Темы курсовых работ

1. Объектно-ориентированное программирование (_Рыбы_) – ХН

1. Определите объект TFish - аквариумная рыбка. Рыбка имеет координаты, скорость, размер, цвет, направление движения. Методами объекта являются:

Init - устанавливает значения полей объекта и рисует рыбу на экране методом Draw.

Draw - рисует рыбу в виде уголка, с острием в точке Coord и направленного острием по ходу движения рыбы.

Look - проверяет несколько точек на линии движения рыбы. Если хоть одна из них отличается по цвету от воды, возвращается её цвет и расстояние до рыбы.

Run - перемещает рыбу в текущем направлении на расстояние, зависящее от текущей скорости рыбы. Иногда случайным образом меняет направление движения рыбы. Если рыба видит препятствие, направление движения меняется, пока препятствие не исчезнет из поля зрения рыбы.

2. Определите объект Taquarium, который является местом обитания рыб. Он представляет собой область экрана, наполненную водой. Рыбы живут в аквариуме, поэтому экземпляры объекта Tfish должны быть полями объекта Taquarium.

Методы:

Init - включает графический режим, заполняет аквариум водой, скалами и рыбами.

Run - организует бесконечный цикл, в котором выполняется метод Run всех обитателей аквариума.

Done - выключает графический режим.

3. Определите два объекта Tpike и Tkarp , которые наследуют объект Tfish. Оба они отличаются от Tfish тем, что по разному изображают себя на экране: Tpike - в виде зеленой стрелки, а Tkarp - в виде красного треугольника. Воспользуйтесь виртуальными методами. Для этого вернитесь к определению Tfish и откорректируйте его, сделав Draw пустым и виртуальным.

4. Объедините карпов и щук в две стаи. Стая - это связанный список рыб в динамической памяти. Для связи добавьте в объекты Tpike и Tkarp поле Next - указатель на следующую рыбу в стае. Сделайте аквариум не владельцем рыб, а двух стай и позвольте пользователю пополнять стаи, вводя рыб с клавиатуры.

5. Позвольте щукам проявить свой дурной характер и поедать карпов, как только они их увидят. Здесь возникнет проблема - установить, какого именно карпа видит щука. Она решается путем просмотра всей стаи карпов и поиска того, чьи координаты близки к координатам данной щуки. Найденный карп удаляется из стаи.

2. Задача Прима-Краскала (жадный алгоритм) - L

Дана плоская страна и в ней n городов. Нужно соединить все города телефонной связью так, чтобы общая длина телефонных линий была минимальной.

Уточнение задачи. В декартовой системе координат положение i-го города, i = 1,...,n, задано парой координат (x[i],y[i]). d[i,j] - декартово расстояние между i-ым городом и j-ым городом , j=1,...,n. В задаче речь идет о телефонной связи, т. е. подразумевается транзитивность связи: если i-й город связан с j-ым, а j-ый с k-ым, то i-й связан с k-ым. Подразумевается также, что телефонные линии могут разветвляться только на телефонной станции, а не в чистом поле. Наконец, требование минимальности (вместе с транзитивностью) означает, что в искомом решении не будет циклов. В терминах теории графов задача Прима-Краскала выглядит следующим образом:

Дан граф с n вершинами; длины ребер заданы матрицей (d[i,j]), i,j = 1,...,n. Найти остовное дерево минимальной длины. Как известно, дерево с n вершинами имеет n-1 ребер. Оказывается, каждое ребро надо выбирать жадно (лишь бы ни возникали циклы).

Алгоритм Прима-Краскала

(краткое описание)

В цикле n-1 раз делай:

выбрать самое короткое еще не выбранное ребро при условии, что оно не образует цикл с уже выбранными.

Выбранные таким образом ребра образуют искомое остовное дерево.

Напишите программу для решения задачи Прима-Краскала.

