Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №3 дисциплины

«Теория автоматов»

Выполнил студент группы ИВТ-22 /Крючков И. С/ Проверил /Мельцов В. Ю./

Киров 2021

# Задание: разработать бота для игры «Морской бой»

**Словесное описание алгоритма:**

Для работы алгоритмов: поле делится на 4 равные части (5x5) и в каждом сете сохраняется информация в какую из частей соперник стрелял позже, а в какую раньше. Так же сохраняется порядковый номер и координаты выстрелов соперника.

При анализе берется среднее арифметическое этих данных.

По ходу игры для каждого сета сохраняется расположение подбитых кораблей соперника.

**Расстановка кораблей.**

Для первого сета берется заранее указанная расстановка. Для всех последующих сетов расстановка кораблей генерируется при запросе сервером карты, в зависимости от координат выстрелов соперника во всех предыдущих сетах.

4х-палубный: каждый крайний ряд (столбец) с каждой стороны поля (у стен) делится пополам (итого 8 частей). Из этих частей выбирается та, в которую соперник стрелял меньше всех и позже всех. Корабль устанавливается в определенные для полученной части координаты.

3х-палубные: анализируется каждый крайний ряд (столбец) с каждой стороны поля (у стен) и выбираются два таких, в которые соперник стрелял меньше всего и позже всего. Затем в полученных рядах происходит поиск и установка корабля в первую свободную позицию.

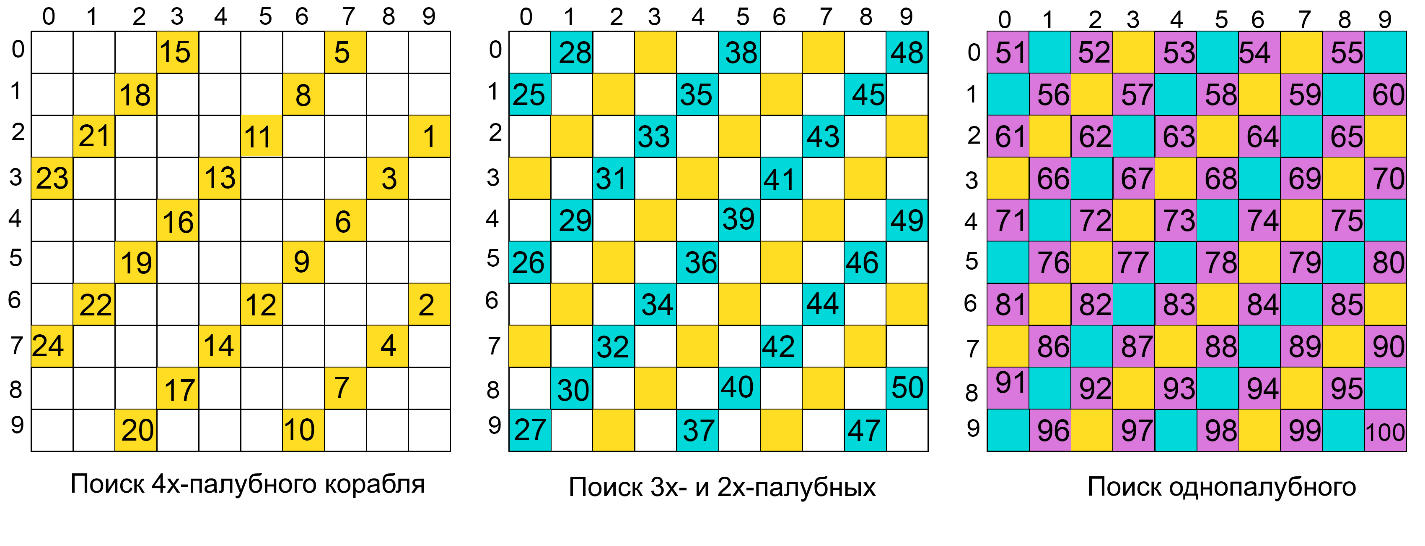
2х-палубные: выбирается сектор поля, в который соперник стрелял реже всего. В этом секторе корабль устанавливается в свободные координаты, в которые соперник стрелял позже всех. После установки одного корабля в данной секторе, а так же при невозможности установить корабль, происходит переход в следующий по возрастанию сектор.

