i <= n; i++) {
    scanf("%d", &p[i]);
    while (p[i] <= p[st.top()]) {
      st.pop();
    }
    printf("%d ", st.top());
    st.push(i);
  }
}
```

II Областная олимпиада школьников по информатике
2017-2018 учебный год
9-10 классы
Отборочный тур

Материалы для проверки

Введение

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

Решение задачи 1 должно представлять собой текстовый файл, содержащий одну строку с ответом − текстом формулы. В данную формулу подставляются значения N из приложенного набора тестов, и результаты сравниваются с эталонными. За каждый верный ответ даётся одинаковое количество баллов.

Решение задачи 2 должно представлять собой текстовый файл, содержащий ответы в формате, описанном в условии задачи. За каждый верный ответ даётся одинаковое количество баллов.

Решения задач 3 и 4 должны представлять собой исходный текст программы на одном из допустимых языков программирования. 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'.

Проверка решений участников олимпиады проводится после окончания тура.

Система оценки задач для 9 - 10 классов

Задача	Тестов	Баллов за один тест	Проверяющая программа
1. Украшение	10	1	не обязательна
2. Треугольники	5	2	не обязательна
3. Игра	10	1	не обязательна
4. Отчёт	10	1	не обязательна

Автоматизация процесса проверки решений участников

Существуют различные способы проверки решений участников. Например, можно выполнять проверку вручную. Однако, такой способ трудоёмок и чреват случайными ошибками. Поэтому рекомендуется использовать специальные средства для автоматизации проверки решений:

1. **Рекомендуемый вариант** − автоматическая проверяющая система, доступная по адресу http://atpp.vstu.edu.ru/acm. Инструкция по работе с ней приведена далее.
2. Использование специальных программ для проверки решений на наборе тестов — например, Tester (http://acm.timus.ru/tester). Инструкция по работе с данной программой приведена в конце документа.
3. Самостоятельное развёртывание полноценной проверяющей системы - например, Ejudge. Сравнительно трудоёмко и требует навыков работы с ОС Linux.

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

1). Откройте сайт http://atpp.vstu.edu.ru/acm

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, **obl_vologda**
Отображаемое имя	Например, **Областная олимпиада, 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	Можно оставить по умолчанию