3. Шифр Цезаря – М

Юлий Цезарь был, якобы первым, кто придумал собственно шифр. Алфавит размещается на круге по часовой стрелке (при этом в русском алфавите, после А идет Б, а после Я - А). Для зашифровки буквы текста заменяются буквами, отстоящими по кругу на заданное число букв дальше по часовой стрелке. Если, скажем, сдвиг на 3, то вместо i-й используется (i+3)-я буква, например, вместо А пишется Г а вместо Я пишется В. При расшифровке наоборот берут букву на заданное число букв ближе, т. е. двигаясь против часовой стрелки.

Шифр Цезаря расшифровать легко. Известны вероятности букв p[i],i =1,2,...,n, в языке сообщения (n - число букв в алфавите). подсчитаем частоты букв f[i] в зашифрованном сообщении. Если оно не очень короткое, то f[i] должны сравнительно хорошо согласовываться с p[i]: f[i] = p[i-s] для некоторого сдвига s. Затем начнем делать перебор по сдвигам. Когда сдвиг не угадан, общее различие между p[i] и f[i+s] , равное D(s) = S | p[i] - f[i+s] | (суммирование берется по всем i от 1 до n), будет велико, а когда сдвиг угадан - мало. Минимизация D(s) по всем s = 1,2,...,n дает ключ к расшифровке кода Цезаря.

Напишите и испытайте программу взлома шифра Цезаря.

4. Игра Жизнь - Н

Это игра создана в 1970 г., ее автор - английский математик Дж. Конвей (Conway). В этой игре партнер не нужен - в неё можно играть одному. Возникающие в процессе игры ситуации очень похожи на реальные процессы, происходящие при зарождении, развитии и гибели колонии живых организмов.

Правила игры. Вообразите бесконечное поле, разделенное на клетки. На каждой клетке поля живет, рождается или погибает животное. Это зависит от условий Среды, т. е. от того, сколько соседей у него на ближайших восьми клетках (четырех по сторонам и четырех по углам).

Действуют три правила существования животных:

1. Каждое животное, у которого два или три соседа, живет и сохраняется до следующего поколения.
2. Животное погибает, если у него более нежели три соседа (от недостатка места), совсем нет соседей или только один сосед (от одиночества).
3. Когда рядом с какой-нибудь клеткой есть три животных (соседа), то на этой клетке рождается новое животное.
4. Важно понять, что животные погибают и рождаются одновременно. Они образуют одно поколение. За один ход в игре в соответствии с упомянутыми правилами осуществляется переход от одного поколения к другому.

Дж. Конвей рекомендует следующий способ осуществления ходов (при наличии клетчатой доски и косточек двух цветов):

1) начать с желаемой конфигурации (колонии животных), состоящей из черных косточек;
2) найти все косточки, которые должны *погибнуть*, и на каждую из них одеть по одной черной косточке;
3) найти все свободные клетки, на которых должно *родиться* животное, и положить на них по одной белой косточке;
4) удалить с доски всех *погибших* животных (т. е. столбики из двух косточек), а *новорожденных* (белые косточки) заменить черными. Выполнив эти операции, т. е. после

первого хода, получим второе поколение. Аналогичным образом происходит и все остальные ходы в игре. Так получаются все новые поколения.

Тема. Напишите программу, моделирующую колонию Жизни. Исходными данными служит начальное расположение животных (заданное пользователем или получаемое случайно - реализовать оба случая), а в качестве результата нужно получить вид сверху в графическом режиме всех поколений колонии.

Некоторые колонии разрастаются невероятным образом при весьма скромных начальных размерах. Есть другие колонии, которые медленно перемещаются по пустыне, переходя на все новые и новые территории. Ваша программа должна обрабатывать большие колонии без чрезмерной траты памяти или времени. Многократный просмотр большого массива для построения следующих поколений - это банальный подход; здесь хороший программист выбрал бы более экономичные структуры данных и алгоритмы. Вам, возможно, захочется испытать какой-либо метод, отслеживающий только занятые квадраты. В программе нельзя определить бесконечно большое поле. Должно хватать поля некоторой известной величины m*n. Что делать, если эволюция достигает границ поля? Один из возможных выходов - прервать эволюцию. Однако эволюция могла бы продолжаться, если бы мы устранили границы поля: соединили бы любые два противоположных края поля, затем концы полученного цилиндра. Полученная фигура, имеющая форму бублика, называется тор. Используйте этот подход в вашей программе (для этого достаточно более аккуратно определить соседей клетки, находящейся на краю поля). История колонии Жизнь зачаровывает, если её просматривать как фильм, но она будет еще более увлекательней, если предстанет в цвете. Каждой клетке при рождении может быть приписан некоторый цвет, определяемый, возможно, её поколением или генами, переданными ей родителями. Циклические, но при этом движущиеся колонии (а таких немало) великолепны в своем сверкающем многоцветном наряде.

