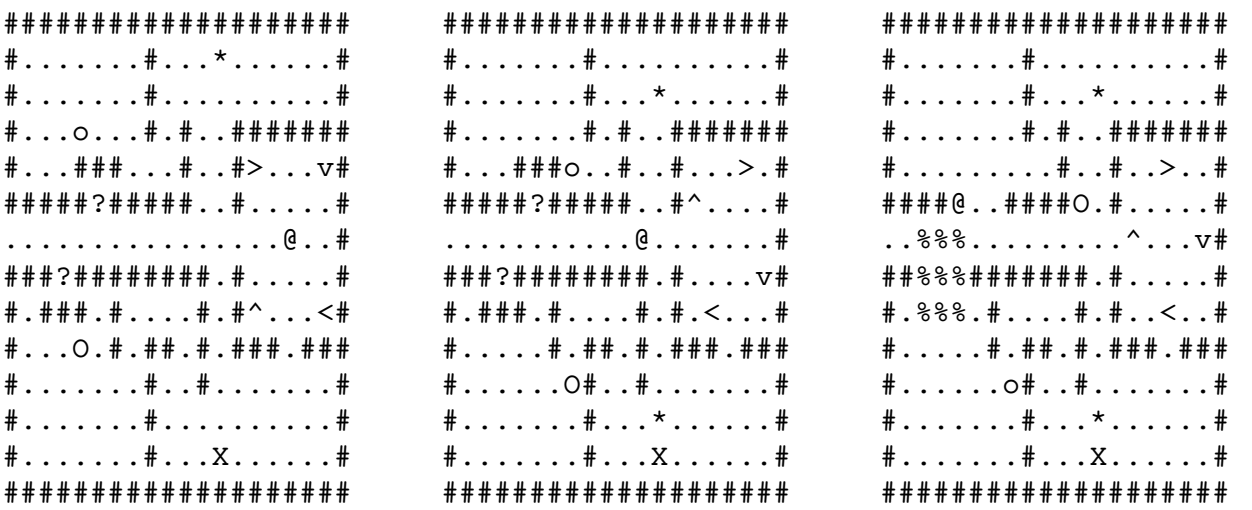
Компьютерная игра

Общая цель в этой тренировке — реализовать простую компьютерную игру.



Общее описание

Действие игры происходит на прямоугольном поле, состоящем из $\mathrm{rows} \times \mathrm{cols}$ клеток. Поле представлено в текстовом виде. Игрок управляет героем. Цель игрока — вывести героя на одну из крайних клеток поля. В этом ему могут помешать стены, монстры и устройства, находящиеся на поле. Игра происходит по ходам: сначала ходит герой, затем по очереди получают ход все другие объекты на поле. В начале каждого хода состояние игры полностью описывается символами на игровом поле.

Задача студента — реализовать эту игру. Эту задачу предлагается решать поэтапно, разбив на подзадачи. Каждая подзадача вводит в игру один новый тип объектов.

Общий протокол взаимодействия

Для удобства проверки игра оформлена в виде интерактивной задачи. Используются стандартные потоки ввода и вывода.

Входные данные начинаются со строки, в которой через пробел записаны два числа rows и cols ($3 \leqslant \mathrm{rows} \leqslant 24$, $3 \leqslant \mathrm{cols} \leqslant 80$). Далее следует rows строк по cols символов в каждой: содержимое поля. Возможные символы и связанные с ними ограничения подробно описаны ниже.

Для удобства тестирования стоит научиться при локальном тестировании вводить исходное содержимое поля из буфера обмена. Например, в консоли Windows это может делаться последовательностью нажатий «Alt+Space, E (Edit), P (Paste)», а в консоли Linux сочетанием «Ctrl+Shift+V».

Далее игра происходит по ходам: решение задачи работает как компьютерная игра, а студент (при локальном тестировании) или жюри (при проверке на сервере) действует за игрока.

В начале каждого хода следует вывести текущее положение. Для этого нужно вывести rows строк по cols символов в каждой: содержимое поля в текущий момент. После этого нужно вывести одну из следующих трёх строк:

- «`TURN` t `- YOU WON`», если герой находится на краю поля: в первой или последней строке поля, или в первом или последнем столбце поля.
- «`TURN` t `- GAME OVER`», если героя нет на поле (например, на его клетку наступил монстр, или это первая подзадача, где героя ещё вообще нет).
- «`TURN` t», если игра ещё может продолжаться.