Кодовое слово	**`p2q38u1`** Все буквы в слове маленькие. Кодовое слово позволит вам получить доступ к проверке задач олимпиады. Не сообщайте его посторонним.

3). Нажмите на кнопку «Задачи». Слева будут показаны темы (разделы). Раскройте тему «**II Областная олимпиада школьников по информатике**». Если всё получилось, то переходите к пункту 4.

Если у вас не отображается тема «**II Областная олимпиада школьников по информатике**», 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). Рекомендуем потренироваться в работе с системой на пробных задачах. В подразделе «Пробные задачи» имеется несложная задача с названием «Точные

квадраты». Откройте её условие (щелкнув по названию). Попробуйте написать и послать решение на каком-нибудь языке программирования. Для отправки решения выполните 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 отборочного тура здесь может писаться фамилия (либо шифр) участника, чьё решение проверяется.
- нажмите кнопку «**Отправить**». Пример показан на следующем рисунке.

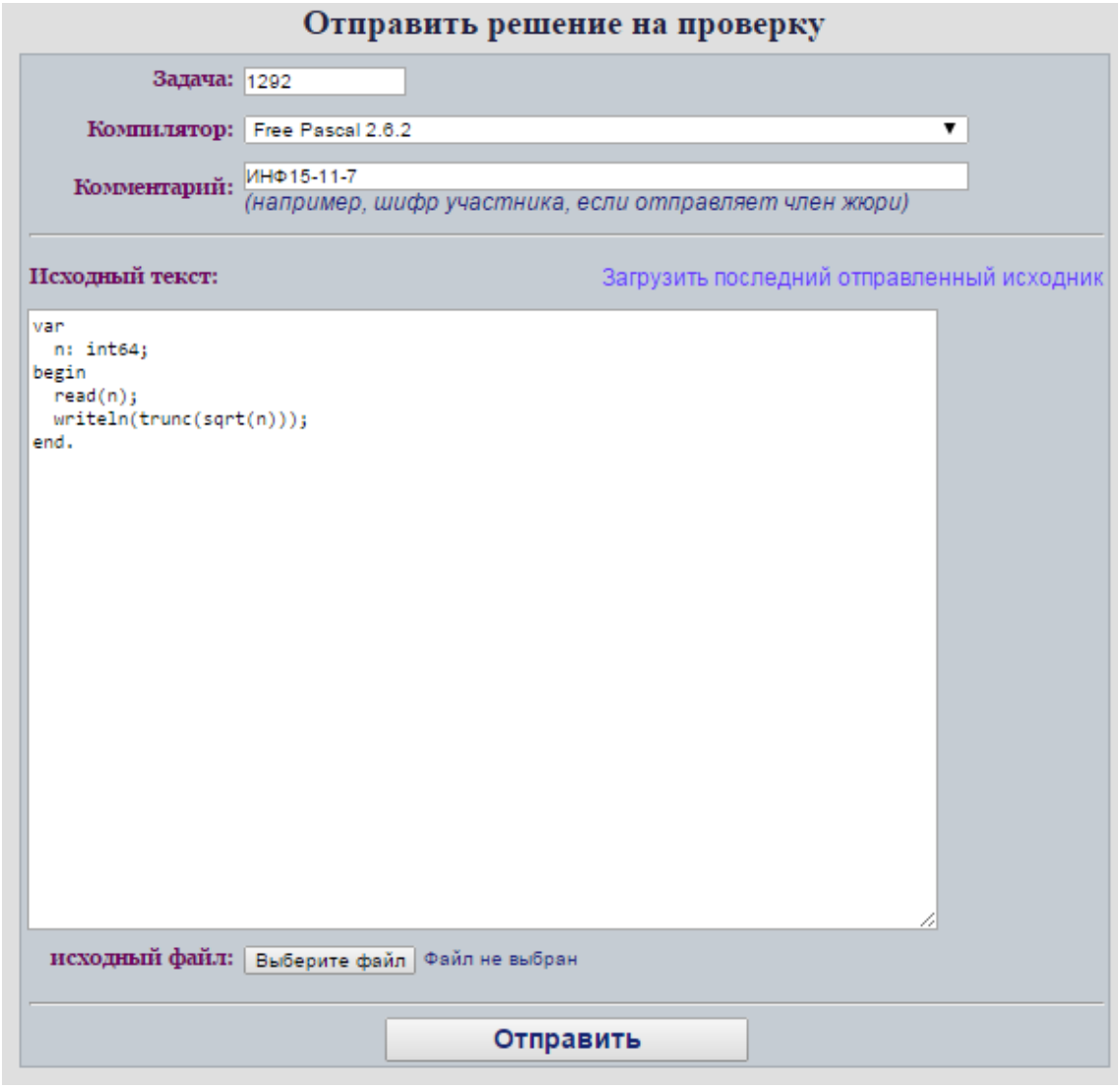


Рис.1. Отправка решения на проверку

Откроется окно «**Online статус**». В выпадающем списке «**Показать решения**» убедитесь. что выбран вариант «**Свои**». Если нет, то выберите его 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	Решение полностью неверно
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
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 отборочного тура 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)

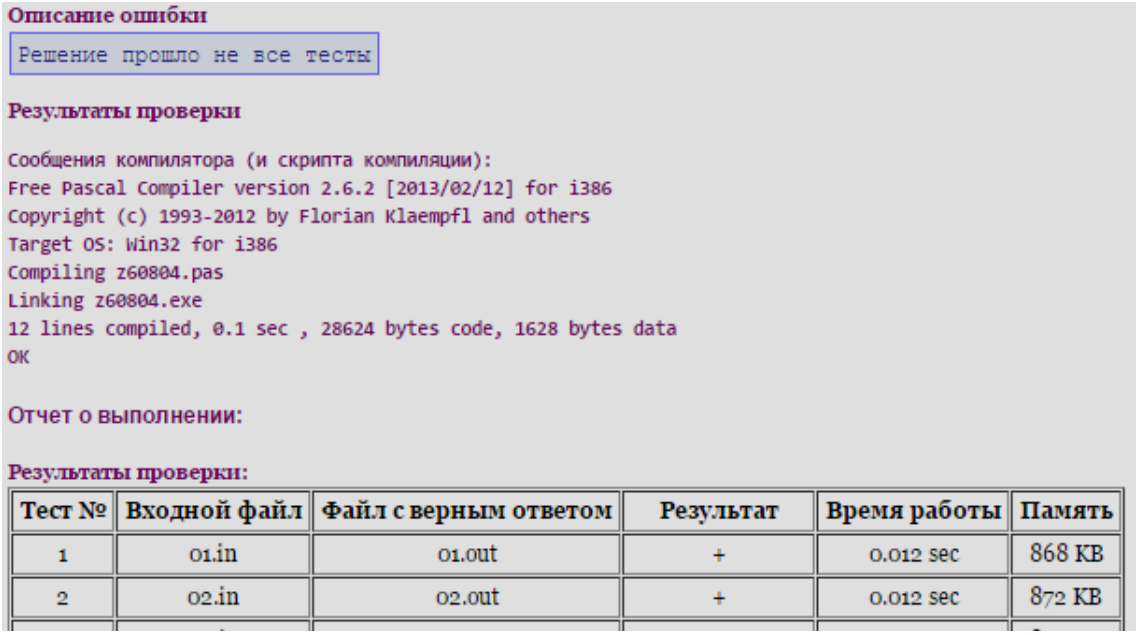
Обнаружено бездействие	Программа не использует процессорное время, но и не завершается (например, ожидает нажатия клавиши - попробуйте убрать readln в конце программы или т.п.) ***Данный результат возможен в связи с кратковременной перегрузкой сервера - попробуйте послать задачу повторно.***

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

Рис. 2. 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. 3. 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). Попробуйте сдать решение второй пробной задачи с названием «A + B, версия 2». Для этой задачи в систему посылается уже не программа, а текст с готовым ответом.

6). Рекомендации по проверке решений

Проверка решений выполняется жюри после окончания тура.