5. Объектно-ориентированное программирование (_Солнечная система_) - ХН

Тема: объектно-ориентированное программирование астрономической модели солнечной системы. Модель описывает Солнце и планеты Меркурий, Венеру, Землю, Марс и их спутники. Программа работает следующим образом: на экране изображается Солнце и планеты со своими спутниками располагаются вокруг Солнца на своих астрономических местах. Планеты начинают вращаться вокруг Солнца по своим орбитам с правильным соотношением скоростей. В то же время спутники начинают вращаться вокруг своих планет по траекториям, складывающимся из двух вращательных движений: вращение планеты вокруг Солнца и вращение спутника вокруг планеты. Чтобы обобщить определения разных небесных, определите объект Tbody. Планеты и спутники так же, как и Солнце, - это небесные тела. Их надо определить как объекты-наследники от Tbody. Объекты-наследники должны содержать поля: 1)текущие координаты тела; 2) центр, вокруг которого тело вращается; 3) радиус орбиты; 4) список спутников; 5) скорость вращения; 6) размер; 7) цвет тела.

Вращение как планет, так и спутников вокруг центрального тела происходит по одним и тем же законом природы. Для планет телом, вокруг которого они вращаются, является Солнце, а для каждого спутника некоторая планета. Это движение для всех небесных тел можно определить одним методом - *Вращайся!*. Идея метода состоит в осуществлении движения тела наращиванием углового перемещения с шагом в 10 градусов. Перемещение каждого тела вычисляется в виде относительной величины, зависящей от значения его скорости. При каждом изменении угла вычисляются новые координаты положения тела. Каждая планета, начав вращаться должна запустить соответствующий метод вращения для своих спутников.

Относительные параметры для планет и спутников

название радиусскорость размер

Меркурий	58	0.416	3
Венера	108	0.416	5
Земля	150	0.1	6
Марс	228	0.053	4
Луна	15	1.3	2
Фобос	7	114.4	1
Деймос	12	30.4	1

6. Множество Мандельброта - H

В Книге рекордов Гиннеса самым сложным математическим объектом названо множество Мандельброта. Это плоское множество является ярким примером фрактала. Фракталы - это математические объекты, имеющие дробную размерность в отличие от традиционных геометрических фигур целой размерности (например, одномерных линий или двухмерных поверхностей). Фракталы - это нечто больше, чем математический курьез. Они дают чрезвычайно компактный способ описания объектов и процессов. Многие структуры обладают фундаментальным свойством геометрической регулярности, известной как инвариантность по отношению к масштабу, или *самоподобие*. Если рассматривать эти объекты в различном масштабе, то постоянно обнаруживаются одни и те же фундаментальные элементы. Эти повторяющиеся закономерности определяют дробную, или фрактальную, размерность структуры. В природе все фрактально: облака, изрезанная линия побережья, кромка листа, нервные и кровяные сосуды и т. д.

Множество Мандельброта описывает поведение динамического процесса, определенного на комплексных числах формулой

$z(n+1) = z(n)*z(n) + c.$ (1)

Опишем алгоритм построения окрашенного в черный цвет множества Мандельброта с окружением, раскрашенным в разные цвета. Для произвольного комплексного числа $c = x + i*y$ положим $z(0) = 0$ и устроим итерацию по формуле 1. Максимальное число итераций Max =150. Для последовательности z(n) имеются две возможности:

1. Числа становятся все большими и большими, стремясь к бесконечности.
2. Точки находятся и продолжают оставаться на расстоянии меньшим 2 от 0.

