Андреева Татьяна Анатольевна

СТРУКТУРА И КЛАССИФИКАЦИЯ ТЕКСТОВ ОЛИМПИАДНЫХ ЗАДАЧ

Среди олимпиадных задач по программированию и информатике (уже в силу широты предметной области) можно найти настолько разнообразные, совершенно не похожие друг на друга варианты, что порой кажется, будто никаких закономерностей построения в их условиях нет и быть не может. Но именно это разнообразие и предоставляет самый богатый материал для анализа.

Чтобы не заслужить упрека в ненужной затейливости, ограничимся лишь теми направлениями систематизации, которые могут представлять практический интерес с точки зрения людей, решающих или составляющих какие-либо задачи.

1. СТРУКТУРА ТЕКСТА ЗАДАЧИ

В структуре условий (текстов) задач можно с очевидностью выделить следующие части:

ВИ. ***Вводная информация*** обрисовывает и обозначает предметную область, к которой относится задача, намечает границы этой области.

НО. ***Необходимые определения*** – соглашения и терминология, используемые в тексте задачи.

ОП. ***Описание проблемы*** – более или менее формализованная постановка собственно задачи и задание ее границ.

Зд. ***Задание*** – требования, при выполнении которых задача будет считаться решенной.

ФД. ***Форматы входных и выходных данных*** [1].

Пр. ***Пример.***

Любая из шести перечисленных частей может отсутствовать (см. также пункт «Классификация задач с точки зрения полноты условия»), другие части могут идти в произвольном порядке. Однако обычно более удобными для понимания являются те задачи, структурные части текста которых располагаются именно в приведенном выше порядке.

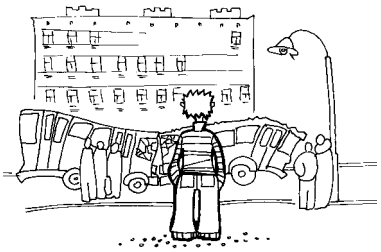
Из приведенных ниже примеров видно, что степень формализации любой из перечисленных частей также может быть различной: рассказ – 0, описание – 1, объяснение – 2, формальное задание – 3.

Кроме того, в тексте задачи могут присутствовать два-три варианта одной и той же части. Например, в задаче «Возня с мелочью» (см. таблицу 1) первая часть дана два раза – на разных уровнях формализма. А в задаче «Гусеницы» (см. таблицу 1) дано двухуровневое (описание и объяснение) третьей и четвертой частей условия.

Более того, на базисе одной и той же четвертой части, наращивая на нее различные описания, можно получить разные по виду, но совершенно одинаковые по сути задачи, что очень полезно для составителей олимпиадных задач, для кружковой работы и т. п. Например, задачи «Родственники» и «Подсчет компонент связности» (см. таблицу 2) являются одной и той же задачей, поставленной в разных степенях формализма. Остальные задачи в этой таблице также разбиты на эквивалентные пары.

[1] Этот пункт актуален только для задач по программированию.