Для проверки решений зайдите в тему «II Областная олимпиада школьников по информатике» и далее в подраздел «Отборочный тур, 9-10 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 фамилию или 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 проверки через web-сайт (например, нет доступа к Интернет), то в качестве альтернативы можно воспользоваться свободно распространяемой программой "Тестер" (http://acm.timus.ru/tester). Данная программа имеется в архиве материалов олимпиады. Для удобства запуск этой программы с нужными параметрами прописан в файлах test.bat, которые лежат в папках для каждой задачи.

Рассмотрим порядок действий для проверки решения. Пусть участник сдал на проверку решение, которое называется program.pas. Скомпилируем решение, чтобы получился файл program.exe. Скопируйте файл program.exe в каталог с материалами по проверяемой задаче. В командной строке перейдите в каталог с материалами по задаче и выполните команду "`test.bat program.exe`". На экран выведется протокол проверки:

```
C:\>y:

Y:\>cd squares

Y:\squares>test.bat 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: 10
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).

Примечание 3. Иногда (редко) встречается ситуация, когда программа участника, запущенная из-под "Тестера", выполняется существенно дольше, чем если бы её запускали отдельно. Это может привести к ложному вердикту "Предел времени". Чтобы этого избежать, можно сделать следующее. Пусть проверяемая программа называется program.exe. В системном реестре Windows нужно создать подраздел с названием program.exe в разделе HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\, это легко можно сделать из командной строки (командная строка должна быть запущена под правами администратора):

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\program.exe"
```

Рассмотрим кратко особенности проверки решений на разных языках. В случае, если нужно проверить решение на Java, 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.bat "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.bat C:\Python33\python.exe program.py
```

Программу "Тестер" можно применить и для проверки задач 1 и 2, хотя решениями в них являются не программы, а текстовые файлы. Пусть answer.txt - файл с ответом участника. Тогда команда будет выглядеть так:

```
test.bat answer.txt
```