Как правильно писать программы для автоматической проверки на олимпиадах по информатике

Версия от 2 декабря 2012 г.

Данная инструкция разработана для окружной олимпиады в Москве в 2012 году, но может быть использована и на других олимпиадах.

Оглавление

На большинстве олимпиад по информатике для проверки задач используется автоматическая тестирующая система. Это означает, что проверяемая программа должна быть написана специальным образом для того, чтобы программа была проверена на олимпиаде.

Как выглядит олимпиадная задача по информатике

Рассмотрим задачу «Конец K-го урока» с окружной олимпиады в г. Москве в 2009 году. Ниже записано условие задачи.

В школе продолжительность каждого урока 45 минут, а перемены между уроками – всего 5 минут. Первый урок начинается ровно в 8 часов утра. Напишите программу, отвечающую на вопрос «во сколько в этой школе заканчивается *K*-ый урок?»

Входные данные. Вводится одно натуральное число *K*, не превышающее 15.

Выходные данные. Выведите время окончания *K*-ого урока: сначала часы, потом минуты, разделяя их пробелом.

Пример ввода	Пример вывода
1	8 45
6	12 55

Условие олимпиадной задачи состоит из нескольких частей. Сначала написана «легенда» — условие задачи. Легенда может быть довольно длинной и на первый взгляд иметь довольно отдаленное отношение к задаче. Это — довольно важная часть задачи по информатике — умение по легенде составить модель задачи и придумать алгоритм ее решения. На серьезных олимпиадах перед легендой идут ограничения, в которые должно укладываться решение задачи — ограничение по времени работы программы на одном тесте (как правило — 1-2 секунды) и ограничение по используемой памяти (как правило — 64-256 мегабайт). Также если предусматривается файловый ввод-вывод, то в начале условия записываются имена файлов.

Легенда отвечает на самый важный вопрос — «Что именно должна делать программа, которая является решением задачи?».

Далее идет часть, в котором описываются входные данные к задаче. В этом примере входными данными является одно целое число. В описании формата входных данных указываются **ограничения** на эти данные. В нашем случае сказано, что входное число является натуральным, то есть целым и положительным, и оно не превосходит 15. Ограничения на входные данные даются для того, чтобы участники олимпиады понимали, какие типы данных использовать для хранения чисел (например, если входное число может быть до 10^6, то для его представления понадобится 32-битный целочисленный тип, например, тип integer для языка Паскаль не подойдет).

В следующей части описываются требования к тому, что должна вывести программа. В нашем случае программа выводит два числа — час и минуту, в которые заканчивается K-й урок.

Проверка программ на олимпиадах по информатике проводится с использованием автоматической тестирующей системы. В условиях этой задачи, например, программа должна давать правильный ответ для K=1, 2, …., 15. То есть если K=1, то ответом будет 8 часов 45 минут, если K=2 — 9 часов 35 минут, если K=3 — 10 часов 25 минут и т. д. При этом вывод программы должен соответствовать в точности описанию формата выходных данных, то есть программа должна вывести только два целых числа. Для облегчения понимания, что должна выводить программа, в конце условия даются примеры того, что должна выводить программа при одном или нескольких вариантах входных данных.

Проверка заключается в том, что программа несколько раз запускается тестирующей системой и ей на вход подается заранее подготовленный жюри набор входных данных. Программа должна считать эти данные, решить задачу и вывести ответ. Ответ сверяется с правильным ответом, если ответ совпадает с правильным (или с одним из нескольких возможных правильных ответов, как это бывает в некоторых задачах), то тест засчитывается, как успешно пройденный. Если программа проходит все тесты (на всех тестах выдает правильный ответ), то задача считается верно решенной и получает полный балл, если не на всех те-

стах — то задача считается частично решенной и получает частичный балл в зависимости от количества пройденных тестов. Все это происходит в автоматическом режиме без участия человека, члены жюри только наблюдают за процессом проверки.