Таблица 1. **Примеры полных текстов задач**

Выделяемая часть	Текст задачи
1	2
ВИ	**«Из жизни гусениц».**  Видели Вы когда-нибудь, как веселились мюмзики в траве в жаркие летние дни? Видели Вы, как зазеркальные гусеницы резвились под солнечными часами в те дни, когда Единорог еще жил в Лесу, рыцари еще проезжали по нему, и Большой Свалки еще не было? Это было незабываемое зрелище! Зазеркальные гусеницы всех расцветок гонялись друг за другом, кувыркались, дурачились. Иногда большие гусеницы возили на себе маленьких, иногда две гусеницы играли в чехарду, переползая друг через друга.
НО	Вы, конечно, знаете, что место вокруг солнечных часов имеет вид идеально плоской площадки, на которую в незапамятные времена неизвестно кем была нанесена координатная сетка, так что каждой точке были сопоставлены координаты x и y, не превышающие по абсолютной величине 1000**.** Конечно, солнечные часы находятся в точке с координатами (0, 0). В один момент оказалось, что все гусеницы расположились так, что приняли форму отрезков прямой с вершинами в узлах целочисленной координатной сетки.
ОП	Другими словами, положение каждой гусеницы стало возможным описать набором координат (x_1, y_1, x_2, y_2), где все числа x_1, y_1, x_2, y_2 – целые и задают координаты головы и хвоста гусеницы. Отдельные очень маленькие гусеницы могут стягиваться в точку.
Зд	Рассмотрим какую-нибудь пару гусениц. Вы должны определить, чем они занимаются: бегают наперегонки (то есть не пересекаются), играют в чехарду (пересекаются по одной точке) или одна из них возит на себе другую (пересекаются по многим точкам, не обязательно при этом один из отрезков полностью содержится в другом, допускается и частичное наложение).
ФД-1	Входные данные Входной файл состоит из двух строк, в каждой из которых находятся по четыре числа, задающих в формате (x_1, y_1, x_2, y_2) расположение гусеницы на плоскости. Выходные данные В выходной файл следует вывести одно из чисел 0, 1, 2 в зависимости от того, имеют ли данные гусеницы ноль, одну или не менее двух точек пересечения.
Пр	*Примеры:* input.txt 1 0 2 0 1 1 2 1 output.txt 0 input.txt -1 0 1 0 0 1 0 –1 output.txt 1 input.txt -2 –2 1 1 110 110 0 0 output.txt 2

Продолжение таблицы 1

1	2
ВИ	**«Возня с мелочью».**  «Негусто, – подумал Петя, пошарив в кармане, – даже, если честно сказать, очень не густо». Где-то в глубине кармана глухо позвякивали две неизвестно как закатившиеся туда монетки – и все. Петя осторожно ощупал их: монетки были сред них размеров. Явно не пятирублевые, что уж совсем огорчительно. Но, с другой стороны, и не копейки. То ли рублевые, то ли двух– рублевые, – на ощупь не понять. Ну, ладно, хватит ощупывать их, пора извлекать на свет божий из кармана: кондуктор уже, кажется, целую вечность стоит над душой. Билет стоит два рубля. Интересно, удастся ли заплатить за билет? Неприятностей хотелось по возможности избежать. Ну, конечно, проблем не будет: даже если в кармане две рублевые монетки, то на билет хватит. Пете явно везло в тот день: обе монетки оказались двухрублевыми, так что вдобавок к билету у него появился вполне реальный шанс стать еще и счастливым обладателем порции мороженого. В этих радостных размышлениях он наткнулся еще на одну мысль: «Забавно. Вне зависимости от того, какие монетки были у меня в кармане, я все равно мог заплатить за проезд без сдачи – либо одной двухрублевой монеткой, либо двумя рублевыми.
Пр	Немного поразмыслив, Петя пришел к выводу, что 2 рубля – единственная сумма, которую он с гарантией мог заплатить без сдачи. В самом деле, 1 рубль он не смог бы заплатить, окажись в кармане две двухрублевые монетки, а 3 или 4 рубля – если две рублевые.
НО	Давайте рассмотрим возникшую ситуацию более пристально. Представьте себе, что в некотором государстве в обращении имеются монеты K различных достоинств номиналом в V_1, V_2, ... V_K тугриков. Допустим также одно упрощение задачи: будем считать, что каждый следующий номинал делится на предыдущий, то есть V_1,$\|V_2\|$... $\|V_K$.
Пр	Это значит, что в обращении могут быть монеты достоинством, например, 1, 2, 6 и 30 тугриков, но не 1, 2 и 5 тугриков (так как 5 не делится на 2).
ОП	Представьте себе, что Вы живете в этой самой «некоторой стране» и нащупали у себя в кармане ***N*** монет, которые могут быть любого номинала из имеющихся в обращении.
Зд	Вы хотите определить максимальную сумму, которую Вам заведомо удастся набрать без сдачи этими монетами, какого бы достоинства они ни оказались.
ФД-12	Входные данные Входной файл для этой задачи состоит из двух строк: в первой находится единственное число N **–** количество монет в Вашем кармане, причем $1<=N<=100$. Во второй строке находится число K – количество различных номиналов монет, находящихся в обращении ($1<=K<=20$); а затем через пробелы K натуральных чисел V_1, V_2, ... V_K **–** достоинства этих монет, удовлетворяющие соотношению $1<=V_1<=V_2<=...<=V_K<=1000$.

