**ЛР1(14/15)**

Реализовать следующую программу:

Создать класс Поле, хранящий двумерный массив символов – игровое поле. Реализовать **случайное заполнение** поля различными символами, например ‘\_’ для пустой клетки, ‘0’ для препятствия, ‘\*’ ‘!’ для специальных плиток.

Создать класс Юнит, хранящий координаты юнита и различные доп. характеристики. Сделать класс Юнит **абстрактным**, объявить в нём абстрактный метод движения, после **унаследовать** от него два подкласса и в них определить конкретные реализации этого метода. Например, класс Unit и наследующие ему Rook и Bishop, передвигающиеся как соответствующие шахматные фигуры.

Альтернативно можете сделать абстрактный класс поле с разными наследниками – прямоугольны и круглым полем.

Также реализуйте взаимодействие с разными клетками. Например, при перемещении на клетку ‘$’ у юнита добавляется счёт, а на клетке ‘\*’ - снижается количество жизней (если оно достигает нуля - программа досрочно прекращается). Можете придумать свою вариацию работы с плитками.

Перемещение юнита должно реализовываться вводом команд с клавиатуры. Каждый ход нужно перерисовывать поле, со специальным символом на месте юнита. Придумайте свои вариации наследных классов.  
  
Структурно в программе должно быть не меньше 3 файлов - один исполнительный, с управляющей логикой программы, 1 - заголовочный для класса Поля, описывающий его логику, 2 - заголовочный для класса Юнит и его подклассов.

**ЛР2(15/20)**

Реализовать на основе первой лабораторной следующую программу:  
  
Симулятор гладиаторских боёв.

Оставить класс Поле и класс Юнит, а также наследные классы.  
  
Реализовать для Юнита и его наследников метод поиска пути (алгоритмом А\*).  
Реализовать разный поиск пути для разных наследников, например через разное взвешивание клеток поля, или разное целеполагание юнитов.   
  
Добавить в класс Юнит поле: указатель на Юнит - ссылка на врага. Прописать для юнита автоматический поиск пути к врагу и правила ведения боя.  
В программе поместить двух юнитов разных классов на поле и запустить цикл до победы одного из юнитов или достижения максимального количества ходов (например 100)  
В цикле юниты по очереди делают 1 шаг в соответствии с алгоритмом поиска пути и, если могут, наносят друг другу удар.

Структурно в программе должно быть не меньше 3 файлов - один исполнительный, с управляющей логикой программы, 1 - заголовочный для класса Поля, описывающий его логику, 2 - заголовочный для класса Юнит и его подклассов.

**ЛР3(5/15) какой позор, делал лабу за пару часов до дедлайна без понятия как ее сделать**

Оценка сложности алгоритмов

Реализовать примитивную работу базы данных (массив указателей на произвольную структуру с id номером).

Для базы данных реализовать функции поиска (по id), добавления (случайного), удаления (по id) элемента.

Также реализовать два алгоритма сортировки для этой базы данных - сортировки пузырьком и быстрой сортировки. Алгоритм быстрой сортировки можете подключить из библиотеки или взять из интернета.

Алгоритмы поиска, удаления и добавления элементов должны работать по разному,  в случае если база данных отсортированна. Например поиск в неотсортированной базе данных реализуется полным перебором, в отсортированной - бинарным поиском.

Создайте статическую переменную для подсчёта количества всех итераций циклом при работе с базой данных.

Реализуйте также стимулятор работы базы данных, берущий на вход начальное количество записей в базе данных, количество симулируемых запросов, соотнощение запросов по типам (добавление, удаление, поиск).

Прогонять стимулятор для двух наборов данных, для каждого из наборов сравнить количество получившийся итераций для разных конфигураций - не сортированной базы данных, сортированной пузырьком и быстрой сортировкой.

**ЛР4(20/20)**

Алгоритмы машинного обучения

Реализовать базовый алгоритм машинного обучения - эволюционный алгоритм.  
  
Объявите эталонную функцию в виде y(x) = k1\*x^3 + k2\*x^2 + k3\*x + k4  
Задайте величины коэффициентов самостоятельно, прямо в программе.  
  
Далее реализуйте программу, находящую параметры целевой функции по косвенному и по прямому признаку.  
Для этого задавайте с клавиатуры начальные параметры k1, k2, k3, k4, максимальное случайное отклонение параметра d и количество потомков в одном поколении N.  
  
Далее по начальным параметрам сгенерируйте первое поколение потомков, где каждый параметр случайно отличается от родительского не более чем на d. Проведите оценку потомков по выбранному признаку, выберете потомка, наиболее близкого к эталонной функции, как родителя для следующего поколения.  
Повторяйте генерацию поколений пока позитивное изменение не сократиться до пренебрежимо малых значений.  
  
В качества косвенного признака сравнивайте определенный интеграл от -100 до 100 у эталонной и проверяемой функции.  
В качества прямого признака считайте расстояние между графиками функций на интервале от -100 до 100.  
  
В программе выбирайте, по какому из признаков будет проходить проверка.  
  
Программа должна выводить значение признаков родителя каждого поколения.