Кировское областное государственное автономное образовательное учреждение дополнительного образования детей –
«Центр дополнительного образования одаренных школьников»

ЦДООШ
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ ОДАРЕННЫХ ШКОЛЬНИКОВ

ИНФОРМАТИКА, 2013

ЗАДАНИЯ, РЕШЕНИЯ

И МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проверке и оценке решений
II (муниципального) этапа
всероссийской олимпиады школьников
по информатике

Кировской области
в 2013/2014 учебном году

Киров
2013

Печатается по решению методической комиссии II (муниципального) этапа всероссийской олимпиады школьников по информатике в Кировской области

Задания, решения и методические указания по проверке и оценке решений II (муниципального) этапа всероссийской олимпиады школьников по информатике в Кировской области в 2013/2014 учебном году / Сост. А.С. Латышев, О. А. Пестов // Под ред. О. А. Пестова. – Киров: Изд-во ЦДООШ, 2013. – 16 с.

Авторы, составители:
Латышев Алексей Сергеевич,
Пестов Олег Александрович

Научная редакция (рецензирование):
О. А. Пестов

Компьютерный набор О. Пестова
Верстка К. Коханова
Подписано в печать 01.10.2013
Формат 60x84 1/16. Бумага типографская. Усл. печ. л. 0,75

Тираж 1000 экз

ВВЕДЕНИЕ

Предметно-методическая комиссия муниципального этапа всероссийской олимпиады школьников по информатике предоставляет организаторам и жюри муниципального этапа следующий комплект материалов:

• тексты олимпиадных задач;

методику проверки решений задач, включая комплекты тестов для каждой задачи в электронном виде;

• проверяющие программы, позволяющие для каждой задачи определять правильность полученного решения;

• описание системы оценивания решений задач;

• рекомендации по использованию систем автоматической проверки;

• рекомендации по разбору предложенных олимпиадных задач.

Все вышеназванные материалы являются конфиденциальными и не подлежат преждевременному распространению до завершения муниципального этапа по информатике во всех районах Кировской области.

МЕТОДИЧЕСКИЕ ТРЕБОВАНИЯ К ПРОВЕДЕНИЮ МУНИЦИПАЛЬНОГО ЭТАПА

1. Олимпиада проводится в один очный компьютерный тур. Форма проведения – практическое решение задач. Рекомендуемая длительность олимпиады – 4 астрономических часа.

2. Рабочее место участников должно быть оснащено персональным компьютером с процессором с тактовой частотой не менее 1 ГГц и объёмом оперативной памяти не менее 512 Мбайт. На компьютере должны стоять такие среды разработки, чтобы участник мог писать программы, используя следующие языки программирования: Pascal (PascalABC.Net, Free Pascal), C/C++ (Code::Blocks, Microsoft Visual C++ Express Edition), C# (MonoDevelop, Microsoft Visual C# Express Edition), Java (JDK, Eclipse), Python (Python 2, Python 3, WingIDE 101, PyCharm). Не обязательно устанавливать все программы – достаточно только необходимые участникам.

3. Результатом работы участника является только один вариант решения каждой задачи: файл с исходным текстом программы. Черновик не учитывается при оценки работы.

4. Сдаваемая на проверку программа должна быть консольным приложением, не использующим какие-либо графические возможности операционной системы (диалоговые окна, формы ввода, средства рисования и т. д.). Программы на языке Паскаль не должны использовать модули crt и graph, программы на языке Delphi не должны использовать модуль Windows.

5. Сдаваемая программа должна читать данные со стандартного ввода (клавиатуры) и выводить результат на стандартный вывод (экран). Программа должна использовать не более 1 секунды процессорного времени, используя при этом не более 64 Мбайт памяти.

6. Участникам во время тура разрешается использовать принесённую с собой литературу на бумажном носителе, заранее заготовленные распечатанную документацию и личные записи.

7. Участникам запрещается во время тура пользоваться личными компьютерами, калькуляторами, электронными записными книжками, устройствами для чтения электронных книг, средствами связи (пейджерами, мобильными телефонами и т. п.), принесёнными электронными носителями информации (дискетами, CD и DVD, модулями флэш-памяти и т. п.).

8. Во время олимпиады участникам запрещается пользоваться Интернетом.

9. В случае возникновения технических проблем (например, сбоев в работе компьютера) участнику олимпиады по решению организаторов олимпиады может быть продлён тур на время, необходимое для устранения данных проблем. Между тем участники олимпиады самостоятельно отвечают за сохранность своих файлов и обязаны регулярно сохранять исходные коды программ.