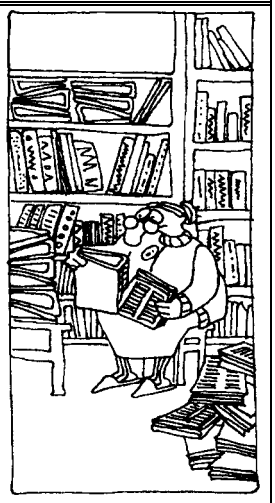
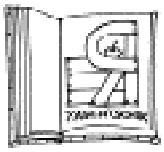
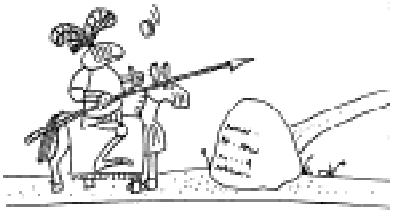
Продолжение таблицы 1

1	2
ФД-12	Выходные данные В выходном файле должно находиться единственное число – максимальная сумма, которую заведомо можно набрать без сдачи с помощью N монет, каких бы достоинств они не оказались. В том случае, когда такой суммы не существует, в выходной файл следует вывести **0**.
Пр	*Пример:* input.txt output.txt 15 12 3 1 2 6

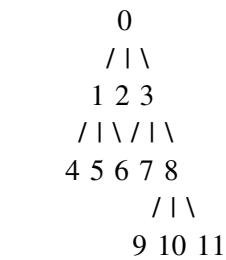
Таблица 2. **Примеры эквивалентных задач**

Отсутствующие части	Выделяемая часть	Текст задачи
1	2	3
ОП, Пр	**ВИ**	**«Родственники» («Подсчет компонент связности-1»).** В деревне Большие Ключи живут несколько семей. Будем считать, что семья – это абсолютно все люди, являющиеся родственниками друг другу, не зависимо от степени родства. Для каждой пары жителей известно, родственники они или нет 
	НО	
	Зд	Напишите программу, подсчитывающую количество семей в деревне Большие Ключи.
	ФД-13	Формат входных данных: Входной файл input.txt содержит в первой строке натуральное число $N < 1000$ – количество жителей, в остальных строках по два имени, разделенных пробелом – пары родственников. (Считаем, что имена всех жителей различны и состоят не более чем из 10 латинских символов.) Формат выходных данных: одно число – количество семей.
ВИ, ОП, Пр	**НО**	**«Подсчет компонент связности-2».** Граф – это структура, состоящая из одной или нескольких вершин, которые могут быть соединены между собой ребрами, по которым можно двигаться в обе стороны. Считаем, что вершина V достижима из вершины S, если найдется хотя бы один непрерывный путь по ребрам от вершины S до вершины V. Компонента связности – это изолированная часть графа, все вершины которой достижимы друг из друга, а вершины из любой другой компоненты связности – не достижимы.