1-палубные: если после 5 сетов на поле обнаружены координаты, в которые соперник ни разу не стрелял ни в одном из сетов, корабли, при возможности устанавливаются в данные координаты. В других случаях корабли устанавливаются по алгоритму для 2х-палубных кораблей.

**Стрельба**

В первом сете происходит стрельба по заранее определенным (стандартным) шаблонам.

В следующих сетах: если сохранена одна карта кораблей соперника (идёт второй сет), то выбираются координаты подбитых на предыдущем сете кораблей (если корабли не были подбиты, этап пропускается и считается, что координаты выстрела не были определены), при условии, что по этим координатам раннее не было выстрела и они не входят в область вокруг убитого корабля соперника (разрешенные координаты выстрелов). Если сохранено от двух карт соперника, для каждой карты вычисляется процент совпадения расположения кораблей на ней с расположением кораблей в текущем сете. Если есть хотя бы одна карта с процентом совпадения больше нуля, стрельба на текущем ходу происходит по разрешенным координатам из данной карты. Если определилось совпадение с несколькими картами, стрельба производится по карте с бОльшим процентом совпадения. Если таким образом не удалось определить координаты выстрела, выстрел производится по наиболее часто используемым позициям соперника из сохраненных карт. Если в ходе алгоритма не удалось определить координаты выстрела, они берутся из стандартных шаблонов. Если при стрельбе по картам соперника было больше 5 промахов (без учета добивания), дальнейшая стрельба производится по стандартным шаблонам.

****

**Добивание**

Если подбита одна клетка (опорная клетка): выполняется перебор каждого «живого» корабля соперника, начиная от двухпалубного, на возможность установки его в каждом направлении вокруг опорной клетки (слева, справа, сверху, снизу), у каждого направления есть свой вес, чем он больше, тем больше шанс нахождения корабля в данном направлении. Если корабль можно установить, то к «весу» соответствующего направления прибавляется количество палуб, возможно находящихся на данном направлении.

Напр.: если 4х-палубный корабль можно установить (нет пересечения палуб с границей карты или с ранее прострелянными клетками/обводкой корабля) слева от опорной точки (включая опорную клетку), то вес левого направления увеличится на 3 (3 клетки слева + 1 опорная), варианты расположения 3х-палубного корабля (вес слева – вес справа от опорной точки): 2-0, 1-1, 0-2

Если подбито от 2х клеток, алгоритм такой же, за исключением того, что опорной является последняя подбитая клетка, а подстановка живых кораблей соперника и вычисление весов происходит только по известному направлению (OX, OY).

Выстрел производится по ближайшей доступной для выстрела клетке в направлении с бОльшим весом.