10. Сразу после выполнения заданий проводится разбор решений, о чем следует объявить учащимся заранее, перед началом олимпиады.

11. Для участников олимпиады из районного центра (города) и близлежащих населённых пунктов не позднее, чем через 3 дня необходимо провести апелляцию, о сроках которой следует объявить перед началом олимпиады. В процессе апелляции учащиеся знакомятся со своими результатами, и, в случае несогласия с оценкой жюри, имеют право обосновать свое решение, после чего жюри может повысить оценку или оставить её без изменения.

12. Победители и призёры муниципального этапа олимпиады определяются отдельно по классам.

13. Перед проведением олимпиады, членам жюри необходимо внимательно ознакомиться с методическими рекомендациями и подготовиться к тестированию решений. В случае возникновения со стороны жюри каких-либо вопросов или замечаний по условиям задач или системе оценивания необходимо обращаться в предметно-методическую комиссию по информатике. Для оперативной связи с комиссией можно использовать адрес электронной почты oleg.pestov@gmail.com.

КОМПЛЕКТ ОЛИМПИАДНЫХ ЗАДАЧ

Для проведения муниципального этапа олимпиады по информатике предметно-методическая комиссия разработала комплект из шести задач. Все задачи пронумерованы от 1 до 6.

В разных классах используются разные задачи:

- Учащиеся 7-х классов: № 1, № 2, № 3, № 5.
- Учащиеся 8-х классов: № 1, № 2, № 3, № 5.
- Учащиеся 9-х классов: № 1, № 2, № 4, № 6.
- Учащиеся 10-х классов № 1, № 2, № 4, № 6.
- Учащиеся 11-х классов № 1, № 2, № 4, № 6.

Номера задач соответствуют их сложности, например, задачи с номерами 1 и 2 являются, по мнению методической комиссии, простейшими и должны быть доступны практически всем участникам муниципального этапа. В свою очередь, задачи с номерами 5 и 6 являются задачами повышенной сложности и ориентированы на сильнейших участников.

Важно! Во всех задачах ограничение времени работы решения на одном тесте составляет одну секунду. Ограничение на объём используемой памяти 64 мегабайта. Участники должны читать данные из стандартного потока ввода и выводить в стандартный поток вывода.

Задача 1. «Цифра»

Каждой цифре от 0 до 9 сопоставим «псевдографическое» изображение размером 5 × 3, составленное из пробелов и символов #.

```
###    #  ###  ###  # #  ###  ###  ###  ###  ###
# #   ##    #    #  # #  #    #      #  # #  # #
# #  # #  ###  ###  ###  ###  ###    #  ###  ###
# #    #  #      #    #    #  # #    #  # #    #
###    #  ###  ###    #  ###  ###    #  ###  ###
```

Требуется написать программу, которая считывает цифру и выводит изображение, соответствующее этой цифре.

Формат входных данных

Дано целое число от 0 до 9.

Формат результата

Необходимо вывести изображение, соответствующее введённой цифре.

Примеры

стандартный ввод	стандартный вывод
6	### # ### # # ###
0	### # # # # # # ###

Задача 2. «Автомобиль — не роскошь, а средство передвижения»

Коля хочет проверить, насколько верна, вынесенная в название задачи цитата из романа «Золотой телёнок». Более точно, он хочет в перспективе на три года узнать, что выгоднее — купить автомобиль или ездить на общественном транспорте и иногда на такси.

Для этого используется следующая модель:

1) При покупке машина стоит A_1 рублей, а через три года при продаже будет стоить A_2 рублей. Владение машиной требует вложений на бензин, техническое обслуживание, страховку, транспортный налог и т.д. В среднем, по статистике, A_3 рублей на один километр пробега.

2) Каждый месяц, потенциальный владелец автомобиля совершает B_1 поездок на общественном транспорте, проезжая суммарное расстояние B_2 километров. Одна поездка стоит B_3 рублей.

3) Каждый месяц, потенциальный владелец автомобиля совершает T_1 поездок на такси, проезжая суммарное расстояние T_2 километров. Стоимость вызова такси составляет T_3 рублей, а километр поездки на нём — T_4 рублей.

Напишите программу, которая по данным A_1, A_2, A_3, B_1, B_2, B_3, T_1, T_2, T_3, T_4:

- выяснит стоимость владения автомобилем в ближайшие три года;
- выяснит стоимость пользования общественным транспортом и такси в ближайшие три года.