Так вот, множество Мандельброта - это множество тех чисел c, для которых выполняется вторая возможность. Граница множества сильно изрезанна. Причем под лупой она выглядит столь же изломанной, как и без нее. Она напоминает линию морского берега, многие естественные границы, которые становятся тем длиннее, чем более мелкий масштаб используется для измерения. Одной из характерных особенностей этой границы является её самоподобие. Если взглянуть на любой из её поворотов или заливов, то можно обнаружить, что одна и та же форма встречается в различных местах и имеет разные размеры.

В программе каждая точка (пиксел) экрана представляет соответствующее комплексное число c. Если число Max увеличить, то граница множества определится точнее, так как для некоторых точек с последовательности z(n) уйдут всё-таки на бесконечность. Все точки множества Мандельброта отметим черным цветом. Для всех других точек с соответствующая последовательность z(n) уходит на бесконечность, причем скорость ухода оценивается соответствующим цветом точки c, пропорциональным количеству итераций, достаточным для того, чтобы z(n)*z(n) стало большим 4. Всего используется цветовая палитра из 16 цветов и так как Max-1 значительно больше 15, то цвета периодически повторяются. Окраска внешности множества Мандельброта позволяет более точно увидеть границу множества.

Множество Мандельброта строится в прямоугольнике с координатами xmin= -2.25, xmax=0.75, ymin=-1.5, ymax=1.5 (т. е. пиксел с координатами (0,0) представляет комплексное число -2.25 + i*1.5). Программа должна позволять пользователю менять координаты вершин прямоугольника, чтобы можно было изобразить в увеличенном виде отдельные

фрагменты множества Мандельброта. Так как множество Мандельброта строится достаточно медленно, то необходимо предусмотреть возможность записи создаваемого изображения в файл. Такой файл просто хранит цвета всех пикселей экрана. Необходимо так же вспомогательная программа, которая сразу же извлечет предварительно созданное изображение из файла и изобразит на экране. Динамическое изменение цветовой палитры в цикле позволит получить более красочные изображения множества Мандельброта.

7. Перенос слов - M

Как показывают многочисленные эксперименты, разбиение русского слова на части для переноса с одной строки на другую с большой вероятностью выполняются правильно, если пользоваться следующими простыми приемами:

1) Две идущие подряд гласные можно разделить, если первой из них предшествует согласная, а за второй идет хотя бы одна буква (буква й при этом рассматривается вместе с предшествующей гласной как единое целое).

2) Две идущие подряд согласные можно разделить, если первой из них предшествует гласная, а в той части слова, которая идет за второй согласной, имеется хотя бы одна гласная (буквы ь, ъ вместе с предшествующей согласной рассматриваются как единое целое).

3) Если не удается применить пункты 1), 2), то следует попытаться разбить слово так, чтобы первая часть содержала более чем одну букву и оканчивалась на гласную, а вторая содержала хотя бы одну гласную. Вероятность правильного разбиения увеличивается, если предварительно воспользоваться хотя бы неполным списком приставок, содержащих гласные, и попытаться прежде всего выделить из слова такую приставку.

Дан текст на русском языке. Выполнить форматирование его строк по длине с помощью на переноса слов.

8. Мультфильм - L

В рисованных мультфильмах иллюзия движения создается последовательной сменой кадров, каждый из которых фиксирует очередное положение движущегося объекта.

Используя этот принцип, получить мультфильм, показывающий: а) идущего человечка; б) бегущего человечка; в) человечка, выполняющего приседания; г) человечка, выполняющего сигнализацию флажком.

9. Морской бой - L

На поле 10 на 10 позиций стоят невидимые вражеские корабли: 4 корабля по одной клетке, три корабля по 2 клетки, 2 корабля по 3 клетки, 1 корабль в 4 клетки. Позиции указываются русскими буквами от А до К (по строкам) и цифрами от 1 до 10 (по столбцам). Конфигурация и положение кораблей на поле выбираются с помощью датчика случайных чисел. Если клетка корабля угадана играющим верно, она отмечается крестиком; в противном случае точкой.

