Лекция №13. Игра «Жизнь». Ну и немного Ватор'а еще

Игра жизнь (ИЖ) — несложная модель роста популяции, скажем, микроорганизмов на некоторой области. Излишняя скученность или выход клеток из колонии ведут к вымиранию. Еще буквально капельку деталей можно почерпнуть в Тарасевиче.

Постановка. Моделируем на прямоугольном поле. Замкнутость поля не иммеет особого смысла, поэтому поле может как замыкаться по тору, так и обладать «глухими» границами. Состояние поля содержится в двумерном массиве целых чисел или в бинарном массиве. Этого достаточно, ведь у каждой ячейки есть лишь два состояния — живая и мертвая клетка. У каждой ячейки есть 8 нахожящихся рядом. Прямо и наискосок. В этих ячейках могут содержаться соседние клетки. И их количество определяет судьбу ныне обрабатываемой клетки.

- 0-1 сосед — клетка умирает или остается мертвой;
- 2 соседа — клетка остается в прежнем состоянии;
- 3 соседа — клетка появляется или остается живой;
- 4-8 соседей — клетка умирает или остается мертвой.

Подсказки Модель, как видно выше, крайне проста. Но есть пара нюансов. Время, как видно, дискретно. Клетки существуют в течение неких интервалов времени. Предположим, что у нас есть нынешнее состояние поля. Это карта всех ячеек, где видно, какие клетки живые, а какие мертвые. Для моделирования следующего шага необходимо использовать другой массив. Нынешнее состояние клеток влияет на то, какое распределение будет в дальнейшем. То есть в течение моделирования следующего шага в нынешний массив поля внесение изменений запрещено. Когда следующий шаг будет полностью смоделирован, он становится нынешним, а массив предыдущего шага становится ненужным, и его можно использовать для хранения следующего шага. То есть нужно, как минимум (как оптимум), два массива — для хранения нынешнего шага и для записи следующего. На следующем шаге они меняются местами. Для простоты моделирования можно использовать трехмерный массив с двумя двумерными слоями.

ИЖ распространена массово для обучения программированию и моделированию. Готовых программ море. В качестве референса можно использовать какой-нибудь готовый программный продукт. Мне искать было лень, поэтому вот: ИЖ.

ВАЖНО! Хотя как хотите. Если вы хоть отчасти планируете связать свою жизнь с кодингом. Будь то верстка, написание кода, управление командой разрабов, не важно что. Писать программы самому — превращать набор текстовых описаний в последовательность действий и затем реализовывать эту последовательность с помощью какого-то инстумента, это не просто важно, это системообразующе. Сейчас программы, которые требуются от вас, несложны, можно их {подмигивает} потырить у друга. Так ведь быстрее. Даже код-то понятен, вот суммирование, вот цикл. Все же ясно. Не так. Вы остаетесь на этапе «я могу написать код, если кто скажет, что в какой последовательности делать». То есть упускаете самое главное. В дальнейшем программы, которые от вас будут требоваться, будут разные. И легкие, и требующие некоторых усилий, в том числе гораздо больших, чем на нынешние программы.

Отвлеченная аналогия. Вы — турист-пешеходник. Учитесь ходить по горам-лесам (и ориентироваться там). Сейчас вы легко выходите из машины, делаете круг длиной 100 метров, и довольные собой едете обратно домой, полные впечатлений. Если вы сразу захотите на следующий шаг 20-дневный автономный поход на 250-километровый маршрут, то может вдруг оказаться, что шлепки — не лучшая обувь, что для ночевки нужны палатка и спальник. Что есть одно печенье 20 дней совершенно не комильфо. И много всякого другого. И все. Маршрут завален. Срочное отступление с самопозором. И Человек начинает копаться в себе и думать о себе невесть что.

Чтобы избегнуть подобных ситуаций, нужно развиваться постепенно. Делать круг 100 метров, потом 500, потом пять километров, на которых становится очевидной необходимость замены обуви на более удобную. Потом 20 километров, потом поход выходного дня с привалом и перекусом, где становятся очевидны минусы костра. Потом двух-трехдневная вылазка, где становится очевидным, что для ночевки с собой нужно довольно много всего тащить. А только потом 20-дневный маршрут. И на нем вы уже знаете все о себе и о своих инструментах.

Но конечно же, такая аналогия совершенно... СОВЕРШЕННО неверна, не так ли, ведь все мы программисты с рождения, и учиться и получать опыт совершенно не нужно.

Ладно, с давлением на совесть закончили. Я про что: считаю, что если вы сейчас сделаете лабораторные самостоятельно, пусть криво и косо, но с допиливанием и обсуждением недостатков, это, по меньшей мере, не будет лишним, а вообще — это опыт написания.

1 Алгоритм Ва-тор

Для хранения состояния требуется прямоугольное поле. Каждая особь моделируется с помощью набора изменяющихся параметров, в данном случае — счетчиков. То есть для сохранения состояния особи (контекста) требуется несколько целочисленных переменных. Кроме того, есть особи нескольких видов, у которых часть параметров пересекается, а часть нет, посему, если использовать одинаковые хранилища для всех особей, то нужно, чтобы они (хранилища) могли содержать контекст любой особи.

Полный ход (или поколение, как хотите) — это дать возможность всем особям на поле сделать ход. Второе поле, как в ИЖ, ввести здеь не удастся, особи непосредственно взаимодействуют с нынешними соседями и ходят в зависимости от того, что находится рядом. Результат каждого поколения довольно очевидно зависит от порядка хода.