Формат входных данных

Дано 10 неотрицательных целых чисел A_1, A_2, A_3, B_1, B_2, B_3, T_1, T_2, T_3 и T_4. Гарантируется: $A_2 \le A_1 \le 10^7$; $A_3 \le 100$; B_1, $T_1 \le 1000$; B_2, $T_2 \le 10^4$; B_3, T_3, $T_4 \le 1000$.

Формат результата

В первой строке выведите затраты на автомобиль за три года. Во второй — затраты на общественный транспорт и такси.

Примеры

стандартный ввод	стандартный вывод
360000 260000 4 60 300 15 10 100 50 12	157600 93600

Задача 3. «Вычислить выражение»

Дано выражение (((((1?2)?3)?4)?5)?6) в котором вместо каждого из знаков вопроса могут стоять знаки одного из следующих арифметических действий: + (сложение), – (вычитание), * (умножение).

Напишите программу, которая вычисляет значение выражения.

Формат входных данных

Дана строка, представляющая выражение.

Формат результата

Выведите значение выражения.

Примеры

стандартный ввод	стандартный вывод
(((((1-2)*3)*4)*5)+6)	-54
(((((1+2)-3)*4)+5)-6)	-1

Задача 4. «Расставить знаки»

Требуется в выражении (((((1?2)?3)?4)?5)?6) вместо каждого из знаков вопроса вставить знак одного из арифметических действий: + (сложение), – (вычитание), * (умножение), / (деление) так, чтобы получилось данное число X (при делении дробная часть в частном отбрасывается).

Формат входных данных

Дано целое число X ($-10^9 \le X \le 10^9$).

Формат результата

Выведите правильное выражение или impossible, если число X получить невозможно. Если правильных выражений несколько — выведите любое.

Внимание. Выражения выводите в формате (((((1?2)?3)?4)?5)?6). В них не должно быть пробелов, все цифры и скобки должны присутствовать. Вам необходимо лишь заменить знаки вопроса.

Примеры

стандартный ввод	стандартный вывод
-54	(((((1-2)*3)*4)*5)+6)
10	(((((1*2)*3)*4)/5)+6)
22	impossible

Примечание.

Правильные решения работающие при $-5 \le X \le 5$ будут оцениваться из 40 баллов.

Задача 5. «Правый больший»

Дан массив положительных целых чисел A = (a_1, a_2, …, a_n). Заменим каждый элемент на ближайший справа больший, чем он. Если такого нет, то заменим элемент на ноль.

Например, массив 2 9 8 5 9 3 4 5 2 после замены станет 9 0 9 9 0 4 5 0 0.

Требуется написать программу, которая выполняет данное преобразование и выводит получившийся массив.

Формат входных данных

В первой строке содержится целое число n ($1 \le n \le 10^5$) — размер массива. Во второй строке через пробел записаны элементы массива a_1, a_2, …, a_n ($1 \le a_i \le 10^9$).

Формат результата

Выведите получившийся массив.

Примеры

стандартный ввод	стандартный вывод
9 2 9 8 5 9 3 4 5 2	9 0 9 9 0 4 5 0 0
6 1 2 3 2 3 4	2 3 4 3 4 0

Примечание.

Правильные решения, работающие при $1 \le n \le 10^4$ будут оцениваться из 50 баллов.

Задача 6. «Массив»

Дан массив положительных целых чисел A = (a_1, a_2, ..., a_n). Требуется написать программу, которая проверяет, можно ли убрать несколько элементов (возможно ноль) с начала и конца массива так, чтобы сумма оставшихся чисел делилась без остатка на число S. При этом хотя бы один элемент должен остаться.

Например, если A = (2, 9, 8, 5, 9, 3, 4, 5, 2), а S = 15, то можно удалить один элемент с конца и сумма элементов оставшейся части будет делится на 15. Но если S = 27, то получить такой подмассив удалением элементов только с начала и конца массива нельзя.

Формат входных данных

В первой строке содержатся размер массива n ($1 \le n \le 10^5$) и S ($1 \le S \le 10^5$). Во второй строке через пробел записаны элементы массива a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$).

Формат результата

Выведите часть массива, сумма чисел в которой делится на S (помните, что элементы можно удалять только с начала и с конца исходного массива). Если ответов несколько — выведите любой. Если ответа не существует, выведите 0.