Как выглядит решение олимпиадной задачи

Решением задачи является программа на одном из допустимых языков программирования, которые поддерживаются тестирующей системой. Тестирующая система подает программе данные на стандартный ввод и ожидает от программы вывода результата на стандартном выводе. В «обычных» условиях запуска программы, когда она пишется в среде разработки и запускается на компьютере участника олимпиады, стандартный ввод соответствует вводу данных с клавиатуры а стандартный вывод — выводу на экран. То есть программа должна для считывания данных использовать стандартные функции ввода-вывода языка программирования, например, read и write в Паскале, input и print в бейсике, scanf и printf в C, cin и cout в C++, input и print в Питоне и т. д. В этом случае в тестирующей системе программа будет запущена с перенаправленным вводом-выводом для возможности автоматической проверки.

Решение не должно использовать никакие графические функции языка программирования и операционной системы, то есть нельзя осуществлять ввод-вывод через формы, окна диалогов и т. д., поскольку проверка происходит без участия человека, который будет вводить входные данные в формы. То есть создаваемое приложение должно быть простым «консольным» приложением, а не графическим приложением, использующим окна.

По этим же причинам не могут быть проверены программы, предполагающие взаимодействие через интернет или с использованием браузера. Например, программа на языке PHP предполагающая, что она будет запущена на web-сервере и будет получать данные через GET или POST-методы протокола http, не может быть проверена в автоматической системе. Аналогично не может быть проверена программа на JavaScript, для исполнения которой нужен web-браузер.

Прежде всего программа должна считать входные данные. При этом если программа будет выводить какие-то дополнительные сообщения, например, «Введите количество уроков», то данный текст также будет выведен программой и обработан тестирующей системой, то есть программа выведет не только два числа в ответе, но и дополнительное сообщение и не сможет быть проверена тестирующей системой, то есть не надо писать в программе команды вида Writeln('Введите количество уроков').

Не надо проверять входные данные на корректность — ограничения в условиях задачи означают, что во всех тестах, на которых будет проверяться ваша программа, будут выполнены данные ограничения, то есть не надо писать команды вроде if K > 15 then writeln('Введите правильное число уроков') — это не является ошибкой, но просто бессмысленно.

При выводе программы не надо выводить ничего лишнего, если требуется вывести два числа, то нужно вывести только два числа (разделив их при этом пробелом или переходом на новую строку в зависимости от условий задачи). Не нужно выводить никаких дополнительных сообщений типа «Ответ», не нужно выводить слова «часы» или «минуты» — весь этот вывод также не будет проверен автоматической тестирующей системой.

В конце программы часто ставят дополнительную задержку — например, ожидают ввода чего-нибудь при помощи функции Readln в Паскале или вызывают функцию system("pause") в языке C — это также не требуется, т. к. программа должна сразу же завершить свою работу, не дожидаясь какого-либо действия от человека, т. к. в автоматической системе этого не нужно.

Еще одним возможным способом ввода-вывода является чтение данных из файла с

именем input.txt, находящимся в одном каталоге с решением задачи и вывод результата работы в файл с именем output.txt.

Резюмируя все записанное выше, программа для проверки в автоматической тестирующей системе должна выглядеть примерно так:

Программа на языке Паскаль

```
var k, time: integer;
begin
    read(k);
    time := 45 * k + 5 * (k — 1);
    writeln(8 + time div 60, ' ', time mod 60)
end.
```

Программа на языке Питон версии 3

```
k = int(input())
time = 45 * k + 5 * (k — 1);
print(8 + time // 60, time % 60)
```

Программа на языке Си

```
#include<stdio.h>
int main()
{
    int k, time;
    scanf("%d", &k);
    time = 45 * k + 5 * (k — 1);
    printf("%d %d", 8 + time / 60, time % 60);
    return 0;
}
```