**Схема алгоритма:**

****

****

****

****

****

****

# Листинг кода:

# unit bot;

# interface

# type

# TMapRow = array of integer;

# TMap = array of TMapRow;

# TCoordinates = array of integer;

# TListCoordinates = array of TCoordinates;

# const

# // Коды результатов выстрела:

# SHOT\_RESULT\_EMPTY: integer = 0; // Промах

# SHOT\_RESULT\_DAMAGE: integer = 2; // Корабль соперника ранен (подбит)

# SHOT\_RESULT\_KILL: integer = 3; // Корабль соперника убит (подбиты все палубы)

# {

# Вызывается один раз перед началом игры.

# Передаёт параметры, с которыми запущен турнир:

# - setCount: максимальное количество сетов в игре

# }

# procedure setParameters(setCount: integer);

# {

# Вызывается один раз в начале игры.

# }

# procedure onGameStart();

# {

# Вызывается один раз в начале сета.

# }

# procedure onSetStart();

# {

# Возвращает карту с расстановкой кораблей.

# Вызывается в начале каждого сета (каждый сет можно делать новую расстановку).

# Карта должна иметь размер 10х10 и быть заполнена следующим образом:

# - 0 - пустая клетка

# - 1 - клетка с палубой корабля

# На карте должно быть четыре 1-палубника, три 2-палубника, два 3-палубника и один 4-палубник.

# Корабли должны быть расположены строго вертикально или горизонтально (т.е., не иметь изгибов).

# Корабли не должны соприкасаться.

# }

# function getMap(): TMap;

# {

# Возвращает координаты клетки, в которую на текущем ходу стреляет бот.

# Координаты задаются массивом из двух целых чисел.

# Первая соответствует строке (координата по вертикали), вторая - столбцу (координата по горизонтали).

# Отсчёт ведётся с левого верхнего угла.

# Т.е., чтобы выстрелить в клетку map[i][j], необходимо вернуть [i, j].

# Результат выстрела будет гарантированно передан следующим вызовом через shotResult.

# }

# function shoot(): TCoordinates;

# {

# Вызывается сразу после shoot, чтобы передать результат выстрела.

# Возможные значения resultCode:

# - 0 - Промах

# - 2 - Корабль соперника ранен (подбит)

# - 3 - Корабль соперника убит (подбиты все палубы)

# }

# procedure shotResult(resultCode: integer);

# {

# Вызывается после выстрела соперника.

# Параметр cell - координаты клетки, в которую выстрелил соперник.

# }

# procedure onOpponentShot(cell: TCoordinates);

# {

# Вызывается один раз в конце сета.

# }

# procedure onSetEnd();

# {

# Вызывается один раз в конце игры.

# }

# procedure onGameEnd();

# implementation

# uses math;

# const

# numLoseShots: integer = 5; // количество промахов подряд при стрельбе по шаблону противника

# numOpTemplates: integer = 6; // количество сохраняемых шаблонов противника

# var

# firstSet: boolean = True;

# map:TMap;

# shots\_map: array[0..2] of TListCoordinates; // карта выcтрелов (4, 3-2, 1)

# // stats: array[0..5, 0..3, 0..2] of double;

# stats: array[0..3, 0..2] of double; // 0 - попаданий в сектор, 1 - вес сектора, 2 - номер сектора

# averageStats: array[0..3, 0..3] of double; // 0 - суммарный вес сектора, 1 - кол-во сумм, 2 - ср. арифметическое сумм, 3 - номер сектора

# shot\_op\_num: integer = 0; // количество выстрелов соперника

# histShots1:array[0..99] of array[0..1] of double; // история выстрелов соперника в текущем сете (0 - позиция, 1 - номер выстрела)

# averageHistShots:array[0..99] of array[0..4] of double; // история выстрелов соперника (ср. арифм) (0 - позиция, 1 - сумма номеров выстрелов по позиции, 2 - кол-во сумм, 3 - ср. арифм)