Примеры

стандартный ввод	стандартный вывод
9 26 2 9 8 5 9 3 4 5 2	5 9 3 4 5
8 13 3 1 4 1 5 9 2 6	0

Примечание.

Правильные решения, работающие при $1 \le n \le 500$ будут оцениваться из 30 баллов.

Правильные решения, работающие при $1 \le n \le 10^4$ будут оцениваться из 60 баллов.

МЕТОДИКА ПРОВЕРКИ РЕШЕНИЙ ЗАДАЧ

Для проверки решений участников предметно-методическая комиссия по информатике подготовила для муниципального жюри следующие материалы:

- комплекты тестов в электронном виде, содержащие для каждой задачи файлы входных данных и соответствующие им файлы выходных данных;
- проверяющие программы, позволяющие для каждой задачи определять правильность полученного решения в автоматическом режиме;
- эталонное решение по каждой задаче для отладки системы проверки.

Все перечисленные выше материалы для каждой задачи представлены в архиве, который поступает в распоряжение организаторов муниципального этапа олимпиады до его начала. Материалы для непосредственной проверки решений каждой задачи размещены в этом архиве в отдельной папке с соответствующим именем. Эта папка содержит:

- Папку «tests» с тестами для окончательной проверки и оценивания решений участников, и правильные ответы. Каждый тест содержится в отдельном файле. Входные файлы называются «01», «02» и т.д. Файлы с правильными ответами называются «01.a», «02.a» и т.д.
- Файл «check.cpp», представляющий программу для проверки решений участников с использованием специализированных проверяющих программных систем. Для компиляции файла «check.cpp» можно использовать GNU C++ или Visual C++. Необходимая для этого библиотека testlib находится в папке «lib» в файле «testlib.h». Скомпилированный файл «check.exe» также находится в каталоге задачи. Компиляция выполнялась компилятором g++ (tdm) 4.7.1 на операционной системе Windows 8.
- Примеры правильных решений. Каждое решение находится в отдельном файле. Этот файл имеет имя, построенное по маске «problem_ab.pas», где «problem» – идентификатор задачи, а «ab» – инициалы автора решения. Решения предоставляются только для ознакомления членов жюри с возможной реализацией правильных решений, а также для проверки работоспособности проверяющих систем. Их использование для генерации правильных ответов на тесты не требуется, так как соответствующие материалы уже содержат готовые правильные ответы на все тесты.

При проверке решений участников необходимо учитывать следующее.

1) Во всех задачах ограничение времени работы решения на одном тесте составляет одну секунду. Ограничение на объём используемой памяти 64 мегабайта. Программы должны читать данные из стандартного потока ввода и выводить в стандартный поток вывода. Участники должны знать об этих ограничениях.

2) Результатом решения всех предложенных задач является исходный текст программы на одном из языков программирования.

3) Проверка решений участников осуществляется путём исполнения программы с входными данными, соответствующими каждому тесту из предложенного предметно-методической комиссией по информатике комплекта тестов с последующим анализом получаемых в результате этого выходных файлов.

Поскольку участники олимпиады должны сдавать на проверку решения в виде исходного текста программы, то проверка решений каждого участника должна осуществляться в следующей последовательности:

- компиляция исходного текста программы;
- последовательное исполнение программы с входными данными, соответствующими тестам из набора тестов для данной задачи,

• сравнение результатов исполнения программы на каждом тесте с правильным ответом.

Если решение не может быть скомпилировано, то член жюри пытается устранить причины ошибки компиляции, внося исправления в исходный код программы, **не модифицируя алгоритм решения задачи**. Если этого сделать не удалось, то результатом является «Ошибка компиляции» и за задачу не начисляется ни одного балла.

При исполнении программы на каждом тесте, в первую очередь, жюри определяет, успешно ли программа была исполнена и не нарушаются ли ограничения на время работы программы на отдельном тесте и размер доступной программе памяти в процессе её исполнения. В случае нарушения имеющих место ограничений баллы за этот тест участнику не начисляются. Возможные результаты тестирования в этом случае на каждом тесте могут быть следующими:

Ошибка выполнения	Программа совершила некорректную операцию во время работы (выход за границы массива, деление на ноль, ошибки при работе с памятью и т.д.)
Превышено максимальное время работы	Программа использовала более 1 секунды процессорного времени
Превышен лимит по памяти	Программа использовала больше 64 Мбайт оперативной памяти

Если приведённые в условии задачи ограничения не нарушаются в процессе исполнения программы с входными данными, соответствующими конкретному тесту, то после завершения исполнения программы осуществляется проверка правильности полученного ответа. В некоторых задачах невозможно проверить правильность сравнением ответа с ответом жюри и необходимо использовать проверяющую программу. Проверяющая программа запускается в командной строке с тремя параметрами — входной файл, выходной файл и файл с правильным ответом (например, check.exe 02 02.o 02.a). Выходной файл можно получить если при запуске решения в командной строке перенаправить вывод в файл.

