Рейнгольд Григорий Борисович
педагог дополнительного образования
МБОУДОД «Центр детского технического творчества» г. Иркутска
г. Иркутск, Иркутская область
Рейнгольд Михаил Григорьевич
педагог дополнительного образования,
МБОУДОД «Центр детского технического творчества» г. Иркутска
г. Иркутск, Иркутская область

ПРОБЛЕМЫ, ВОЗНИКАЮЩИЕ ПРИ ВЫВЕДЕНИИ ЛИНЕЙНЫХ ФОРМУЛ, НЕОБХОДИМЫХ ПРИ РЕШЕНИИ ПРОГРАММИСТСКИХ ЗАДАЧ

Аннотация: *в статье раскрываются проблемы, возникающие при решении задач у юных программистов.*

При решении задач у юных программистов часто возникает необходимость вывода формулы. И в этом, на первый взгляд, несложном деле, часто возникают проблемы, ведь в школе этому не учат. Приведём пример такой задачи:

Имеется линейная железная дорога, состоящая из некоторого количества станций (задаётся во входных данных). Все станции пронумерованы подряд натуральными числами. Задаётся номер начальной станции и количество станций, которое надо проехать (положительное – вправо, отрицательное – влево). Если поезд приезжает в тупик, то разворачивается и едет в обратную сторону столько, сколько осталось.

Сделать программу, определяющую, где окажется поезд.

Примерные тесты:

№	*Входные данные*	*Выходные данные*
1	*10 5 3*	*8*
2	*10 5 8*	*7*
3	*10 5 -3*	*2*
4	*10 5 -8*	*5*

Заметим, что эта задача предлагается в нашем объединении при прохождении темы «Ветвление», ещё до изучения циклов, то есть предполагается, что количество разворотов не больше 1.

Вообще, эту задачу, как и многие другие, можно решить и без выведения конечной формулы, «по действиям». Но учиться выводить формулы всё равно нужно, так как во многих случаях это может привести к гораздо более эффективному решению задачи.

В нашем объединении «Юный программист» существует такой порядок решения задач:

1. Внимательно прочесть *условие задачи* и разобраться с *входными* и *выходными* данными.
2. Убедиться в правильности авторских *тестов.*
3. Составить несколько тестов, желательно на все частные случаи.
4. Составить *алгоритм*, написать и отладить *программу*.
5. Протестировать её.
6. В случае успешного прохождения *тестов* позвать учителя для сдачи *программы*. В противном случае найти ошибку на одном из предыдущих этапов работы и исправить её.

Но на практике этот порядок не всегда соблюдается, и случается, что наши воспитанники пропускают этап составления собственных тестов, либо делают это «спустя рукава», и приступают к программированию. И тут они становятся перед необходимостью выведения формулы.

Как мы советуем это делать:

1. Провести некоторое количество «математических экспериментов» и постараться подметить закономерность.
2. Выдвинуть гипотезу, согласно подмеченной закономерности.
3. С помощью достаточно большой серии «математических экспериментов» провести основательную проверку выдвинутой гипотезы, и в случае её неподтверждения вернуться к предыдущим пунктам, либо для уточнения старой гипотезы, либо для выдвижении новой.

Но когда предполагаемая формула должна получиться заведомо линейной, возникает искушение сократить объём своей работы. Ведь ученики знают, что прямая однозначно определяется двумя точками, и если выдвинутая успешно гипотеза проверена на двух примерах, то она должна быть признана верной. На первый взгляд, так и должно быть. Но бывают и исключения, как, например, в нашем случае.

И так, ученик поленился составлять свои собственные тесты, и положился лишь на те, что приложены к задаче. Он сразу понял, какие здесь имеются частные случаи:

1. При движении не возникает необходимости разворачиваться.
2. Разворот на последней станции.
3. Разворот на первой станции.

Очевидно, что случай 2 самый трудный, и мы рассмотрим лишь его. И так, после небольших раздумий, приходит на ум такая формула:

$$d = a + b – c \text{ (1)},$$

где a – количество станций, b – номер исходной станции, с – количество станций, которое нужно проехать.

Вроде бы, эта формула выдерживает проверку всеми авторскими тестами. Более того, формула подтверждается таким, например, тестом:

№	*Входные данные*	*Выходные данные*
1	*10 5 6*	*9*

И юный программист эту формулу использует. Но в процессе сдачи программы педагогу происходит вот такой казус. Педагог предлагает проверить программу по следующим тестам, предварительно доказав их правильность:

№	*Входные данные*	*Выходные данные*	*Выход, согласно формуле* $d = a + b - c$
1	*10 7 9*	*4*	*8*
2	*12 7 9*	*8*	*10*

Таким образом, доказана неправильность программы, а значит, и неправильность формулы (1).

Так в чём же дело? Почему линейная формула, прошедшая несколько проверок, оказалась неверной?

Давайте разберёмся. Для этого решим задачу «с вопросами по действиям», как это делается в начальной школе:

1) Сколько станций ехать с b–й до a–й?

$$a – b$$

2) Сколько останется ехать, если надо было проехать c, а проехал $a – b$?

$$c – (a – b) = c – a + b = b + c – a$$

3) Где окажется, если поедет с a–й станции $(b + c – a)$ станций в обратную сторону?

$$d = 2a – b – c$$

Теперь нам ясно, что правильная формула:

$$d = 2a – b – c \text{ (2)}$$

Осталось выяснить, почему неверная формула дала столько правильных прохождений программы. Сравним две формулы (1) и (2):

$d = a + b – c$
$d = 2a – b – c$
Отсюда:
$a + b – c = 2a – b – c$
$a + b = 2a – b$
$b = a – b$
$2b = a$

Так вот, в чём причина: в авторских тестах случайно (а, возможно, и нет!) оказалось, что номер начальной станции был равен половине количества станций! Но при хорошем составлении собственных тестов, юный программист не попался бы в эту ловушку.

Но остаётся ещё один вопрос: разве для проверки линейной функции не достаточно установить совпадение в 2–х точках? Достаточно, но только в случае, если это функция от одного аргумента! А в рассмотренном случае аргументов 3.

Кроме того, теперь стало ясно, что «подгонка» – не самый надёжный способ выведения формул, для этого необходимо провести более основательную работу.

Список литературы

1. Жилин Майк «Методы решения олимпиадных задач (Факультатив по информатике)» / http://ipg.h1.ru/fakultes/informatika/fakes01/zan/z02.html.
2. «Как стать чемпионом мира по программированию, или разбор полетов» / http://algorithm.ce.cctpu.edu.ru/member/victory_solution2.phtml.