# opMap:TMap; // Поле противника

# shot\_num:integer = 0; // количество моих выстрелов

# ishots:array[0..2] of integer = (24,26,50); // оставшееся кол-во выстрелов в шаблонах

# iNumShotsTmpl:array[0..2] of integer = (0,0,0); // i номер выстрела для каждого шаблона

# opShips:array of integer; // оставшиеся корабли соперника

# finDirection:integer = -1; // направление добивания

# tmpFinDirection:integer = -1; // временное направление добивания

# finActivate:boolean = False; // добивание

# lastShotCoords:TCoordinates; // мой последний выстрел

# lastGoodShotCoords:TCoordinates; // мой последний успешный выстрел

# countInj:integer = 0; // количество раненых клеток

# finCoords:array[0..3] of integer = (-1, -1, -1, -1); // раненые клетки

# histCoords:array of array[0..19] of integer; // история расположения кораблей противника в последних сетах

# curOpCoords:array[0..19] of integer; // текущее расположение кораблей противника

# nhc:integer = 0; // id истории расположения кораблей противника на текущем сете

# nhb:integer = 0; // id послденей раненой клетки противника в сете

# shootId:integer = 0; // режим стрельбы (0 - по шаблону, 1 - по шаблону противника)

# countHistCoords:integer = 0; // текущее кол-во сохраненных карт кораблей соперника

# loseShotsCount:integer = 0; // текущее количество промахов подряд при стрельбе по шаблону противника

# sn:integer = 0;

# au:boolean = False;

# allCountInj:integer; // общее кол-во попаданий по сопернику в сете

# opShotsMap1:TMap; // карта выстрелов соперника в текущем сете

# opShotsMap2:TMap; // карта выстрелов соперника в предыдущем сете

# countPlayedSets:integer = 0; // кол-во сыгранных сетов

# function getSector(y, x:integer):integer;

# begin

# if((x >= 0) and (x <= 4) and (y >= 0) and (y <= 4)) then

# getSector := 0

# else if((x >= 5) and (x <= 9) and (y >= 0) and (y <= 4)) then

# getSector := 1

# else if((x >= 5) and (x <= 9) and (y >= 5) and (y <= 9)) then

# getSector := 2

# else

# getSector := 3;

# end;

# function fromXYtoSingle(y, x:integer):integer;

# begin

# fromXYtoSingle := y\*10 + x;

# end;

# function fromSingleToXY(c:integer):TCoordinates;

# begin

# fromSingleToXY := TCoordinates.Create(c div 10, c mod 10);

# end;

# //обводка убитого корабля

# procedure stroke();

# var

# i, j, k:integer;

# cords:TCoordinates;

# begin

# for i := 0 to 3 do

# begin

# if finCoords[i] <> -1 then

# begin

# // обводимая клетка

# cords := fromSingleToXY(finCoords[i]);

# for j := cords[0]-1 to cords[0]+1 do

# begin

# for k := cords[1]-1 to cords[1]+1 do

# begin

# // если соседняя клетка в пределах поля

# if ((j >= 0) and (j <= 9) and (k >= 0) and (k <= 9)) then

# begin

# // не "закрашиваем" раненые клетки

# if(opMap[j, k] <> 1) then

# begin

# opMap[j, k] := 2;

# end;

# end;

# end;

# end;

# end;

# end;

# finCoords[0] := -1;

# finCoords[1] := -1;

# finCoords[2] := -1;

# finCoords[3] := -1;

# end;

# // Добивание

# // dir = -1 - XY (ранена одна клетка)

# // dir = 1 - X

# // dir = 2 - Y

# // TODO - добивание в соотвестчие шаблону противника (при стрельбе по шаблону)

# function finShot(y, x, dir:integer):TCoordinates;

# var

# i,l,j,t,k,o:integer;

# e:integer;

# xp, xm, yp, ym:integer;

# dix:array[1..4] of integer = (1,2,3,4);

# dirvals:array[1..4] of integer;

# tmpFinDir:integer;

# dirAnalize:array[1..4] of array[1..3] of array[0..1] of integer;

# predCord:TCoordinates;

# q:Array[0..1] of integer;

# dl:integer;

# begin

# // начальные веса каждой клетки по каждому направлению (относительно y, x)

# // вероятность возможной установки корабля

# for i := 1 to 4 do

# begin

# for j := 1 to 3 do

# begin

# dirAnalize[i, j, 0] := 0; // вес

# dirAnalize[i, j, 1] := 0; // координаты в однозначной форме

# end;

# end;

# l := Length(opShips)-1;

# xp := 0; // вес положительного направления OX (вероятность установки корабля) - 2

# xm := 0; // вес отрицательного направления OX (вероятность установки корабля) - 4

# yp := 0; // вес положительного направления OY (вероятность установки корабля) - 3

# ym := 0; // вес отрицательного направления OY (вероятность установки корабля) - 1

# if ((dir = -1) or (dir = 1)) then

# begin

# //X

# for i := 0 to l do

# begin

# 

# j := max(0, x-(opShips[i]-1));

# dl := min(x+(opShips[i]-1), 9);

# // если длина корабля больше длины подбитых клеток

# if opShips[i] > countInj then

# begin

# t := x;

# while ((j <= x) and (t <= dl))do

# begin

# k := j;

# e := 0; // кол-во ошибок

# while ((k <= t) and (e = 0)) do

# begin

# // если входим в границы

# if ((k >= 0) and (k <= 9)) then

# begin

# // если клетка не опорная

# if k <> x then

# begin

# // если клетка пустая или подбитая

# if ( (opMap[y, k] = 0) or (opMap[y, k] = 1) )then

# begin

# // если клетка пустая - увеличить её вес на 1.