Возможные результаты тестирования в этом случае могут быть такими:

OK	Программа выдала правильный ответ
Неправильный ответ	Программа выдала неправильный ответ
Неправильный формат вывода	Выведенный программой результат не соответствует описанию формата выходных данных, указанных в условии.

Решение должно выдавать одинаковые ответы на одинаковые тесты, вне зависимости от времени запуска и программного окружения. Жюри олимпиады вправе произвести неограниченное количество запусков программы участника и выбрать наихудший результат по каждому из тестов.

Если программа не проходит ни одного теста, то причина этого, как правило, заключается в неправильной реализации ввода-вывода, например, в наличии лишнего вывода в программе. В этом случае член жюри изучает исходный код программы и пытается внести исправления в программу так, чтобы программа удовлетворяла требованиям к решениям. При этом **запрещается модифицировать алгоритм решения задачи**, т.е все исправления должны касаться только ввода-вывода или каких-либо частей программы, не связанных с алгоритмом решения (например, подключение тех или иных библиотек, устранение «задержки» после работы программы и т. д.). Внеся необходимые изменения, член жюри повторно проверяет программу в автоматическую тестирующую систему, повторяя этот процесс при необходимости.

СИСТЕМА ОЦЕНИВАНИЯ РЕШЕНИЙ УЧАСТНИКОВ

Система оценивания решений каждой задачи основана на следующих положениях:

1) Максимальное количество баллов, которое участник может получить за полное решение каждой задачи, составляет 100 баллов.

2) Общая оценка за решение отдельной задачи конкретным участником складывается из суммы баллов, начисленных ему по результатам исполнения всех тестов из набора тестов для этой задачи в процессе окончательной проверки всех решений после олимпиады.

3) Итоговый результат каждого участника подсчитывается как сумма полученных этим участником баллов за решение каждой задачи. С учётом того факта, что всего предлагается четыре задачи, то максимально возможное количество баллов, которое может набрать участник по итогам муниципального этапа, составляет 400 баллов.

Для предложенных методической комиссией по информатике задач муниципального этапа оценка для каждого теста из комплекта тестов для каждой задачи является одинаковой. В таблице ниже эти оценки представлены.

Задача	*Количество тестов*	*Оценка теста*
1. «Цифра»	10	10
2. «Автомобиль – не роскошь...»	10	10
3. «Вычислить выражение»	10	10
4. «Расставить знаки»	10	10
5. «Правый больший»	10	10
6. «Массив»	10	10

Окончательные результаты проверки решений всех участников фиксируются в пяти итоговых таблицах – для обучающихся 7-х, 8-х, 9-х, 10-х и 11-х классов. Каждая таблица представляет собой ранжированный список участников соответствующих классов, расположенных по мере убывания набранных ими баллов. Участники с одинаковыми баллами располагаются в алфавитном порядке. На основании этих таблиц жюри принимает решение о победителях и призёрах муниципального этапа олимпиады по каждому классу.

АВТОМАТИЗАЦИЯ ПРОЦЕССА ПРОВЕРКИ РЕШЕНИЙ УЧАСТНИКОВ

Существуют различные способы проверки решений участников. Самый трудоёмкий, 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. Если учесть, что для каждой задачи эти файлы предоставляются жюри методической комиссией по информатике, то члены жюри вполне могут справиться с задачей проверки решений участников таким «ручным» способом при наличии достаточного количества членов жюри.

Конечно, описанный способ возможен, но он не совсем корректен в плане проверки соответствия решения ограничениям на время и используемую память. Оценить сколько времени работала программа: 1 секунду или на одну десятую дольше практически невозможно без использования вспомогательных средств. Аналогичное замечание относится и к количеству используемой памяти.

Поэтому, методическая комиссия рекомендует применять специальные средства для проверки решений участников:

1) Автоматическая тестирующая система, доступная по адресу http://olymp43.dyndns.org. Данная система настроена специально для проведения муниципального этапа олимпиады по информатике в Кировской области. За её функционирование отвечает предметно-методическая комиссия.