Программа на языке C++

```
#include<iostream>
using namespace std;
int main()
{
    int k, time;
    cin >> k;
    time = 45 * k + 5 * (k — 1);
    cout << 8 + time / 60 << " " << time % 60;
    return 0;
}
```

Программа на языке Free Basic (аналог qbasic)

```
DIM k, time AS INTEGER
INPUT k
time = 45 * k + 5 * (k — 1)
print 8 + time \ 60, time MOD 60
```

Программа на языке Mono Visual Basic

```
Module ProgramA
Sub Main()
DIM k, time AS INTEGER
k = CInt(Console.Readline())
time = 45 * k + 5 * (k - 1)
Console.WriteLine(CStr(8 + time \ 60))
Console.WriteLine(CStr(time MOD 60))
End Sub
End Module
```

Программа на языке Кумир

```
алг Задача А
нач
    цел k, time
    ввод k
    time := 45 * k + 5 * (k - 1)
    вывод 8 + div(time, 60), ' ', mod(time, 60)
кон
```

Программа на языке Perl

```
my $n = <>;
my $m = <>;
my $k = <>;
my $p = int($k / $n);
my $a = int( ($m + $p - 1) / $p);
print $a, "\n";
```

Программа на языке Ruby

```
n = gets.to_i
m = gets.to_i
k = gets.to_i
p = k / n
a = (m + p - 1) / p
print a, "\n"
```

Программа на языке PHP

```
<?php
$n = fgets(STDIN);
$m = fgets(STDIN);
$k = fgets(STDIN);
$p = floor ($k / $n);
$a = floor (($m + $p - 1) / $p);
print $a . "\n";
?>
```

Программа на языке C#

```
using System;
using System.IO;
```

```
class Program
{
    static void Main()
    {
        int n = int.Parse(Console.ReadLine());
        int m = int.Parse(Console.ReadLine());
        int k = int.Parse(Console.ReadLine());
        int p = k / n;
        int a = (m + p - 1) / p;
        Console.WriteLine("{0}", a);
    }
}
```

Программа на языке Java

```
import java.io.*;

public class Main
{
    public static void main(String[] args) throws Exception
    {
        DataInputStream in = new DataInputStream(System.in);
        int n, m, k, p, a;
        n = Integer.parseInt(in.readLine());
        m = Integer.parseInt(in.readLine());
        k = Integer.parseInt(in.readLine());
        p = k / n;
        a = (m + p - 1) / p;
        System.out.println(a);
    }
}
```

Типичные ошибки, связанные с использованием тестирующей системы

В большинстве случаев ошибки связанные с тем, что программа не может быть сдана в тестирующую систему, имеют одну из следующих причин:

1. Программа не является консольным приложением, либо используются какие-то нестандартные возможности компилятора, привязанные к конкретной операционной системе, например, использование функции ClrScr в Паскале, прекомилированных заголовочных файлов в Visual C++ и т. д.
2. Программа выводит дополнительные сообщения, «улучшающие» интерфейс пользователя, например, «Введите количество уроков».
3. Программа выводит дополнительные сообщения, не предусмотренные форматом выходных данных, например, «часы», «минуты», «ответ».
4. Программа содержит «задержку» после окончания работы, то есть ждет от пользователя нажатия на какую-либо клавишу или иных действий.

Ниже указаны особенности конкретных языков программирования и приведены примеры правильных программ на данных языках программирования.

Особенности использования языка Паскаль

Язык программирования Паскаль представлен в тестирующей системе тремя компиляторами: Free Pascal, Borland Delphi, PascalABC.Net. Компилятор Turbo Pascal тестирующей системой не поддерживается, вместо Turbo Pascal рекомендуется использовать Free Pascal.

В программах на языке Паскаль запрещается использовать модуль crt (даже подключение этого модуля делает невозможным проверку программы).