# if opMap[y, k] = 0 then

# begin

# // клетки слева от X

# if k < x then

# o := 4

# else // клетки справа от X

# o := 2;

# dirAnalize[o, 4-abs(x - k), 0] += 1;

# dirAnalize[o, 4-abs(x - k), 1] := fromXYtoSingle(y, k);

# end;

# 

# inc(k);

# end

# else

# inc(e);

# end

# else

# inc(k);

# end

# else

# inc(e);

# 

# end;

# // если корабль можно поставить (все клетки при подстановке пустые и входят в границы поля)

# if e = 0 then

# begin

# if ((j < x) and (opMap[y, j] = 0)) then

# xm += abs(x-j);

# if ((t > x) and (opMap[y, t] = 0)) then

# xp += t-x;

# end;

# if j = x then

# begin

# inc(t);

# end

# else if t = dl then

# begin

# inc(j);

# end

# else

# begin

# inc(j);

# inc(t);

# end;

# 

# end;

# end;

# end;

# end;

# if ((dir = -1) or (dir = 2)) then

# begin

# //Y

# for i := 0 to l do

# begin

# j := max(0, y-(opShips[i]-1));

# dl := min(y+(opShips[i]-1), 9);

# if opShips[i] > countInj then

# begin

# t := y;

# while ((j <= y) and (t <= dl)) do

# begin

# k := j;

# e := 0; // кол-во ошибок

# while ((k <= t) and (e = 0)) do

# begin

# 

# // если входим в границы

# if ((k >= 0) and (k <= 9)) then

# begin

# // если клетка не опорная

# if k <> y then

# begin

# // если клетка пустая или подбитая

# if ( (opMap[k, x] = 0) or (opMap[k, x] = 1) )then

# begin

# // если клетка пустая - увеличить её вес на 1.

# if opMap[k, x] = 0 then

# begin

# // клетки выше y

# if k < y then

# o := 1

# else // клетки ниже y

# o := 3;

# dirAnalize[o, 4-abs(y - k), 0] += 1;

# dirAnalize[o, 4-abs(y - k), 1] := fromXYtoSingle(k, x);

# end;

# inc(k);

# end

# else

# inc(e);

# end

# else

# inc(k);

# end

# else

# inc(e);

# 

# end;

# // если корабль можно поставить (все клетки при подстановке пустые и входят в границы поля)

# if e = 0 then

# begin

# if ((j < y) and ((opMap[j, x] = 0))) then

# ym += abs(y-j);

# if ((t > y) and (opMap[t, x] = 0)) then

# yp += t-y;

# end;

# if j = y then

# inc(t)

# else if t = dl then

# inc(j)

# else

# begin

# inc(j);

# inc(t);

# end;

# end;

# end;

# end;

# end;

# 

# // сортировка весов по направлениям (вверх, вправо, вниз, влево)

# dirvals[1] := ym;

# dirvals[2] := xp;

# dirvals[3] := yp;

# dirvals[4] := xm;

# for i := 1 to 3 do

# for j := 1 to 3 do

# if dirvals[j] > dirvals[j+1] then begin

# k := dirvals[j];

# dirvals[j] := dirvals[j+1];

# dirvals[j+1] := k;

# k := dix[j];

# dix[j] := dix[j+1];

# dix[j+1] := k;

# end;

# tmpFinDir := dix[4];

# // установка предположительного направления размещения

# // если направление по OX

# if ((tmpFinDir = 2) or (tmpFinDir = 4)) then

# tmpFinDirection := 1

# else

# tmpFinDirection := 2; // по OY

# // сортировка по весу клеток в направлении

# for i := 1 to 2 do

# for j := 1 to 2 do

# if dirAnalize[tmpFinDir, j, 0] > dirAnalize[tmpFinDir, j+1, 0] then begin

# q := dirAnalize[tmpFinDir, j];

# dirAnalize[tmpFinDir, j] := dirAnalize[tmpFinDir, j+1];

# dirAnalize[tmpFinDir, j+1] := q;

# end;

# 

# predCord := fromSingleToXY(dirAnalize[tmpFinDir, 3, 1]);

# // вернуть координаты выстрела

# finShot := TCoordinates.Create(predCord[0], predCord[1]);

# end;

# // стрелба по моим шаблонам

# // TODO - трансформировать шаблон под частые позиции соперника

# function templateShot():TCoordinates;

# var

# f:boolean = False;

# i:integer;

# id:integer;

# x,y:integer;

# begin

# i := 0;

# // перебираем оставшиеся выстрелы в шаблонах

# while((i<= 2) and (f = False)) do

# begin

# while ((ishots[i] > 0) and (f = False)) do

# begin

# id := iNumShotsTmpl[i];

# x := shots\_map[i][id][1];

# y := shots\_map[i][id][0];

# //если по позиции уже был выстрел

# if opMap[y, x] <> 0 then

# begin

# inc(iNumShotsTmpl[i]);

# end

# else

# begin

# inc(shot\_num); // увеличимть кол-во выстрелов

# inc(iNumShotsTmpl[i]); // изменить id выстрела в шаблоне выстрелов

# templateShot := TCoordinates.Create(y,x);

# f := True;

# end;

# 

# dec(ishots[i]); // уменьшить кол-во доступных выстрелов в шаблоне

# end;

# // переход к след. шаблону

# inc(i);

# end;

# end;

# // стрельба по шаблону противника

# function templateOpShot():TCoordinates;

# var

# i, j, k:integer;

# cords:TCoordinates;

# rCords:TCoordinates;

# t:boolean = False;

# persnts:array of array[0..1] of double; // коэффициент совпадений по каждому шаблону (0 - коэффициент, 1 - номер шаблона)

