

Инструкция по использованию проверяющей системы на олимпиаде по информатике

Олимпиада проводится в компьютерной форме с использованием централизованной автоматической тестирующей системы. Учащимся 7–8 классов будет предложено пять заданий: три теоретического характера и две задачи по программированию. Учащимся 9–11 классов будет предложено четыре задачи по программированию.

«Почти как в ЕГЭ». Пример теоретического задания¹.

Условие

Автомат получает на вход четырёхзначное число (его первая цифра не должна быть равна нулю). По этому числу строится новое число по следующим правилам:

1. складываются первая и вторая, вторая и третья, третья и четвёртая цифры числа;
2. полученные числа записываются без разделителей в порядке убывания.

Пример. Исходное число: 3165. Суммы: $3+1=4$, $1+6=7$, $6+5=11$. Числа 4, 7 и 11 записываются друг за другом в порядке убывания, получается 1174.

Вам даны пять чисел, которые получил автомат при вводе в него каких-то пяти неизвестных четырёхзначных чисел: 321, 860, 1276, 13123, 171615.

Для каждого из этих чисел найдите **минимальное** целое число, при подаче которого на вход автомату было получено данное число (то есть нужно найти пять чисел, таких, что из первого числа автомат получит число 321, из второго числа – 860, из третьего числа – 1276, из четвёртого числа – 13123, из пятого числа – 171615, причём найденные числа должны быть минимально возможными).

В ответ нужно записать пять целых чисел. Каждое число должно быть в отдельной строке. Порядок записи менять нельзя. Если вы не можете найти ответ для какого-то числа, то вместо ответа запишите знак «-» (минус без кавычек).

Правильный ответ

В полностью правильном ответе должны быть числа 1021, 2600, 1570, 4930 и 6988.

¹ Пример взят из заданий муниципального этапа всероссийской олимпиады школьников по информатике в г.Москве в 2015/2016 учебном году http://olympiads.ru/moscow/2015-16/vsosh/okrug_archive.shtml.

«Уроки». Пример задачи по программированию

Проверка решений автоматической системой накладывает определённые требования на оформление программ. В данном разделе приведены пример задачи и её решения на различных языках программирования².

Условие

ограничение по времени:	1 секунда
ограничение по памяти:	256 Мб

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 К-й урок?»

Формат входных данных

Вводится одно целое и положительное число К, не превышающее 15.

Формат результата

Выведите время окончания К-го урока: сначала часы, потом минуты, разделяя их пробелом.

Примеры

входные данные	результат
1	8 45
6	12 55

Структура условия

Условие олимпиадной задачи состоит из нескольких частей. Сначала 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). Потом написана «легенда» в которой говорится, что именно должна делать программа, являющаяся решением задачи.

Далее идёт описание входных данных. В описании формата входных данных указываются ограничения на эти данные. В следующей части описываются требования к тому, что должна вывести программа. В конце условия даются конкретные примеры входных и выходных данных.

² Пример взят из инструкции разработанной для школьного и окружного этапа всероссийской олимпиады школьников по информатике в г.Москве http://olympiads.ru/moscow/2013-14/vsosh/ejudge_user_manual.pdf

Процесс проверки решения

Проверка заключается в том, что программа несколько раз запускается тестирующей системой и ей на вход подается заранее подготовленный набор входных данных. Программа должна считать эти данные, решить задачу и вывести ответ.

Если ответ является правильным, то тест засчитывается как пройденный. Если программа проходит все тесты (на всех тестах выдаёт правильный ответ), то задача считается верно решенной и получает полный балл; если не все тесты — 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.

Требования к решению

Решением задачи является программа на одном из допустимых языков программирования. Тестирующая система запускает программу и подает ей данные на стандартный ввод, ожидая от программы ответа на стандартном выводе. Решение не должно использовать никакие графические функции языка программирования и операционной системы, то есть нельзя осуществлять ввод-вывод через формы, окна диалогов и так далее, поскольку проверка происходит без участия человека. То есть создаваемое приложение должно быть простым «консольным» приложением, а не графическим, использующим окна. По этим же причинам не могут быть проверены программы, предполагающие взаимодействие через интернет или с использованием браузера.