Одной из типичных ошибок ввода-вывода при использовании языка Паскаль является неправильное использование функции ReadLn, которая после считывания данных считывает конец строки (и пропускает все данные, находящиеся после считанных данных и до конца строки). Как правило это возникает в случае, когда программа получает на вход два или более чисел, записанных в одной строке через пробел. В этом случае нельзя читать данные при помощи ReadLn(a); ReadLn(b), так как такое использование функций ввода означает, что после первого числа должен быть конец строки, и следующее число записано в новой строке. Правильным чтением данных будет ReadLn(a, b) или Read(a); Read(b).

Особенности компилятора Free Pascal

При этом нужно учитывать то, что компилятор Free Pascal по умолчанию работает в 16-битном режиме, в частности, размер переменной integer составляет 2 байта (16 бит) и может принимать значения от -32768 до 32767. Для хранения 32-битных целых чисел во Free Pascal следует использовать тип longint. Длина текстовой строки во Free Pascal ограничена 255 символами.

Данные параметры можно менять внутри программы при помощи директив компилятора, описанных в документации на язык Free Pascal.

В программах на Free Pascal нельзя использовать модуль `crt` и функции из этого модуля, например, `ClrScr`.

Особенности компилятора Borland Delphi

При создании программы в Borland Delphi (меню File — New — Other) необходимо выбрать тип программы «Console application».

При использовании модулей их названия должны быть записаны точно так же, как это делает среда Delphi. Правильная запись (обратите внимание на заглавные буквы):

```
uses SysUtils, Math;
```

соответственно, названия sysutils, Sysutils, math — неправильные. Модуль Windows использовать нельзя.

Особенности компилятора PascalABC.NET

В программах на PascalABC.NET нельзя использовать модуль `crt` и функции из этого модуля, например, `ClrScr`.

Особенности использования языков Си и C++

В тестирующей системе доступен только компилятор GNU C/C++.

Для разработки программ можно использовать среду Visual C++, но проверять решения нужно под компилятором GNU C/C++. При создании проекта в Visual Studio (меню File — New — Project) необходимо выбрать «Win32 Console Application». При создании проекта

необходимо отключить использование прекомпилированных заголовочных файлов (в окне диалога «Application Settings» убрать галочку «Additional options: Precompiled headers»). В готовом файле с программой не должно быть строки

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

(наличие этой строки означает, что не были отключены прекомпилированные заголовочные файлы). Также описание функции main необходимо изменить на

```
int main()
```

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

Программа на языках C или C++ должна заканчиваться с кодом возврата 0 (`return 0`), ненулевой код возврата может быть воспринят тестирующей системой, как ошибка в работе программы. Функция `main` должна возвращать значение типа `int` (а не `void`).

В программе необходимо явно подключать заголовочные файлы, т. к. компилятор в тестирующей системе не подключает ни одного заголовочного файла с функциями стандартной библиотеки. То есть в программах на языке C должно быть написано

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

В программах на языке C++ должно быть написано

```
#include <iostream>
```

Компиляторы Turbo C/C++, Borland C/C++ не поддерживаются, можно использовать компилятор GNU C/C++, при этом программа на C++ должна соответствовать современному стандарту языка C++, в частности, программа должна начинаться так:

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

Особенности использования языка Бейсик

Региональная методическая комиссия всероссийской олимпиады школьников по информатике в г. Москве не рекомендует использовать язык программирования Бейсик на олимпиадах и для обучения программированию!

Компилятор Qbasic не поддерживается в тестирующей системе, вместо него можно использовать компилятор Free BASIC (http://www.freebasic.net/), который будет запускаться в режиме совместимости с Qbasic (с ключом компиляции -lang qb). В среде разработки QuickBasic 4.5 при сохранении файла обязательно нужно выбрать формат «Text — Readable by Other Programs».

Компилятор Visual Basic будет доступен в варианте Mono Visual Basic, при этом для работы программы необходимо создавать консольное приложение, а не графическое приложение. Программа должна читать данные со стандартного ввода, выводить результат на стандартный вывод.