Продолжение таблицы 2

1	2	3
ВИ, ОП, Пр	**Зд**	Напишите программу, подсчитывающую количество компонент связности в графе, заданном перечислением всех его ребер (начальной и конечной вершины каждого ребра).
	ФД-13	Изолированная вершина задается просто номером на отдельной строке.
НО, ОП, ФД, Пр	**ВИ**	**«Поиск по ключевым словам – 1».**  В библиотечном деле для быстрого поиска нужных материалов их содержание обычно характеризуется так называемыми ключевыми словами, заданными или автором материала, или библиотечными работниками с привлечением специалистов по тематике. Нередко такие ключи реально соотносятся с чересчур большим числом источников, среди которых затруднительно выделить нужное подмножество. Эта информация может при необходимости учитывать синонимию.
	Зд	Разработать программные средства для характеристики текстов с учетом заданного списка терминов и отношения их синонимии.
НО, ВИ, ОП, Пр	**ФД-3**	**«Поиск по ключевым словам – 2».**  Имеется набор ключевых слов. Для некоторых из них заданы также списки их синонимов. Все слова состоят только из латинских букв. Предполагается, что информация о ключевых словах и их синонимах уже записана в файл keys.txt: каждый синонимический ряд – на отдельной строке.
	Зд	Напишите программу, которая для каждого ключевого слова подсчитывала бы, сколько раз оно и все его синонимы встречаются в некотором тексте (файл text.txt), состоящем из латинских слов, пробелов и различных знаков препинания.
	ФД	Считайте, что искомые слова встречаются в проверяемом тексте только в той форме, в какой они представлены в списках ключей и синонимов (множественное число, падежи и т. п. не учитываются). Результат выдать в порядке убывания частот.
НО, Зд, ФД, Пр	**ВИ**	**«Дороги–1».**  В некотором сказочном царстве-государстве из стольного града ведут три дороги. А каждая дорога через день пути или в деревеньку приводит, или, как в сказках водится, снова делится на три пути и т. д.

Продолжение таблицы 2

1	2	3
НО, Зд, ФД, Пр	**ОП**	Каждая деревенька может выставить 10 воинов. А случись беда, сколько воинов за неделю в столицу соберутся?
НО, ОП	**ВИ**	**«Дороги–2».** В некотором сказочном царстве-государстве из стольного града ведут три дороги. А каждая дорога через день пути или в деревеньку приводит, или, как в сказках водится, снова делится на три пути и т. д.
НО, ОП	**Зд**	Напишите программу, которая подсчитывала бы, сколько воинов смогут собраться в столицу за указанный срок, если каждая деревенька может выставить 10 воинов.
	ФД-13	Входные данные: Во входном файле input.txt должна содержаться следующая информация: в первой строке: одно натуральное число *D* – количество дней, отведенных для сбора войска; во второй строке: одно натуральное число *N* – количество деревень и перекрестков в стране (столица здесь не считается); в каждой из оставшихся строк: список деревень либо перекрестков, до которых можно за 1 день добраться с перекрестка, номер которого стоит в списке первым, если двигаться по дорогам в направлении удаления от столицы. (Этот список может не быть упорядоченным). Формат выходных данных: Одно число – количество успевающих к битве воинов.
	Пр	Пример входного файла, соответствующего картинке: 2 11 0 1 2 1 4 5 6 3 6 7 8 8 9 10 11  Для этого входного файла ответом будет 50 – именно столько воинов успеют собраться из деревень 2, 4, 5, 6, 7 за отведенные 2 дня. Воины из 9, 10 и 11 деревень к битве не успеют. Пункты 1, 3, 8 деревнями не являются.

1.1. КЛАССИФИКАЦИЯ ЗАДАЧ С ТОЧКИ ЗРЕНИЯ ПОЛНОТЫ УСЛОВИЯ

1) Если в условии задачи полностью или почти полностью отсутствует вводная информация (часть ВИ), то условие становится «сухим», чисто математическим – идеальный вариант, когда кто-то уже формализовал задачу, что на практике (когда задача возникает сама по себе, а не составляется специально) встречается редко.