2) Программа для проверки решения на наборе тестов «Тестер» (адрес для скачивания http://acm.timus.ru/tester/). Если по каким-то причинам отсутствует возможность пользоваться автоматической тестирующей системой, то можно воспользоваться данной программой для автоматизации большей части процедуры проверки.

Кроме указанных, жюри также может использовать другие программные системы проведения олимпиад по информатике. В том числе – собственные разработки.

Использование автоматической тестирующей системы

По адресу http://olymp43.dyndns.org доступна система для автоматического тестирования решений. Используя её, члены жюри могут проверить решения участников за гораздо меньшее время, чем при «ручном» способе.

Порядок действий жюри в этом случае:

1) Перед началом процедуры проверки, необходимо будет написать письмо по адресу oleg.pestov@gmail.com и получить имена и пароли учётных записей для доступа к тестирующей системе, сообщив количество участников муниципального этапа по каждому из классов. Одна учётная запись используется для проверки решений одного участника.

2) Зайдя в систему под выбранной учётной записью, проверяющий по очереди сдаёт решения участника в тестирующую систему. После проверки будет доступен полный протокол с указанием пройденных тестов и набранных баллов.

3) Если автоматическая тестирующая система не смогла скомпилировать решение (результат тестирования – «Ошибка компиляции»), то член жюри изучает протокол тестирования и пытается устранить причины ошибки компиляции, внося исправления в исходный код программы, **не модифицируя алгоритм решения задачи**. Если этого сделать не удалось, то нужно перейти к пункту 6.

4) Если решение было скомпилировано в автоматической тестирующей системе, то член жюри смотрит на количество пройденных тестов в автоматической тестирующей системе. Если программа прошла хотя бы один тест, то признается, что программа была проверена автоматической тестирующей системой и в этом случае программа оценивается тем количеством баллов, которым было оценено решение при помощи автоматической тестирующей системы.

5) Если программа не проходит ни одного теста, то причина этого, как правило, заключается в неправильной реализации ввода-вывода, например, в наличии лишнего вывода в программе. В этом случае член жюри изучает протокол тестирования и исходный код программы и пытается внести исправления в программу так, чтобы программа удовлетворяла требованиям к решениями,

проверяемых автоматической тестирующей системой. При этом **запрещается модифицировать алгоритм решения задачи**, т.е все исправления должны касаться только ввода-вывода или каких-либо частей программы, не связанных с алгоритмом решения (например, подключение тех или иных библиотек, устранение «задержки» после работы программы и т. д.). Внеся необходимые изменения, член жюри повторно сдаёт программу в автоматическую тестирующую систему, повторяя этот процесс при необходимости.

6) Если программа не может быть скомпилирована автоматической тестирующей системой или она не проходит ни одного теста, то член жюри проводит «ручное» тестирование решения. То есть, самостоятельно компилирует и запускает решение на тестах, разработанных предметно-методической комиссией. При этом большие тесты желательно копировать через «буфер обмена», а не вводить руками.

В автоматической тестирующей системе поддерживаются следующие языки программирования и компиляторы:

- Pascal: компиляторы Free Pascal, PascalABC.NET, Borland Delphi.
- C: компилятор GNU C.
- C++: компилятор GNU C++.
- Python: версии 2 и 3.
- C#: компилятор Mono C#
- Java: версия 1.7

Подробная инструкция о том, как пользоваться тестирующей системой расположена по адресу http://olymp43.dyndns.org/doc/guide.html. До 30 ноября 2013 года на сайте http://olymp43.dyndns.org будет доступен пробный тур, который можно использовать для ознакомления работы с системой.

Использование программы «Тестер»

Если по каким-то причинам отсутствует возможность пользоваться автоматической тестирующей системой, то можно воспользоваться свободно распространяемой программой «Тестер» для автоматизации большей части процедуры проверки. Программу можно скачать с официального сайта http://acm.timus.ru/tester/. Также она будет находиться в архиве, который предоставляется организаторам муниципального этапа (файл «tester100326.rar» расположен в папке «lib»). Программа предназначена для операционной системы Windows.

Официальная документация по использованию находится в архиве «tester100326.rar» в файле «readme.txt». Ниже, в качестве примера, приводится порядок действий для тестирования решения по задаче 1. Тестирование других задач осуществляется аналогично.