Возможные результаты тестирования

После сдачи задачи в тестирующую систему программа компилируется в исполняемый машинный код. Если программа содержит синтаксические ошибки и не может быть скомпилирована, то в столбце «Результат» в тестирующей системе будет написано «Ошибка компиляции», при этом в столбце «Отчет о проверке» будет ссылка на вывод компилятора, исходя из которого можно узнать, какие были ошибки при компиляции программы. Для интерпретируе-

мых языков программирования компиляция не производится.

Если программа была успешно скомпилирована, то она запускается на тестах, при этом в столбце «Отчет о проверке» будет ссылка на протокол тестирования программы на всех тестах. По каждом из тестов возможен один из следующих результатов тестирования:

OK

Тест пройден, программа выдала правильный ответ на этом тесте.

Неправильный ответ

Программа выдала неправильный ответ на данном тесте, это означает, что программа содержит ошибку.

Неправильный формат вывода

То, что вывела программа, не соответствует описанию формата выходных данных, приведенному в условию задачи. Возможные причины для этого такие:

1. Программа не вывела ничего (это может быть, например, при наличии ошибки в программе или при ошибке в написании имени выходного файла, если используется файловый ввод-вывод).
2. Программа должна вывести два числа, а вывела одно число, или три числа, или текст и т. д.
3. Программа выводит лишние сообщения типа «Введите число» или «Ответ».

Просмотрите отчет о проверке по каждому тесту чтобы понять, что именно вывела программа.

Ошибка выполнения

Программа совершила некорректную операцию во время тестирования. Возможные причины для этого:

1. Некорректное арифметическое действие или математическая операция, например, деление на ноль, извлечение корня из отрицательного числа, переполнение переменных.
2. Ошибки при работе с памятью — выход за границы массива, обращение к невалидным указателям в языке Си или C++, переполнение стека, выделение слишком большого объема динамической памяти.
3. Ошибка в написании имени входного файла при использовании файлового ввода-вывода.
4. В программе на языке Питон и иных интерпретируемых языках — синтаксическая ошибка в программе или любая иная ошибка при исполнении программы.
5. Бесконечная (или очень большая) рекурсия.
6. В программе на языке Си или C++ явно указан ненулевой код завершения программы.

Превышено максимальное время работы

Программа не закончила свою работу за время, отведенное на исполнение одного теста. Возможные причины для этого:

1. Алгоритмическая ошибка — программа попадает в «бесконечный цикл».

2. Неэффективное решение — программа работает слишком долго.
3. Ошибка в считывании данных, например, программа считывает со стандартного ввода два числа, между тем как в описании формата входных данных указано только одно число (тогда программа будет бесконечно долго ожидать ввода второго числа).
4. Программа явно ожидает от пользователя ввода чего-либо, нажатия на клавишу после решения задачи (для организации «задержки» программы).

Превышен лимит по памяти

Программа использовала больше оперативной памяти, чем это предусмотрено ограничениями в задаче. Возможные причины для этого:

1. Используются слишком большие массивы (или слишком много массивов) или иные структуры данных.
2. Бесконечная (или очень большая) рекурсия.
3. Некорректная работа с указателями в Си или C++ также может диагностироваться, как «Превышен лимит по памяти».

Сдача задач на проверку во время тура

Во время участники олимпиады имеют возможность сдавать задачу на проверку в тестирующую систему. Для этого необходимо войти в тестирующую систему, указав свой логин и пароль. Интерфейс тестирующей системы в верхней части экрана содержит следующие ссылки:

«Инфо» - просмотр информации по задачам и сдача задач.

«Итог» - итоговый результат по всем задачам, где показано, по каким задачам были приняты решения на проверку, а после окончания олимпиады отображаются набранные баллы по всем задачам.

«Посылки» - список всех отправленных на проверку решений по всем задачам.