Написать программу для игры против компьютера в односторонний *морской бой*.

10. Линейные фракталы - M

Фракталы - это математические объекты, имеющие дробную размерность в отличие от традиционных геометрических фигур целой размерности (например, одномерных линий или двухмерных поверхностей). Фракталы - это нечто больше, чем математический курьез. Они дают чрезвычайно компактный способ описания объектов и процессов. Многие структуры обладают фундаментальным свойством геометрической регулярности, известной как инвариантность по отношению к масштабу, или *самоподобие*. Если рассматривать эти объекты в различном масштабе, то постоянно обнаруживаются одни и те же фундаментальные элементы. Эти повторяющиеся закономерности определяют дробную, или

фрактальную, размерность структуры. В природе все фрактально: облака, изрезанная линия побережья, кромка листа, нервные и кровяные сосуды и т. д.

Некоторые из фракталов называются линейными, потому что строятся с помощью линейных (аффинных) преобразований плоскости. Такие преобразования изображение перемещают, сжимают, отражают, вращают и трансформируют произвольным образом при условии, что прямые линии на изображении остаются прямыми после преобразования. Описывая фракталы посредством аффинных преобразований, мы можем значительно уменьшить количество данных, необходимых для передачи изображения по линиям связи или для хранения его в памяти компьютера. Сложная форма, подобная форме листа папоротника, может быть полностью описана линейным алгоритмом, основанным лишь на 28 числовых параметрах. Заметим, что представление того же листа в точечном виде, как телевизионное изображение, требует несколько сотен тысяч числовых величин.

Алгоритм построения линейных фракталов.

Аффинное преобразование евклидовых координат задается с помощью двух линейных уравнений:

$$x\_ = a\,x + b\,y + e$$
$$y\_ = c\,x + d\,y + f$$

Для построения фрактала поступают следующим образом:

Выбирается несколько афинных преобразований и каждому преобразованию сопоставляется некоторая вероятность его использования. Вид аффинных преобразований, их количество и вероятность зависят от конкретного фрактала.

Так, например, для листа папоротника используются 4 преобразования:

a	b	c	d	e	f	p
0.00	0.00	0.00	0.16	0.00	0.00	0.01
0.85	0.04	-0.04	0.85	0.00	1.60	0.85
0.20	-0.26	0.23	0.22	0.00	1.60	0.07
0.15	0.28	0.26	0.24	0.00	0.44	0.07

Берется начальная точка с координатами x = 0 и y = 0. Затем применяется одно из данных преобразований координат для определения новых значений и, какое именно преобразование применить определяется случайным образом в соответствии с заданной вероятностью p. Полученная точка изображается на плоскости цветом связанным с примененным преобразованием. Этот процесс повторяется достаточное число раз, пока не построится достаточно реалистическое изображение. Чтобы изображение удачно было расположено на экране нужно задать логический экран - в случае листа папоротника :

по оси x от -4 до 0

по оси y от 6 до 10

Замена значений коэффициентов b и c для листа папоротника во втором уравнении соответственно на 0.06 и - 0.06 увеличивают кривизну стебля папоротника. Замена их на 0.02 и -0.02 - уменьшают кривизну. Если исключить первое уравнение, то пропадает стебель. Программа должна строить несколько фракталов.