Отсутствие этой части не всеми воспринимается как недостаток формулиров-

ки задачи: многие составители считают несолидными задачи с многословным литературным введением. Эти люди полагают, что серьезные олимпиадные задачи не должны содержать в себе лирики. Однако следует отметить, что задача, условие которой написано красочно, с оригинальной вводящей историей, а иногда и с юмором, гораздо приятнее для чтения и понимания: в условиях олимпиадного стресса очень ценна и полезна возможность хоть немного улучшить настроение участникам, снять нервное напряжение, просто доставить удовольствие от остроумной и неожиданной формулировки. Сравните, например, две задачи: «Родственники» и «Подсчет компонент связности» (см. таблицу 2) – обе формулировки задают, по сути, одну и ту же задачу, но делают это на разных уровнях формализма.

2) Если отсутствуют определения (часть НО), то задача опирается только на базовые, общепринятые или интуитивные понятия, что может таить в себе самые сложно обнаруживаемые ошибки, всплывающие только постфактум, когда из предоставленных решений становится ясно, в каком месте получилось разночтение. Однако если предметная область вполне ясна, отсутствие этой части не нанесет урона понимаемости задачи. Примерами могут служить задачи «Поиск по ключевым словам –1», «Дороги-2» (см. таблицу 2, таблицу 3).

3) Если в условии нет описания проблемы (часть ОП), то это – задача на моделирование, требующая творческого подхода и анализа. Например, задача «Дороги» (см. таблицу 2, сравните с более полным вариантом этой же задачи).

В практической жизни такие задачи, особенно с подачи неквалифицированных заказчиков, встречаются очень часто: большей части программистов приходилось попадать в ситуацию, когда заказчик и сам толком не знает, что же, собственно, он хочет получить в результате написания программного комплекса. И здесь приходится уже вникать в проблемную область глубже, чем необходимо при четко сформулированных требованиях, проявлять смекалку и догадливость, зачастую работать на ощупь, постепенно приближаясь к желаемому результату методом «проб и ошибок». (Впрочем, некоторые виртуозы своего дела считают, что если заказчик не слишком четко представляет себе требования, которым должна отвечать программная система, то в этом случае его можно довольно легко убедить в том, что написанный программистом вариант – это и есть именно то, что заказчик на самом деле хотел.)

4) Если в условии задачи нет явно выписанного задания (часть Зд), то это – задача на формализацию задания. Например, задача «Поиск по ключевым словам» (см. таблицу 2).

5) Если нет четко заданных требований к формату входных и выходных данных (часть ФД), то задача подразумевает разработку наиболее удобного интерфейса. Известно ведь, что внутреннее и внешнее представления данных практически никогда не совпадают.

Например, задачи «Подсчет компонент связности», «Поиск по ключевым словам» (см. таблицу 2).

Заметим, что задачи, предполагающие моделирование, разбор вариантов или некоторую формализацию условия, редко решаются в течение очного тура олимпиады полностью. Вполне естественное желание участников сэкономить силы и время приводит к тому, что каждый решающий сводит подобную недоопределенную задачу (порой неявно) к некоторому ее частному случаю.

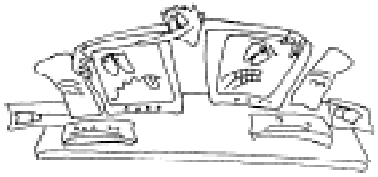
Существует еще один вид олимпиадных задач – так называемые *задачи на понимание условия*. В этих задачах:

- Либо слишком объемистая первая часть условия, где из-за многословности трудно выделить нужную информацию. Иногда этой информации значительно больше, чем требуется для решения задачи.
- Либо намеренно усложнены вторая и третья (иногда и четвертая) части, и сквозь обилие цепляющихся друг за друга определений просто очень трудно продраться.