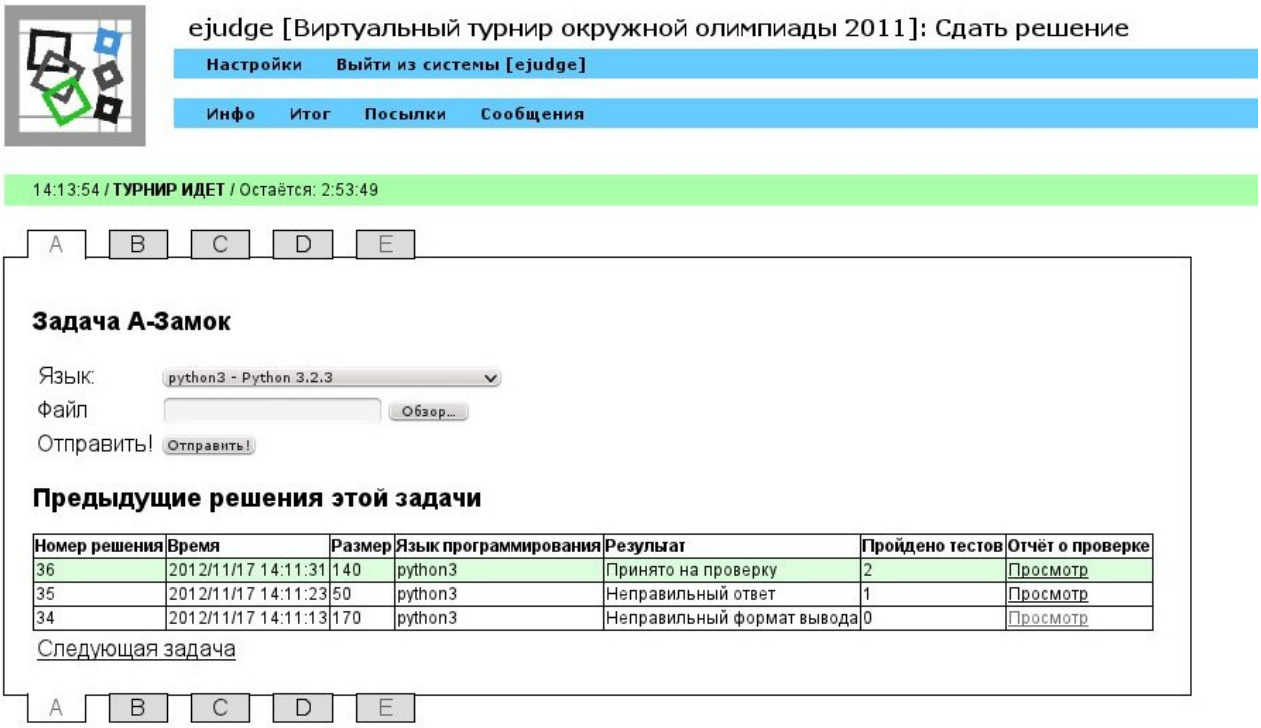
«Отправить вопрос» - страница для отправки вопроса по условиям задачи или для иных сообщений судьям.

«Сообщения» - страница, на которой можно увидеть отправленные сообщения и полученные ответы на них.

Ниже находится набор вкладок для выбора задач. Как правило, задачи обозначаются буквами латинского алфавита: A, B, C, D, E.

Сдача задач на проверку

Для сдачи задачи необходимо выбрать вкладку с названием задачи («A», «B» и т. д.). На этой странице необходимо выбрать язык программирования (из списка возможных языков программирования) и файл с исходным кодом программы (нажав на кнопку «Обзор» и выбрав файл в диалоговом окне выбора файла), затем нажать на кнопку «Отправить».



Выбранный файл отправляется на проверку в тестирующий сервер.

После окончания проверки сданное решение появляется в списке «Предыдущие решения этой задачи» в нижней части страницы.