11. Одномерные клеточные автоматы - M

Рассмотрим клетки, расположенные вдоль прямой. Каждая из клеток может иметь состояние 0 или 1. На каждом шаге по времени новое состояние клетки вычисляется из старого состояния клетки и состояний её соседей. На экране каждый горизонтальный ряд показывает состояния клеток с помощью соответствующего цвета в очередной момент времени. Временная ось идет сверху вниз и каждый ряд вычисляется на основе

предыдущего ряда. Различные классы линейных клеточных автоматов могут быть определены в зависимости от того, как много различных значений (состояний) клетка может иметь (k), и как много соседей на каждой стороне клетки используются при вычислении нового состояния ®. Правила для определения следующего состояния клетки используют сумму состояний самой клетки и клеток-соседей. Сумма состояний отображается правилом на новое состояние клетки. Если k=2, r=1, то только ближайшие соседи клетки используются. Если каждая из клеток имеет состояние 1, то максимальная сумма 1+1+1 равна 3 и сумма может меняться от 0 до 3, т. е. имеем четыре величины. Поэтому правило преобразования можно задать с помощью строки из четырех двузначных цифр. Например, правило 1010, начиная с правой цифры, задает отображение суммы 0 -> 0, 1 -> 1, 2 -> 0, 3 -> 1 (таким образом, если сумма равна 2, то новое состояние равно 0). Для k=2 и r=2 каждая клетка (на экране она изображается пикселем) новое значение получает в зависимости от значений двух соседних клеток с каждой стороны. Соответствующие правила должны иметь 6 двузначных цифр. Если k=3 и r=1, то каждая клетка может иметь значение 0, 1 или 2. Учитывается только единственный сосед с каждой стороны, поэтому результат преобразования задается правилом из 7 цифр и т. д.

Напишите программу для построения линейных клеточных автоматов следующих классов kr = 21, 31, 41, 22, 32, 42.

k	r	количество цифр в правиле	пример правила
2	1	4	1010
3	1	7	1211001
4	1	10	3311100320
2	2	6	0110110
3	2	11	21212002010
4	2	16	2300331230331001

Начальная строка задается либо случайным образом, либо пользователем.

Как ведут себя клеточные линейные автоматы? Происходит ли переход к однородному состоянию независимо от начальных данных? Существуют ли классы автоматов с локализованными стационарными или периодическими конфигурациями? Есть ли автоматы с хаотическим временным поведением? Есть ли автоматы, которые ведут себя по разному при различных начальных данных? Необходимо ответить на эти вопросы. Смотрите программу-аналог Fractint.

12. Инженерный калькулятор - М

Написать программу, которая бы вычисляла арифметическое выражение, введенное с клавиатуры. Арифметическое выражение может содержать числа (в том числе и в экспоненциальной форме, например 1.2e-10), символы арифметических операций, скобки, функции синуса, косинуса, тангенса, логарифма, экспоненты.

Разбор арифметического выражения рекомендуется проводить следующим образом.

Создается рекурсивная функция gettoken(). В зависимости от текущего символа входной строки она производит следующие действия:

+, -, /, * → gettoken(); выполнить операцию

цифра → положить в стек цифру

(→ gettoken(); пропустить)

символ → выяснить что за функция; gettoken(); вычислить значение

13. Баллистическая игра – М

В разных концах экрана расположены две баллисты, принадлежащие разным игрокам. Игроки ходят по очереди. Ход заключается в выборе массы камня для баллисты, начальной скорости камня и угла между вектором начальной скорости и горизонтом. По этим данным

программа должна произвести расчет траектории полета камня и смоделировать полет на экране.

После первого игрока ходит второй и т.д. Игра заканчивается, когда один из камней попадет в баллисту противника.

14. Программа для алгебраических вычислений – M

Объекты, с которыми вы будете работать, - это многочлены от нескольких переменных, представленных в символьном виде с вещественными коэффициентами. Многочлены должны изображаться как арифметические выражения, так умножение изображается знаком *, а возведение в степень - знаком ^. Для манипуляций с многочленами нужны некоторые команды, чтобы пользователь мог получать ответы на вопросы, на которые не удается ответить с помощью традиционных языков программирования. Для этого вам понадобится обозначать многочлены идентификаторами. Команды выполняют некоторые операции над своими операндами и помещают результат в качестве значения некоторого имени многочлена. Список команд.

1. Ввести многочлен и записать его под некоторым именем.

2. Образовать алгебраическую сумму (разность, произведение) двух многочленов и записать полученный многочлен под некоторым именем.