Здесь t — номер хода. В начале самого первого хода этот номер равен нулю.

Поскольку задача проверяется как интерактивная, после вывода всей этой информации (а также завершающего перевода строки) следует очистить буфер вывода: например, это можно сделать командой `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Далее, если игра завершилась, решение должно сразу же корректно завершить свою работу. Если же игра не завершилась, игрок вводит своё следующее действие — строку с одним символом:

- «`w`» — движение вверх,
- «`a`» — движение влево,
- «`s`» — движение вниз,
- «`d`» — движение вправо,
- «` `» (пробел) — ожидание на месте.

После этого происходит ход игрока: если соседняя с героем клетка в указанном направлении пуста («`.`»), герой перемещается туда. В противном случае герой остаётся на месте.

Далее проверяется, не оказался ли герой на краю поля: в этом случае ход сразу заканчивается.

Если этого не случилось, происходит ход монстров. Строки поля рассматриваются сверху вниз. Клетки каждой строки рассматриваются по одному разу слева направо. Если в клетке находится монстр, он получает право действия, но только если до всех ходов монстров он находился в этой клетке.

Наконец, происходит фаза очистки: все оказавшиеся на краю поля объекты, кроме клеток пола, героя и стен («`.@#`»), исчезают с поля и заменяются на «`.`».

После всего этого начинается следующий ход.

Вместо того, чтобы вводить очередной ход, игрок может завершить ввод (конец файла при автоматической проверке, а при ручной — сочетание клавиш «Ctrl+Z» для Windows или «Ctrl+D» для Linux). В этом случае решение должно сразу же корректно завершить свою работу.

Подзадачи

Далее приведены предлагаемые подзадачи. Каждая подзадача вводит в игру один новый тип объектов, указанный в её заголовке, который может встречаться в этой подзадаче и во всех следующих. Правила и ограничения, указанные для этого типа объектов, также выполнены во всех следующих подзадачах.

С другой стороны, решение каждой следующей подзадачи включает в себя решения всех предыдущих подзадач. Решение каждой подзадачи можно послать на проверку в автоматическую проверяющую систему.

A. Floor — пол — «`.`».

Ввод состоит только из описания игрового поля, которое, в свою очередь, целиком состоит из клеток пола («`.`»). Следует вывести игровое поле, а после него — строку «`TURN 0 - GAME OVER`», так как героя на поле нет, и сразу завершить работу.

B. Hero — герой — «`@`».

Игровое поле содержит не более одной клетки с героем. Следует вывести игровое поле, строку состояния, а после этого реагировать на ходы игрока.

C. Wall — стена — «`#`».

Каждый раз, когда игрок пытается сделать ход в стену, он остаётся на месте.

D. Sentinel — часовой — «`^<v>`».

Четыре разных символа отображают четыре различных состояния часового. На каждом своём ходу часовой хочет переместиться в следующую клетку в одном из четырёх направлений: вверх («`^`»), влево («`<`»), вниз («`v`») или вправо («`>`»). Если соответствующая клетка пуста или содержит героя, происходит перемещение — это общий критерий успешного перемещения для этого и всех следующих монстров. А если перемещение не удалось, часовой остаётся на своей клетке, но поворачивается на 90 градусов **по часовой стрелке**.

Если часовой — или любой другой объект, кроме пола, героя и стены, — оказывается на краю поля, в фазу очистки перед началом следующего хода он исчезает с поля. Гарантируется, что в начальной позиции на краю поля могут быть только символы «`.@#`».

E. Seeker — искатель — «`*`».

Искатель смотрит по прямой в каждом из четырёх направлений: вверх, влево, вниз и вправо. Если в каком-то направлении он видит героя, то перемещается на соседнюю клетку в этом направлении. Если нет, искатель остаётся на месте. Правило видимости означает, что отрезок между монстром и героем либо вертикальный, либо горизонтальный, а все клетки отрезка между ними, если они есть — это клетки пола («`.`»).

F. Shooter — стрелок — «`X`».

Стрелок смотрит по прямой в каждом из четырёх направлений: вверх, влево, вниз и вправо. Если в каком-то направлении он видит героя, то создаёт искателя в соседней с собой клетке в этом направлении. Если нет, ничего не происходит. Созданный искатель не двигается на этом ходу: право действия получают только те объекты, которые были на поле в начале хода.