На приведенном выше скриншоте видно, что по задаче «A» было сдано три решения. Решение номер 34 прошло не прошло ни одного теста из условия, решение номер 35 прошло 1 тест из условия, решение номер 36 прошло два теста из условия. Всего в условиях задачи два теста, поэтому решение 36 принимается на проверку, а решения номер 34 и 35 не были приняты на проверку.

Список «Предыдущие решения этой задачи» содержит следующие столбцы:

«Номер решения» - уникальный номер, присваиваемый каждому сданному решению, позволяющий идентифицировать каждый сданный на проверку файл.

«Время» - время сдачи решения по часам тестирующего сервера.

«Размер» - размер сданного файла в байтах.

«Задача» - краткое название задачи.

«Язык программирования» - краткое название языка программирования для этой задачи.

«Результат» - результат проверки сданного решения на тестах из условия. Возможные результаты - «Принято на проверку», «Ошибка компиляции», «Неправильный ответ», «Неправильный формат выходных данных», «Ошибка выполнения», «Превышено максимальное время работы» и т.д.

«Пройдено тестов» - количество успешно пройденных тестов.

«Отчёт о проверке» - ссылка на страницу с текстом сообщений об ошибках компиляции или полного протокола тестирования.

После сдачи решения на проверку необходимо изучить информацию о сданном решении в списке последних сданных решений. Прежде всего необходимо обратить внимание на столбец «Результат». Если в столбце «Результат» написано «Принято на проверку», то это

означает, что решение прошло тесты из условия и будет оценено после окончания олимпиады (если таких решений несколько, то будет оценено последнее принятое решение по каждой задаче). Это не означает, что задача решена правильно — возможно, решение содержит ошибки или вообще неверно, но на тестах из условия задачи выдает правильный ответ.

Если в столбце «Результат» написано «Ошибка компиляции», то это означает, что решение содержит синтаксические ошибки и не было скомпилировано тестирующей системой. Возможные причины для этого:

1. Синтаксическая ошибка в программе.
2. Неверно выбран язык программирования при сдаче задачи.
3. Неверно выбран файл при сдаче задачи.
4. Различия в используемых версиях компилятора в тестирующей системе и участником олимпиады.

Во всех случаях необходимо нажать на ссылку «Просмотр» в столбце «Отчет о проверке» и изучить сообщения об ошибках компилятора, устранить причины ошибки и сдать еще раз.

Если в столбце «Результат» написано «Неправильный ответ», «Неправильный формат выходных данных», «Превышено максимальное время работы», «Ошибка выполнения» или любой другой статус, это означает, что программа не проходит все тесты из условия, например, по причине выдачи неправильного ответа на этих тестах, или неправильного оформления ввода-вывода (программа не соответствует требованиям, предъявляемых к решениям задач) и т.д. В этом случае также необходимо изучить отчет о тестировании, в котором содержится детальная информация о тестах, на которых проверялась программа.

Просмотр отчета о проверке

Если задача не была принята на проверку, необходимо просмотреть отчет о проверке, нажав на ссылку «Просмотр» в столбце «Отчет о проверке.

Если отправленное решение имеет результат «Ошибка компиляции», то отчет о проверке содержит вывод компилятора. Необходимо изучить вывод компилятора и устранить необходимые ошибки.

Во всех остальных случаях отчет о проверке содержит следующую информацию. Вверху идет таблица в которой для каждого теста указан номер теста, результат работы программы на этот тесте («OK», «Неправильный ответ», «Превышено максимальное время работы» и т. д.). Ниже этой таблицы приведен отчет от тестировании на тестах — сначала на тесте 1, затем на тесте 2 и т. д.

Для каждого теста указана следующая информация: содержание теста (что программа получает на вход), результат работы программы (что программа вывела на этом тесте), правильный ответ, вывод программы в стандартный поток сообщений об ошибках, вывод проверяющей программы. Например, на скриншоте ниже приведен отчет о проверки решения на двух тестах. На первом тесте программа выводит правильный ответ, а на втором тест — неправильный (программа вывела число — 2, а правильный ответ — 5).