3. Возвести данный многочлен в целую степень и результат записать под некоторым именем.

4. Заменить каждое вхождение некоторой переменной в многочлене на данный многочлен и результат записать под некоторым именем.

5. Вычислить производную многочлена по переменной и результат записать под некоторым именем.

6. Напечатать данный многочлен.

Многочлен представлять в виде суммы членов, включающих только операции умножения и возведения в степень. В каждом таком одночлене все константы перемножены и образуют числовой коэффициент (первый сомножитель), переменные упорядочены по алфавиту и все степени одной переменной объединены так, что каждая переменная встречается лишь один раз. Следует приводить подобные члены, т. е. объединять одночлены, имеющие одинаковые наборы переменных и степеней, с соответствующим изменением коэффициентов. Для представления многочленов в памяти используйте списковые структуры.

15. Часы с кукушкой – L

Составить программу, моделирующую работу стрелочных часов с кукушкой. У часов должен быть циферблат с тремя стрелками, маятник и окошко. Каждый час в окошке должна появляться кукушка. Она должна появиться столько раз, сколько часов показывают часы.

16. Волчий остров - H

Волчий остров размером 20x20 заселен дикими кроликами, волками и волчицами. Имеется по нескольку представителей каждого вида. Кролики довольно глупы: в каждый момент времени они с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью 0,2 превращается в двух кроликов. Каждая волчица передвигается случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым она охотится. Если волчица и кролик оказываются в одном квадрате, волчица съедает кролика и получает одно очко. В противном случае она теряет 0,1 очка. Волки и волчицы с нулевым количеством очков умирают.

В начальный момент времени все волки и волчицы имеют 1 очко. Волк ведет себя подобно волчице до тех пор, пока в соседних квадратах не исчезнут все кролики; тогда, если волчица находится в одном из восьми близлежащих квадратов, волк гонится за ней. Если

волк и волчица окажутся в одном квадрате и там нет кролика, которого нужно съесть, они производят потомство случайного пола.

Запрограммировать предполагаемую экологическую модель и понаблюдать за изменением популяции в течение некоторого периода времени.

17. Карточная игра - L

Составить программу, которая раздает игральные карты заданному количеству игроков (одним из игроков является человек, за остальных играет компьютер) и моделирует игру в «дурака». Компьютерная программа играет случайным образом, без анализа уже вышедших карт.

18. Крестики-нолики-M

Составить программу, позволяющую играть на бесконечном поле в «крестики-нолики»:

а) игроку с компьютером;

б) двум игрокам.

Если в качестве игрока выступает компьютер, программа делает первый ход. Делая очередной ход, программа анализирует ситуацию, рассчитывая возможные ходы противника вперед на 1 — 2 хода, и в результате проведенного анализа поступает оптимальным образом.

19. Быки и коровы - M

Составить программу, позволяющую играть в «Быки и коровы»:

а) игроку с компьютером;

б) двум игрокам.

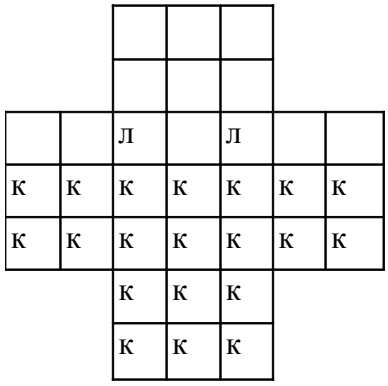
Суть игры в следующем. Каждый из противников задумывает четырехзначное число, все цифры которого различны (первая цифра числа отлична от нуля). Необходимо разгадать задуманное число. Выигрывает тот, кто отгадает первый.

Противники по очереди называют друг другу числа и сообщают о количестве «быков» и «коров» в названном числе («бык» — цифра есть в записи задуманного числа и стоит в той же позиции, что и в задуманном числе; «корова» — цифра есть в записи задуманного числа, но не стоит в той же позиции, что и в задуманном числе).