- Пусть участник сдал решение, которое называется «sol1.pas». Сначала его необходимо скомпилировать, чтобы создать исполняемый файл. Пусть этот файл называется «sol1.exe».
- Далее создадим директорию для проверки. В неё скопируем файл «sol1.exe». Из архива с материалами олимпиады, из папки «1», скопируем в созданную директорию папку «tests» и проверяющую программу «check.exe». Из архива с материалами олимпиады из папки «lib», скопируем в созданную директорию файл «!test.ini». Из архива «tester100326.rar» скопируем в созданную директорию файл «!test.exe».
- Перейдём в командной строке в созданную директорию и выполним команду «!test.exe sol1.exe». На экран выведется протокол проверки.

УКАЗАНИЯ ПО РЕШЕНИЮ ЗАДАЧ

Задача 1. «Цифра»

Задачу можно решить с помощью условного оператора. Программа читает вводимое число и, в зависимости от его значения, выводит искомое изображение.

```
var n : integer;
begin
  read(n);
  if n = 0 then begin
    writeln('###');
    writeln('# #');
    writeln('# #');
    writeln('# #');
    writeln('###');
  end
  else if n = 1 then begin
  ...
end.
```

Решение правильное. Но можно написать более короткое, используя константные массивы и циклы.

```
const s : array[1..5] of string = (
  '###  ######## ################',
  '# # ##  #  ## ##  #    ## ## #',
  '# ## ################  #######',
  '# #  ##    #  #  ## #  ## #  #',
  '###  #######  #######  #######');
var n, i : integer;
begin
  read(n);
  for i := 1 to 5 do
    writeln(s[i][3*n+1], s[i][3*n+2], s[i][3*n+3]);
end.
```

Задача 2. «Автомобиль – не роскошь, а средство передвижения»

Необходимо аккуратно вычислить по формулам то, что требуется в условии. Задача на арифметические операции и ввод/вывод целых чисел.

```
var a1, a2, a3, b1, b2, b3, t1, t2, t3, t4 : longint;
begin
  read(a1, a2, a3, b1, b2, b3, t1, t2, t3, t4);
  writeln(a1 - a2 + (b2 + t2) * a3 * 12 * 3);
  writeln((t1 * t3 + t2 * t4 + b1 * b3) * 12 * 3);
end.
```

Задача 3. «Вычислить выражение»

Заметим, что в выражении всего 5 знаков, которые находятся в строке на позициях 7, 10, 13, 16 и 19. В цикле переберём каждую позицию и, в зависимости от знака, выполним соответствующую арифметическую операцию.

```
var s : string;
    i, j, t : integer;
begin
  readln(s);
  t := 1;
  j := 7;
  for i := 2 to 6 do begin
    if s[j] = '+' then
      t := t + i
    else if s[j] = '-' then
      t := t - i
    else
      t := t * i;
    j := j + 3;
  end;
  writeln(t);
end.
```

Задача 4. «Расставить знаки»

В условии сказано, что правильные решения, работающие при $-5 \le X \le 5$ оцениваются из 40 баллов. Такое решение можно написать, найдя ответы самостоятельно «на бумажке» и затем с помощью условного оператора выводить правильный вариант.

```
var x : integer;
begin
  read(x);
  if x = -5 then
    writeln('(((((1+2)-3)-4)+5)-6)')
  else if x = -4 then
    writeln('(((((1+2)+3)+4)/5)-6)')
  else if x = -3 then
  ...
end.
```

Чтобы набрать полный балл, необходимо написать решение, которое перебирает всевозможные способы расстановки знаков вместо знаков вопроса. Сделать это можно, например, пятью вложенными циклами. Тогда для каждого способа, нужно будет вычислить значение выражения и сравнить с X. Алгоритм вычисления значения выражения описан в разборе задачи 3.

```
const o : array[1..4] of char = ('+', '-', '*', '/');
<объявление переменных>
begin
  read(x);
  s := '(((((1?2)?3)?4)?5)?6)';
  for a := 1 to 4 do
  for b := 1 to 4 do
  for c := 1 to 4 do
  for d := 1 to 4 do
  for e := 1 to 4 do
  begin
    s[7] := o[a]; s[10] := o[b]; s[13] := o[c];
    s[16] := o[d]; s[19] := o[e];
    if <значение выражения s> = x then begin
      writeln(s);
      exit;
    end;
  end;
  writeln('impossible');
end.
```

Другой, более компактный по размеру программы, способ, набрать полный балл – рекурсивная генерация всех возможных расстановок знаков арифметических действий.