# n,m:integer;

# v:array[0..1] of double;

# mid:integer;

# posMap:array[0..99] of array[0..1] of integer; // массив пересечения множеств всех кораблей соперника ( 0 - вес каждой клетки, 1 - координаты )

# q:array[0..1] of integer;

# u:boolean = False;

# begin

# SetLength(persnts, numOpTemplates);

# for i := 0 to numOpTemplates-1 do

# begin

# persnts[i][0] := 0;

# persnts[i][1] := i;

# end;

# for i := 0 to 99 do

# begin

# posMap[i][0] := 0;

# posMap[i][1] := i;

# end;

# // если сохранена только одна карта кораблей соперника

# if countHistCoords = 1 then

# begin

# for i := 0 to 19 do

# begin

# if histCoords[nhc-1, i] <> -1 then

# begin

# cords := fromSingleToXY(histCoords[nhc-1, i]);

# // если клетка еще не простреливалась

# if opMap[cords[0], cords[1]] = 0 then

# begin

# rCords := cords;

# t := True;

# end;

# end;

# end;

# end

# else

# begin

# m := allCountInj;

# // по шаблонам

# for i := 0 to countHistCoords-1 do

# begin

# n := 0;

# // по клеткам в шаблоне

# for j := 0 to 19 do

# begin

# if histCoords[i, j] <> -1 then

# begin

# // по текущим кораблям соперника

# for k := 0 to 19 do

# begin

# if curOpCoords[k] <> -1 then

# begin

# // если координаты из прошлых шалонов совпадают с текущей позицией кораблей

# if histCoords[i, j] = curOpCoords[k] then

# begin

# inc(n); // увеличение кол-ва совпавших клеток

# end;

# end;

# end;

# end;

# end;

# // коэффициент совпадений в шаблоне

# if m <> 0 then

# begin

# persnts[i][0] := n/m;

# 

# // если хотя бы один коэффициент больше 0

# if persnts[i][0] > 0 then

# u := True;

# end;

# end;

# // если хотя бы один коэффициент больше 0

# if u = True then

# begin

# // сортировка коэффициентов

# for i := 0 to numOpTemplates-2 do

# for j := 0 to numOpTemplates-2 do

# if persnts[j][0] > persnts[j+1][0] then begin

# v := persnts[j];

# persnts[j] := persnts[j+1];

# persnts[j+1] := v;

# end;

# mid := round(persnts[numOpTemplates-1][1]);

# 

# for i := 0 to 19 do

# begin

# if histCoords[mid, i] <> -1 then

# begin

# cords := fromSingleToXY(histCoords[mid, i]);

# // если клетка еще не простреливалась

# if opMap[cords[0], cords[1]] = 0 then

# begin

# rCords := cords;

# t := True;

# break;

# end;

# end;

# end;

# end;

# // если координаты выстрела не удалось определить

# // определение координат выстрела по наиболее часто используемым позициям соперника

# if t = False then

# begin

# // если способ запускается первый раз или еще не было ошибки выполнения

# if au = False then

# begin

# // по шаблонам соперника

# for i := 0 to countHistCoords-1 do

# begin

# for k := 0 to 19 do

# begin

# if histCoords[i, k] <> -1 then

# begin

# inc(posMap[histCoords[i, k], 0]); // увеличить вес клетки в которой есть корабль

# end;

# end

# end;

# 

# // сортировка по наиболее часто встречающимся позициям

# for i := 0 to 98 do

# for j := 0 to 98 do

# if posMap[j][0] > posMap[j+1][0] then