G. Blob — блоб — «`oO`».

Блоб движется медленно. А именно, если он находится в состоянии «`o`», то просто переходит в состояние «`O`». Из состояния «`O`» он переходит обратно в состояние «`o`», но может при этом переместиться в соседнюю клетку: для каждой из четырёх соседних клеток блоб вычисляет кратчайшее расстояние от неё до героя, если идти только по полу **и не посещать край поля**. Блоб выбирает ту соседнюю клетку, из которой расстояние до героя минимально, и перемещается туда. Если таких клеток несколько, блоб выбирает из них первое направление в своём списке предпочтений: вверх, влево, вниз, вправо.

Если в текущей конфигурации поля до героя не добраться, блоб переходит обратно в состояние «`o`», оставаясь на той же клетке. Как и другие монстры, блоб может перемещаться в клетку, только если она пуста или занята героем.

H. Mine — мина — «`?!%`».

Мина в состоянии «`?`» реагирует на приближение героя. А именно, если герой оказался в одной из клеток, соседних с миной **по стороне или диагонали**, она переходит в состояние «`!`». Иначе мина остаётся в состоянии «`?`».

Мина в состоянии «`!`» взрывается. А именно, девять клеток — клетка с миной и соседние с ней **по стороне или диагонали** — оказываются заняты взрывом («`%`»), независимо от того, что было в этих клетках раньше: пол, герой, стены или другие монстры.

Взрыв («`%`») исчезает и заменяется на «`.`».

I. Формально по этой подзадаче нужно послать код, решающий предыдущую подзадачу (`H`). Однако, чтобы получить зачётные баллы по этой подзадаче, нужно выполнить два дополнительных условия:

- Если при сборке решения указать версию `COMFORT` (то есть, в случае C или C++, компилировать с флагом `-DCOMFORT`, чтобы активировать код между `#ifdef COMFORT` и соответствующим `#endif`), процесс игры должен становиться удобнее: например, исходная позиция может читаться из файла, при нажатии клавиш может не требоваться нажатие Enter, вывод состояния может быть сделан более красивым... При этом решение может использовать дополнительные библиотеки, но должно остаться кроссплатформенным (должно быть возможно скомпилировать и запустить его в разных операционных системах). Важно не количество изменений, а именно разница в удобстве использования.
- Код решения должен быть читаемым и понятным.

Эти критерии трудно формализовать, поэтому фактические решения будут обсуждаться индивидуально с теми, кто будет сдавать эту подзадачу.

`J.` По этой подзадаче нужно послать не программу, а текстовый файл с входными данными для решения задачи `H`. Текстовый файл должен содержать исходную конфигурацию (строку с rows и cols и описание поля), а после неё — последовательность ходов, каждый ход на отдельной строке. Следует строго соблюдать формат ввода для остальных подзадач. В результате уровень должен быть пройден.

Цель в этой подзадаче — придумать начальную позицию для этой игры, из которой игру пройти можно, но не просто. Оцениваться она пока не будет. По истечении дедлайна по этой тренировке (4 ноября 2018 года, 23:59) у каждого студента, пославшего решение по этой подзадаче, будет взято последнее посланное решение, и исходные позиции (но не последовательности ходов) будут опубликованы. Можно будет получить дополнительные баллы как за прохождение игры в опубликованных позициях (вероятно, можно будет написать программу, которая поможет это сделать), так и за начальные позиции, которые немногим удастся пройти. Более конкретные правила появятся ближе к дедлайну.

Сложность

Сложность входных данных считается как суммарная сложность начальной позиции, умноженная на количество выведенных позиций. Сложность начальной позиции складывается из сложностей всех клеток позиции. Клетки начальной позиции имеют следующую сложность:

- «`.@#%`»: сложность 1.
- «`^<v>`»: сложность 5.
- «`?!`»: сложность 9.
- «`*X`»: сложность $\mathrm{rows} + \mathrm{cols}$.
- «`oO`»: сложность $\mathrm{rows} \cdot \mathrm{cols}$.

Гарантируется, что сложность входных данных в каждой подзадаче не превосходит 100 000. Это должно быть выполнено и для посылок по задаче `J`.

Тесты

Несколько тестов — по одному на каждую из основных подзадач — выложены по адресу `https://acm.math.spbu.ru/trains/181018_m18`.

В первые несколько дней тесты при автоматической проверке будут довольно слабыми. Затем будут добавлены тесты (в том числе, возможно, из задачи `J`), и все решения будут перетестированы.