Прежде всего, программа должна считать данные. При этом она не должна выводить какие-то дополнительные сообщения, например, «Введите количество уроков». Это является ошибкой так, как сообщения предназначены для человека, а проверка происходит автоматически. Поэтому не надо писать 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 программа, будут выполнены данные ограничения, то есть не надо писать команды вида

```
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 условия задачи). Не нужно выводить никаких дополнительных сообщений типа «Ответ:»,

не нужно выводить слова «часы» или «минуты».

В конце программы часто ставят дополнительную задержку — например, ожидают ввода чего-нибудь при помощи функции *readln* в Паскале — это также не требуется так, как программа должна сразу же завершить свою работу, не дожидаясь какого-либо действия от человека.

Резюмируя все записанное выше, приведём примеры правильных программ.

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 C

```
#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 C#

```

using System;
using System.IO;
class Program
{
    static void Main()
    {
        int k = int.Parse(Console.ReadLine());
        int time = 45 * k + 5 * (k - 1);
        Console.WriteLine("{0} {1}", 8 + time / 60, time % 60);
    }
}

```

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 k, time;
        k = Integer.parseInt(in.readLine());
        time = 45 * k + 5 * (k - 1);
        System.out.println((8 + time / 60) + " " + (time % 60));
    }
}

```

Типичные ошибки

1. Программа не является консольным приложением, либо используются какие-то нестандартные возможности компилятора, привязанные к конкретной операционной

системе, например, функция *clrscr* в Паскале.

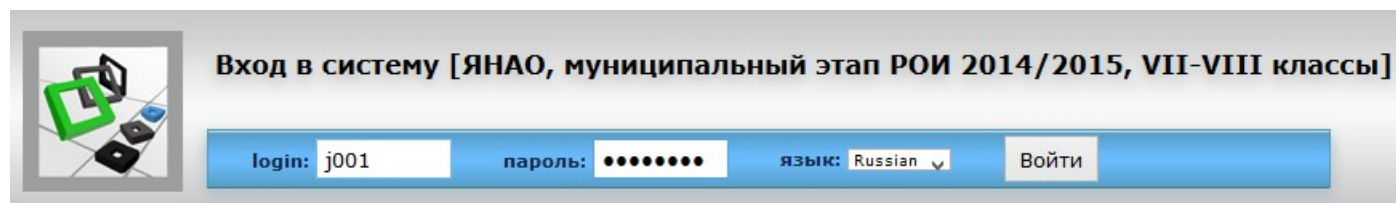
2. В программах на Паскале используется модуль *crt*.
3. Программа выводит дополнительные сообщения, например, «Введите количество уроков».
4. Программа выводит дополнительные сообщения, не предусмотренные форматом выходных данных, например, «часы», «минуты», «ответ».
5. Программа содержит «задержку» после окончания работы, то есть ждёт от пользователя нажатия на какую-либо клавишу, ввода строки или иных действий
6. Входные данные считываются не так, как в условии задачи. Например, в описании входных данных говорится, что в одной строке находится несколько чисел, а в программе каждое число считывается из отдельной строки.
7. Программа должна вывести целое число, но из-за того, что в программе вместо целочисленного типа используется действительный тип данных (например, тип *real* в языке Паскаль), программа выводит ответ в виде действительного числа, запись которого содержит точку.

Проверка решений в тестирующей системе

Данный раздел содержит информацию о том, как проверять решения в тестирующей системе.

Вход в систему

У каждого пользователя есть своя учётная запись. Мы рассматриваем действия на примере пользователя *j001*.



Вход в систему [ЯНАО, муниципальный этап РОИ 2014/2015, VII-VIII классы]

login: j001 пароль: •••••••• язык: Russian ▾ Войти

После ввода логина и пароля пользователь попадает на страницу с информацией об олимпиаде.

1 2 3 4

Состояние сервера

Турнир идёт
Решения участников проверяются

Время на сервере:	2015/04/02 06:53:31
Время начала турнира	2015/04/02 06:50:59
Продолжительность:	4:00:00
Запланированное время окончания:	2015/04/02 10:50:59
Прошедшее время:	0:02:32
Оставшееся время:	3:57:28