2. РАЗМЕРНОСТЬ ПРОСТРАНСТВА ВХОДНЫХ ДАННЫХ

Если обратить внимание на входные данные (в тех задачах, где

Таблица 3. **Примеры неполных текстов задач**

<table>
<tr><th>Отсутствующие части</th><th>Выделяемая часть</th><th>Текст задачи</th></tr>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>ВИ, ОП, НО, Пр, ФД</td><td>Зд</td><td>«Счастливый двоечник».
В интервале от 0 до 1024 подсчитайте количество чисел, двоичная запись которых симметрична. Ведущие нули не учитывайте.
  </td></tr>
<tr><td rowspan="3">ВИ,
НО,
Зд</td><td>ОП</td><td>«Дополнение».
Заданы 2N целых чисел, каждое из интервала [0..10000]. Можно ли разбить их на N пар таким образом, чтобы суммы чисел каждой пары совпадали?</td></tr>
<tr><td>ФД-2</td><td><u>Входные данные:</u>
Во входном текстовом файле input.txt на первой строке записано число 2<i>N</i> (1<=<i>N</i><=10000); во второй строке – 2<i>N</i> чисел, подлежащих разбиению. Эти числа разделены пробелами.
<u>Входные данные:</u>
Вывести либо сумму чисел любой пары, если разбиение возможно, либо слово ‘No’, если разбиение на пары, отвечающие условию, невозможно.</td></tr>
<tr><td>Пр</td><td>Примеры:
<u>input.txt</u>
6
3 2 5 1 6 4
<u>output.txt</u>
7

<u>input.txt</u>
4
1 2 3 5
<u>output.txt</u>
No</td></tr>
</table>

они четко сформулированы), то можно заметить, что все они распадаются на группы:

ФД-1. Количество считываемых программой данных одинаково для любых тестов (размерность пространства входных данных является константой). Это могут быть любые тесты, длина которых задана изначально в тексте самой задачи. Для удобства понимания такие данные могут располагаться на разных строках, однако составленный из них вектор будет иметь одну и ту же длину для всех тестов данной задачи. Пример достаточно сложного, но, тем не менее, конечно-постоянного входного вектора можно найти в задаче «Гусеницы» (см. таблицу 1).

ФД-2. Размер пространства входных данных может быть различным для разных тестов, но конкретное количество входных элементов в текущем тесте (длина вектора входных данных) всегда в нем же самом и указывается. Примером могут служить входные данные для задачи «Дополнение» (см. таблицу 3).

ФД-3. Размерность пространства входных данных явно не определена, длина вектора входных данных заранее неизвестна. Более того, ни один тест не содержит явного указания на собственную длину. Примером могут служить входные данные для задачи «Поиск по ключевым словам–2» (см. таблицу 2).

В чистом виде описанные выше типы ФД-2 и ФД-3 встречаются весьма редко. Обычно эти типы комбинируют с первым. Возможна также их комбинация друг с дру-

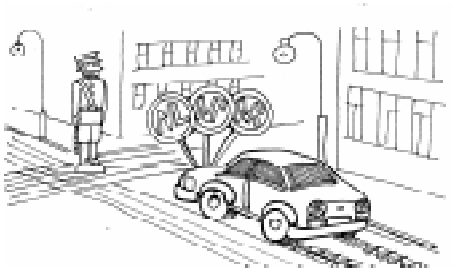
гом. Вот примеры задач, чьи входные данные служат примерами различных комбинаций:

ФД-12. «Возня с мелочью» (см. таблицу 1).

Замечание:

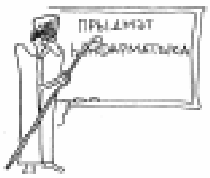
Если у задачи нет входных данных, то, вообще говоря, она не может претендовать на то, чтобы называться задачей по программированию, поскольку любой алгоритм (а в написании именно алгоритма и состоит решение задач по программированию), по определению, должен обладать свойством *массовости,* то есть решать не одну конкретную задачу, а целый класс аналогичных задач. Если же вариативность входных данных отсутствует, то такую задачу можно отнести к теоретическим и решать ее любым доступным способом, хоть на бумаге, хоть в уме. Можно даже написать программу, состоящую ровно из одной строки, которая выдает найденный ответ. Такой «алгоритм» будет, конечно, обладать еще одним базовым свойством любого алгоритма – *результативностью*, но не свойством *массовости.* С точки зрения автоматического тестирования, это не доставит никаких проблем: программа работает и выдает результат. Однако смысла в постановке таких задач все равно не много. Простейшая модификация требований всегда может сделать входные данные вариативными даже без усложнения задачи. Смотри, например, задачу «Счастливый двоечник» (таблица 3). В этом случае достаточно изменить жестко заданные границы рассмотрения [0..1024] на [0..2^N], где N – целое число от 1 до 10, и решить задачу без написания программы станет невозможным.