# begin

# q := posMap[j];

# posMap[j] := posMap[j+1];

# posMap[j+1] := q;

# end;

# for i := 99 downto 0 do

# begin

# // если в клетке больше 1 совпадения кораблей на разных картах

# if posMap[i][0] > 1 then

# begin

# cords := fromSingleToXY(posMap[i][1]);

# // если клетка еще не простреливалась

# if opMap[cords[0], cords[1]] = 0 then

# begin

# rCords := cords;

# t := True;

# break;

# end;

# end;

# end;

# // если координаты не были получены - метод больше не выполняем, до след. сета

# if t = False then

# au := True;

# end;

# end;

# end;

# // если координаты выстрела не удалось определить, вернуть координаты по моим шаблонам

# if t = False then

# rCords := templateShot();

# templateOpShot := rCords;

# end;

# // установка корябля на карту

# // M - карта

# // x0, y0 - координаты крайней левой нижней точки корабля

# // r - ориентация корабля 0 - вертикаль, 1 - горизналь

# // dl - размер корабля

# procedure placeToMap(var m:TMap; y0, x0, r, dl:integer);

# var

# i, j, k:integer;

# tmpCords:array[0..3] of TCoordinates; // координаты выставленного корабля (для дальнейшей обводки)

# begin

# k := 0;

# for i := 0 to 3 do

# tmpCords[i] := TCoordinates.Create(-1,-1);

# for i := 0 to dl-1 do

# begin

# if r = 0 then

# begin

# m[y0-i, x0] := 1;

# tmpCords[k] := TCoordinates.Create(y0-i, x0);

# end

# else

# begin

# m[y0, x0+i] := 1;

# tmpCords[k] := TCoordinates.Create(y0, x0+i);

# end;

# inc(k);

# end;

# // временная обводка корабля (для дальнейшей корректной установки кораблей)

# for i := 0 to 3 do

# begin

# if tmpCords[i][0] <> -1 then

# begin

# for j := tmpCords[i][0]-1 to tmpCords[i][0]+1 do

# begin

# for k := tmpCords[i][1]-1 to tmpCords[i][1]+1 do

# begin

# // если соседняя клетка в пределах поля

# if ((j >= 0) and (j <= 9) and (k >= 0) and (k <= 9)) then

# begin

# // не "закрашиваем" клетки с палубами корабля

# if(m[j, k] <> 1) then

# begin

# m[j, k] := 2;

# end;

# end;

# end;

# end;

# end;

# end;

# end;

# // расстановка

# function placement():TMap;

# var

# tmpMap:TMap;

# i, j, k:integer;

# kf4:array[0..7] of array[0..1] of double; // коэффициенты для расстанвоки 4х-палубных (0 - коэффициент, 1 - положение на карте)

# kf3:array[0..3] of array[0..1] of double; // коэффициенты для расстанвоки 3х-палубных (0 - коэффициент, 1 - положение на карте)

# w:integer;

# t:array[0..1] of double;

# dh, vx, vy, ix, iy, ox, oy:integer;

# fx, fy, tx, ty:integer;

# f:boolean;

# tmpStats: array[0..3, 0..3] of double; // 0 - суммарный вес сектора, 1 - кол-во сумм, 2 - ср. арифметическое сумм, 3 - номер сектора

# tmpkf3:array of array[0..2] of double; // коэффициенты для расстанвоки 3х-палубных ( 0 - коэффициент, 1 - координаты опорной клетки, 2 - ориентация (0 - вертикаль, 1 - горизналь))

# tmpkf1:array of array[0..1] of double; // коэффициенты для расстанвоки 1-палубных ( 0 - коэффициент, 1 - координаты опорной клетки)