Вкладки 1, 2, 3 и 4 это ссылки на задачи в олимпиаде. Чтобы отправить решение по задаче 1, нужно выбрать первую вкладку.

Проверка решения

Для того, чтобы отправить решение на проверку, необходимо выбрать язык программирования и файл с текстом программы.

1 2 3 4

Сдать решение задачи 1-Кодовый замок

Ограничение времени: 1 с

Ограничение памяти: 64М

Язык: pasabc-linux - Pascal ABC.NET (Mono) v2.2.0.790

Файл cipher_op.pas

Отправить!

После отправки в статусной строке будет написано, что идёт тестирование.

07:10:57 / RUNNING/ проверка решений / Остаётся: 3:54:41 / ИДЕТ ТЕСТИРОВАНИЕ...

Время тестирования зависит от загруженности сервера и может составлять несколько минут. Дождитесь изменения статусной строки.

07:11:14 / RUNNING/ проверка решений / Остаётся: 3:54:24/ [ОБНОВИТЬ] / ТЕСТИРОВАНИЕ ЗАВЕРШЕНО

И обновите страницу. Вы увидите результат проверки.

Номер решения	Время	Размер	Язык программирования	Результат	Пройдено тестов	Баллы	Сданный ответ	Отчёт о проверке
0	0:05:17	80	pasabc-linux	OK	10	100	Просмотр	Просмотр

В данном примере, решение прошло все 10 тестов и набрало полный балл (10 баллов за тест).

Протокол проверки

Решение может не пройти все тесты.

Номер решения	Время	Размер	Язык программирования	Результат	Пройдено тестов	Баллы	Сданный ответ	Отчёт о проверке
1	0:09:51	156	pasabc-linux	Неполное решение	8	16	Просмотр	Просмотр

В этом случае необходимо посмотреть отчёт о проверке. В отчёте предоставлена более подробная информация о результатах тестирования.

Неполное решение

Всего тестов: 50, пройдено: 8, не пройдено: 42.
Получено баллов: 16 (из 100).

N	Результат	Время (с)	Баллы
1	Неправильный ответ	0.096	0 (2)
2	Неправильный ответ	0.100	0 (2)
3	Неправильный ответ	0.096	0 (2)
4	Неправильный ответ	0.092	0 (2)
5	Неправильный ответ	0.096	0 (2)
6	Неправильный ответ	0.096	0 (2)
7	Неправильный ответ	0.096	0 (2)
8	Неправильный ответ	0.096	0 (2)
9	Неправильный ответ	0.096	0 (2)
10	Неправильный ответ	0.092	0 (2)
11	ОК	0.092	2 (2)
12	ОК	0.096	2 (2)

Возможные результаты тестирования

После отправки 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 «Отчёт о проверке» будет ссылка на вывод компилятора, исходя из которого можно узнать, какие были ошибки при компиляции программы. Для интерпретируемых языков программирования компиляция не производится.

Если программа была успешно скомпилирована, то она запускается на тестах, при этом

в столбце «Отчёт о проверке» будет ссылка на протокол тестирования программы на всех тестах. По каждом из тестов возможен один из следующих результатов тестирования:

- **ОК.** Тест пройден, программа выдала правильный ответ на этом тесте.
- **Неправильный ответ.** Программа выдала неправильный ответ на данном тесте.
- **Неправильный формат вывода.** То, что вывела программа, не соответствует описанию формата выходных данных, приведённому в условию задачи. Это чаще всего другая форма сообщения «неправильный ответ».
- **Ошибка выполнения.** Программа совершила некорректную операцию во время тестирования. Возможные причины для этого: деление на ноль, извлечение корня из отрицательного числа, переполнение переменных, выход за границы массива, бесконечная (или очень большая) рекурсия.
- **Превышено максимальное время работы.** Программа не закончила свою работу за время, отведённое на исполнение одного теста.
- **Превышен лимит по памяти.** Программа использовала больше оперативной памяти, чем это предусмотрено ограничениями в задаче.