Ход особи состоит в проверке всех условий и собственно ходе. То есть например: рыба; проверяем, нужно ли размножиться, если нет, то просто ход, если да, то оставляем на этом месте потомка; проверяем возможность хода — считаем свободные клетки вокруг, запоминаем, где они, выбираем случайную из них и ходим; если свободных нет, не ходим; акула: проверяем, нужно ли умереть от голода, если да, то уничтожаем ообь; проверяем, нужно ли размножиться, если нет, то просто ход, если да, то оставляем на этом месте потомка; проверяем соседние клетки на наличие рыб вокруг, считаем их, выбираем случайную и едим ее, ели нет рыб вокруг, то проверяем возможность хода — считаем свободные клетки вокруг, запоминаем, где они, выбираем случайную из них и ходим; если свободных нет, не ходим.

Системы хранения. Хранить данные об особях можно множеством разных способов. Я приведу два принципиальных, их реализация может варьироваться.

Первый: поле представляет собой двумерный массив структур (объектов, одномерных массивов, указателей на структуры или объекты). Каждая его ячейка содержит в себе параметры для понимания того, что в ней находится. Пример структуры: символьная переменная для хранения имени ячейки (вода "в рыба "р акула "а"), счетчик размножения (для воды — пустой, значение 0 или -1), счетчик голода (для воды и рыб пустой, зачение 0 или -1). Для простой модели этого хватит. Ход представляет собой вызов функции (цепочки функций), получающей координаты ячейки на поле и выполняющей на поле необходимые изменения. Поле проходится построчно и для каждой ячейки вызывается функция хода. Когда все ячейки пройдены, поколение считается завершенным. Плюсы: простота реализации, каждое поколение поле "пробегается"один раз, что благоприятно отражается

на скорости работы, легко моделируется сколько угодно разных видов. Минусы: нельзя без нескольких пробегов поля промоделировать ход видов по отдельности (сначала акулы, потом рыбы).

Второй: поле — двумерный массив ссылок на ячейки других массивов. Контекст рыб хранится в одномерном массиве структур "рыбы контекст акул в одномерном массиве "акулы". В структурах особей кроме их состояния хранятся координаты на поле, то есть из контекста особи получается ее положение на поле, а из соседей на поле получаются пустые или непустые ссылки на ячейки массивов "рыбы"и "акулы". В таком случае для пробега особей одного вида требуется пройти по одномерному массиву особей этого вида и промоделировать только их. Нет необходимости пробегать все поле с миллионом ячеек, если на нем всего 10 рыб. При пробеге массивов особей требуется либо искать первую свободную ячейку для записи новой особи, либо писать в конец, но делать периодическую дефрагментацию (сжатие) массива для избавления от пустот. Плюсы: удобство моделирования; минусы: в случае внедрения корма все равно нужно пробегать все поле, при дефрагментации массива сохранение взаимной связанности поле-массивы особей становится существенно сложнее.

Последовательность действий. Сначала, естественно, создаем все необходимые переменные, которые будут часто использоваться. Параметры могут задаваться как переменными, так и через переменные командной строки, и через define'ы.

После создания переменных их нужно инициализировать. Например, заполнить поле. Поле заполняется особями всех видов с какой-то вероятностью. Есть вариант раскидать по полю какое-то конкретное количество особей каждого вида, конечно, но при моделировании разных ситуаций, при каждом изменении размеров поля, придется переписывать вручную и количество раскидываемых особей. Оно вам надо? Поэтому пробегаемся по всему полю и с какой-то вероятностью заполняем его ячейки особями разных видов, например с вероятностью 0.2 появляется рыба, с вероятностью 0.05 — акула (не пользуйтесь этими вероятностями для маленьких полей). Каждая особь появляется со случайным набором параметров, но в позволенном диапазоне. То есть если значение счетчика голода 0 приводит к гибели, а при питании его значение становится 16, то случайное значение должно быть в пределах [0..16].

После заполнения поля запускаем ограниченный количеством поколений цикл (или бесконечный, ограниченный каким-то другим условием). Для моделирования поколения или пробегаем по всему полю и запускаем "ход"каждой ячейки, или пробегаем все массивы особей и ходим каждой. В случае отладочного запуска в конце поколения следует вывести состояние

поля с обозначением наличия видов особей в каждой ячейке. Это необходимо для понимания, правильно ли работают основные механики. После вывода поля нужна пауза для анализа состояния. В случае уверенности в работе модели в конце каждого поколения выводятся номер поколения и численности особей всех видов. Считаю, что численности стоит выводить в виде отношения количества особей к общему количеству ячеек на поле. Это доля ячеек поля, занятых особями этого вида.

По окончании моделирования выходим из программи (можно спросить, сколько еще пополений промоделировать и продолжить расчет).

Для тестов размеры поля порядка 20x20 вполне достаточны. Для отслеживания численностей и постоения их графиков поля с размерами меньше 200x200 приниматься не будут. Типичное количество поколений для графика 2000+.

Контрольные вопросы.

1. Сколько может быть соседей (клеток) у клетки в модели ИЖ?

2. Как замыкается поле в ИЖ?

3. Есть два массива A_1 и A_2, в A_2 сейчас лежит нынешнее состояние поля. Пытаемся получить состояние клетки с координатами i,j. Напишите последовательность действий для этого. Обозначение ячейки массива $A_k(i, j)$.