Задача 5. «Правый больший»

В условии сказано, что правильные решения, работающие при $1 \le n \le 10^4$ оцениваются из 50 баллов. Это решения, которые для каждой позиции от 1 до n с помощью цикла ищет правый больший элемент в массиве.

```
var a : array [1..100000] of integer;
    i, j, n : integer;
begin
  read(n);
  for i := 1 to n do
    read(a[i]);
  for i := 1 to n do
  begin
    j := i + 1;
    while (j <= n) and (a[j] <= a[i]) do
      j := j + 1;
    if (j > n) then a[i] := 0 else a[i] := a[j];
  end;
  for i := 1 to n do
    write(a[i], ' ');
end.
```

Сложность такого решения – $O(n^2)$. При больших значениях n оно будет работать дольше секунды. Чтобы набрать полный балл, необходимо рассматривать массив с конца. Заведём вспомогательный массив B, где $B[i]$ – это индекс правого большего элемента для элемента с номером i. Пусть мы хотим узнать $B[k]$, и так, как массив рассматривается с конца, то значения $B[k+1]$, $B[k+2]$, ..., $B[n]$ уже известны. Тогда рассмотрим элемент $A[k+1]$. Если $A[k+1]>A[k]$, то $B[k]=k+1$. Если же нет, то не имеет смысла рассматривать $A[k+2]$, $A[k+3]$ и т. д. Поскольку нам нужен элемент больший $A[k]$, то необходимо сразу перейти к первому элементу большему $A[k+1]$, то есть к элементу с номером $B[k+1]$. И так делать до тех пор, пока не найдём больший.

```
var a, b : array [1..100001] of longint;
    i, n : longint;
begin
  read(n);
  for i := 1 to n do
    read(a[i]);
  a[n + 1] := 2000000000;

  for i := n downto 1 do
  begin
    b[i] := i + 1;
    while a[b[i]] <= a[i] do
      b[i] := b[b[i]];
  end;

  for i := 1 to n do
    if b[i] = n + 1 then
      write('0 ')
    else
      write(a[b[i]], ' ');
end.
```

Чтобы цикл поиска большего гарантировано остановился, необходимо добавить фиктивный элемент в массив на позицию $n+1$. Сложность данного решения это $O(n)$. Заметим, что если мы один раз выполнили переход b[i] := b[b[i]], то больше он не повторится. Поскольку суммарное количество переходов n, то и суммарно количество итераций внутреннего цикла не превосходит n.

Задача 6. «Массив»

Решение которое набирает 30 баллов, перебирает все возможные части и вычисляет сумму на них. Если сумма делится на S, то ответ найден. Три вложенных цикла дают сложность O(n^3). Это решение можно ускорить если заметить, что проверять делится ли сумма элементов с i по j на S, можно без цикла. Для этого заранее подсчитаем массив P, где $P[i]$ это остаток от деления суммы первых i элементов на S. Заметим, что если $P[i\text{-}1]=P[j]$, то сумма элементов с i по j делится на S.

```
var n, s, i, j, k : longint;
    a : array[1..100000] of longint;
    p : array[0..100000] of int64;
begin
  p[0] := 0;
  read(n, s);
  for i := 1 to n do begin
    read(a[i]);
    p[i] := (p[i-1] + a[i]) mod s;
  end;
  for i := 1 to n do
    for j := i to n do begin
      if p[j] = p[i-1] then begin
        for k := i to j do
          write(a[k], ' ');
        writeln;
        exit;
      end;
    end;
  writeln(0);
end.
```

Такое решение имеет сложность O(n^2) и позволяет набрать 60 баллов. Чтобы получить 100 баллов, необходимо заметить, что мы ищем такие i и j, что $P[i - 1]=P[j]$. То есть, мы ищем в массиве P два одинаковых элемента. Это можно сделать с помощью вспомогательного массива Q, в котором для каждого элемента из массива P хранится индекс его первого вхождения.

```
var n, s, i, j : longint;
    a, p, q : array[0..100000] of longint;
begin
  p[0] := 0;
  read(n, s);
  for i := 1 to n do begin
    read(a[i]);
    p[i] := (p[i - 1] + a[i]) mod s;
  end;
  fillchar(q, sizeof(q), 255);
  q[0] := 0;
  for i := 1 to n do begin
    if q[p[i]] <> -1 then begin
      for j := q[p[i]] + 1 to i do
        write(a[j], ' ');
      exit;
    end;
    q[p[i]] := i;
  end;
  writeln(0);
end.
```