3. ОШИБКИ ПОСТАНОВЩИКОВ ЗАДАЧ



Не секрет, что разработчики условий задач и их решений – тоже люди, которым, как известно, свойственно ошибаться. Без ошибок авторов условий или тестов не обходятся даже такие уважаемые олимпиады, как Международный турнир ACM. И чем ниже квалификация, чем меньше опыт автора, тем большую вероятность ошибки стоит ожидать в его случае. Например, ошибки и недочеты составителей задач для олимпиад районного уровня стали чуть ли не правилом. Разумеется, трудно предусмотреть все возможные вопросы и недопонимания, которые может вызвать условие задачи. Однако стоит все же приложить максимум усилий, чтобы избежать если не всех, то хотя бы большей части ошибок и неточностей.

В тексте любой задачи наиболее важными частями являются третья, четвертая и пятая. При ошибках, допущенных в двух первых частях, на основании третьей, четвертой и пятой частей можно восстановить истину. В случае же ошибок или недоопределенностей в частях с третьей по пятую (особенно в четвертой) ошибки становятся источником в лучшем случае потока вопросов, адресованных составителю; в худшем – неправильной трактовки условия. Замечено, что такие задачи участниками олимпиад обычно не решаются (они требуют дополнительного анализа, разбора вариантов и, следовательно, слишком дороги с точки зрения временных затрат в условиях очных туров олимпиад). Предоставленные же решения в большинстве случаев не могут пройти тестирование по причине несовпадения допустимых множеств входных данных.



3.1. КЛАССИФИКАЦИЯ ЗАДАЧ С ТОЧКИ ЗРЕНИЯ КОРРЕКТНОСТИ УСЛОВИЯ

1. **Противоречивое условие** – в одной из самых важных частей допущена явная ошибка. Решение такой задачи невозможно, то есть область решений – пустое множество.

2. **Недоопределенное условие –** пропущено что-то важное, в результате чего становится возможным (или даже необходимым) свободный выбор.

В этом случае возможны два варианта решения:

а) любой (преимущественно – самый простой) вариант;

б) все возможные решения (на практике этот вариант очень редок).

Примером могут служить задачи «Дороги» и «Поиск по ключевым словам» (см. таблицу 2). В их условиях отсутствуют вторая (определения и терминология) и третья (описание проблемы) части, а в задаче «Дороги» – еще и четвертая (являющаяся собственно заданием) часть.

3. **Неоднозначное условие** (условие, допускающее неоднозначное толкование) – нечетко сформулированное, допускающее двоякую (еще хуже – многозначную) трактовку условия или задания. При автоматическом тестировании такой вариант дает особенно много псевдоошибок.

Пример: в задаче «Гусеницы» (см. таблицу 1) неоднозначным является случай с «очень маленькой гусеницей, которая стягивается в точку». Если точка содержится в отрезке, то что означает, что одна гусеница возит другую (пересечение – все точки меньшей гусеницы) или они играют в чехарду (пересечение – одна точка)?

4. **Полное условие**, совершенно лишенное ошибок, встречается крайне редко. Даже на самых престижных международных олимпиадах зачастую не обходится без каких-нибудь проблем с условиями задач, хотя тексты разрабатывают люди с достаточно высокой квалификацией. Понятно, что чем ниже уровень олимпиады и опыт и квалификация составителей задач, тем выше вероятность того, что в условиях заданий окажутся ошибки.

Андреева Татьяна Анатольевна,
аспирант Института систем информатики СО РАН,
г. Новосибирск.