Например, если задумано число 3275 и названо число 1234, получаем в названном числе одного «быка» и одну «корову». Очевидно, что число отгадано в том случае, если имеем 4 «быка».

20. Две лисы и 20 кур - H

На поле, указанной на рисунке формы находятся две лисы и 20 кур. Куры могут перемещаться на один шаг вверх, влево или вправо, но не назад и не по диагонали. Лисы также могут перемещаться только на один шаг (вверх, вниз, влево и вправо).



Лиса может съесть курицу, как в игре в шашки: если в горизонтальном или вертикальном направлении за курицей на один шаг следует свободное поле, то лиса перепрыгивает через курицу и берет ее.

Лисы всегда обязаны есть, и, когда у них есть выбор, они обязаны осуществлять «наиболее длинное поедание». Если два приема пищи имеют одинаковую длину, осуществляется один из них — по выбору лисы.

Составить программу, которая играет за лис (лисы перемещаются вверх, вниз и в стороны, но не по диагонали). Игрок перемещает кур (куры могут двигаться вверх и в стороны, но не назад). Партнеры играют по очереди, причем куры начинают. Они

выигрывают партию, если девяти из них удается занять 9 полей, образующих верхний квадрат поля.

Начальное положение кур и лис изображено на рисунке.

Лисы выигрывают, если им удается съесть 12 кур, так как тогда оставшихся кур недостаточно, чтобы занять 9 верхних полей.

21. Игра в слова - L

Составить программу, позволяющую компьютеру и человеку играть в слова. Предварительно программа объясняет правила игры и позволяет уточнить их в любой момент. Тематикой игры могут быть по выбору города, животные, растения и т. д. Тема выбирается из предложенных компьютером (не менее 3).

22. Морской бой - L

Составить программу для игры в морской бой игрока с компьютером. Программа должна позволять расставлять корабли на поле 10x10, контролировать правильность их расстановки, давать противнику возможность поочередно делать ходы и выдавать соответствующие информационные сообщения. Программа должна анализировать предыдущие ходы и следующий делать на основе проведенного анализа.

23. Графики - L

Составить программу, которая предлагает пользователю некоторый список функций для построения графиков (например, y = ax2 + bx + c, y = ax + b и т.д. — до 10 наименований). После выбора соответствующей функции, задания коэффициентов и отрезка, на котором выполняется построение, программа строит указанный график. Затем значение коэффициентов и положение графика можно менять (например, с помощью клавиш управления курсором), после чего график перестраивается и записывается обновленное уравнение соответствующей кривой.

24. Задача об инфекции стригущего лишая - M

Промоделировать процесс распространения инфекции — стригущего лишая по участку кожи размером N*N клеток (N – нечетное). Предполагается, что исходной зараженной клеткой кожи является центральная. В каждый интервал времени пораженная инфекцией клетка может с вероятностью 0,5 заразить любую из соседних здоровых клеток. По прошествии шести единиц времени зараженная клетка становится невосприимчивой к инфекции. Возникший иммунитет действует в течение последующих четырех единиц времени, а затем клетка оказывается здоровой. В ходе моделирования описанного процесса выдавать текущее состояние моделируемого участка кожи в каждом интервале времени отмечая зараженные, невосприимчивые к инфекции и здоровые клетки.

25-34. Создать информационно-поисковую систему – M

Информационно-поисковая система должна являться клиент-сервер приложением БД (созданным по архитектуре клиент-сервер либо файл-сервер).

Она должна обеспечивать возможность ввода, редактирования, удаления, поиска, фильтрации данных, а так же вывода отчетов по заданным критериям (не менее 3х отчетов)

25. ИПС ”Библиотека”
26. ИПС ”Проф колледж”
27. ИПС ”Расписание занятий”
28. ИПС ”Успеваемость ”
29. ИПС ”Склад оптовой базы”
30. ИПС ”Поликлиника”
31. ИПС “Продажа видео и аудио продукции”

32. ИПС “ПО и состав ПК организации”
33. ИПС ”Каталог музыкальных треков”
34. ИПС ”Ветеринарная клиника”