14:29:34 / **ТУРНИР ИДЕТ** / Остаётся: 2:38:09

Неправильный ответ

Ошибка на тесте 2.

N	Результат	Время (с)	Доп. информация	Ссылка
1	OK	0.076	OK	L I O A E C F
2	Неправильный ответ	0.076	Answers do not match: out = 2, corr = 5	L I O A E C F

L	Параметры командной строки
I	Входные данные
O	Вывод программы
A	Правильный ответ
E	Вывод программы на stderr
C	Вывод проверяющей программы
F	Доп. инф. о тесте

```
====== Тест #1 ======
--- Входные данные ---
2
5

--- Результат работы программы ---
2

--- Правильный ответ ---
2

--- Ошибки ---

--- Вывод проверяющей программы ---
OK

====== Тест #2 ======
--- Входные данные ---
4
15

--- Результат работы программы ---
2

--- Правильный ответ ---
5

--- Ошибки ---

--- Вывод проверяющей программы ---
Answers do not match: out = 2, corr = 5
```

Просмотр итога по всем задачам

Ссылка «Итог» отображает таблицу с результатом сдачи решений по всем задачам. Если задача была принята на проверку, то в столбце «Статус» написано «Принято на проверку», а строка таблицы покрашена в зеленый цвет.

Во всех остальных фон строки для данной задачи — белый, а в столбце «Статус» написана причина, по которой задача не была принята на проверку.

Рекомендуется перед окончанием тура обязательно проверить, что в таблице «Итог» написано «Принято на проверку» по всем задачам, которые сдавались в тестирующую систему.

По ссылке «Посылки» можно просмотреть список всех сданных решений по всем задачам.

Окончание олимпиады

После окончания олимпиады будет осуществлена окончательная проверка решений на всех тестах. Прохождение каждого теста оценивается определенным числом баллов (как правило, 1 или 2), при этом общее число баллов за задачу равно 10. Если задача проходит все тесты, то она набирает максимальный балл (10), а если часть тестов — то программа набирает неполный балл, в зависимости от количества пройденных тестов. Тесты из условия оцениваются в 0 баллов, поэтому решение, принятое на проверку, может набрать и 0 баллов, если не пройдет ни одного теста, кроме теста из условия.

После окончания олимпиады по ссылке «Итог» отображается количество набранных баллов по всем задачам и сумма набранных баллов.

Задачи, прошедшие все тесты, получают статус «OK» и максимальное количество баллов по задаче. Задачи, прошедшие часть тестов, получают статус «Неполное решение» и частичный балл по задаче. В столбце «Номер решения» при этом указан уникальный идентификатор решения по данной задаче, которое было оценено.

По ссылке «Посылки» можно просмотреть список всех отправленных решений во время тура. Можно скачать исходный код каждого сданного решения (по ссылке «Просмотр»

в столбце «Сданный ответ») и отчет о проверке по каждому решению.

14:53:17 / **ЗАКОНЧЕН**

Отправленные решения (последние 15)

Номер решения	Время	Размер	Задача	Язык программирования	Результат	Пройдено тестов	Баллы	Сданный ответ	Отчёт о проверке
41	2012/11/18 14:20:21	1056	E	fpc	Неправильный формат вывода	0		Просмотр	Просмотр
40	2012/11/18 14:20:07	1056	D	fpc	Неполное решение	4	6	Просмотр	Просмотр
39	2012/11/18 14:17:39	405	D	fpc	Ошибка выполнения	0		Просмотр	Просмотр
38	2012/11/18 14:16:28	405	C	fpc	OK	6	10	Просмотр	Просмотр
37	2012/11/18 14:16:21	84	A	fpc	Неполное решение	6	8	Просмотр	Просмотр

Посмотреть всё