# tmpkf1Zero:array of integer; //позиции с нулевыми коэффициенты для расстанвоки 1-палубных ( координаты опорной клетки)

# kf3Num:integer=0; // кол-во найденных позиций для расстановки 3х-палубных

# kf1Num:integer=0; // кол-во найденных позиций для расстановки 1-палубных

# kf1NumZero:integer=0; // кол-во найденных позиций для расстановки 1-палубных с нулевыми коэффициентами

# singleCord1:integer;

# singleCord2:integer;

# b:array[0..2] of double;

# h:array[0..3] of double;

# oxy:TCoordinates;

# okNum:integer=0; // кол-во установленных кораблей

# begin

# tmpMap := TMap.Create(

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

# );

# tmpStats := averageStats;

# // 4х-палубный

# for i := 0 to 7 do

# begin

# kf4[i, 0] := 0;

# kf4[i, 1] := i;

# end;

# // 3х-палубный

# for i := 0 to 3 do

# begin

# kf3[i, 0] := 0;

# kf3[i, 1] := i;

# end;

# // по X верх

# for i := 0 to 9 do

# begin

# if i <= 4 then

# w := 7

# else

# w := 0;

# 

# // если соперник стрелял в клетку

# if opShotsMap2[0, i] = 1 then

# begin

# kf4[w, 0] := kf4[w, 0]+1; // увеличить коэффициент 4x-палубных

# kf3[0, 0] := kf3[0, 0]+1; // увеличить коэффициент 3x-палубных

# end;

# end;

# 

# kf4[7, 0] := kf4[7, 0] \* averageStats[0, 2];

# kf4[0, 0] := kf4[0, 0] \* averageStats[1, 2];

# kf3[0, 0] := kf3[0, 0] \* averageStats[0, 2] \* averageStats[1, 2];

# // по X низ

# for i := 0 to 9 do

# begin

# if i <= 4 then

# w := 4

# else

# w := 3;

# 

# // если соперник стрелял в клетку

# if opShotsMap2[9, i] = 1 then

# begin

# kf4[w, 0] := kf4[w, 0]+1;

# kf3[2, 0] := kf3[2, 0]+1;

# end;

# end;

# kf4[4, 0] := kf4[4, 0] \* averageStats[3, 2];

# kf4[3, 0] := kf4[3, 0] \* averageStats[2, 2];

# kf3[2, 0] := kf3[2, 0] \* averageStats[2, 2] \* averageStats[3, 2];

# // по Y слева

# for i := 0 to 9 do

# begin

# if i <= 4 then

# w := 6

# else

# w := 5;

# 

# // если соперник стрелял в клетку

# if opShotsMap2[i, 0] = 1 then

# begin

# kf4[w, 0] := kf4[w, 0]+1;

# kf3[3, 0] := kf3[3, 0]+1;

# end;

# end;

# kf4[6, 0] := kf4[6, 0] \* averageStats[0, 2];

# kf4[5, 0] := kf4[5, 0] \* averageStats[3, 2];

# kf3[3, 0] := kf3[3, 0] \* averageStats[0, 2] \* averageStats[3, 2];

# // по Y cправа

# for i := 0 to 9 do

# begin

# if i <= 4 then

# w := 1

# else

# w := 2;

# 

# // если соперник стрелял в клетку

# if opShotsMap2[i, 9] = 1 then

# begin

# kf4[w, 0] := kf4[w, 0]+1;

# kf3[1, 0] := kf3[1, 0]+1;

# end;

# end;

# kf4[1, 0] := kf4[1, 0] \* averageStats[1, 2];

# kf4[2, 0] := kf4[2, 0] \* averageStats[2, 2];

# kf3[1, 0] := kf3[1, 0] \* averageStats[1, 2] \* averageStats[2, 2];

# // сортировка коэффициентов 4х-палубного

# for i := 0 to 6 do

# for j := 0 to 6 do

# if kf4[j][0] > kf4[j+1][0] then

# begin

# t := kf4[j];

# kf4[j] := kf4[j+1];

# kf4[j+1] := t;

# end;

# // сортировка коэффициентов 3х-палубного

# for i := 0 to 2 do

# for j := 0 to 2 do

# if kf3[j][0] > kf3[j+1][0] then

# begin

# t := kf3[j];

# kf3[j] := kf3[j+1];

# kf3[j+1] := t;

# end;

# 

# // расстановка 4х-палубного

# case round(kf4[0][1]) of

# 0: placeToMap(tmpMap, 0, 5, 1, 4);

# 1: placeToMap(tmpMap, 4, 9, 0, 4);

# 2: placeToMap(tmpMap, 8, 9, 0, 4);

# 3: placeToMap(tmpMap, 9, 5, 1, 4);

# 4: placeToMap(tmpMap, 9, 1, 1, 4);

# 5: placeToMap(tmpMap, 8, 0, 0, 4);

# 6: placeToMap(tmpMap, 4, 0, 0, 4);

# 7: placeToMap(tmpMap, 0, 1, 1, 4);

# end;

# // НАЧАЛО расстановки 3х-палубных

# // TODO расстановка с учетом веса каждой клетки

# for i := 0 to 1 do

# begin

# // параметры для установки корабля

# case round(kf3[i, 1]) of

# 0: begin // установка по OX сверху

# dh := 1; // расположение корабля - горизонтально

# vx := -1; // итерации будут по X

# vy := 0; // y - постоянная

# end;

# 1: begin // установка по OY справа

# dh := 0; // расположение корабля - вертикально

# vx := 9;

# vy := -1;

# end;

# 2: begin // установка по OX снизу

# dh := 1;

# vx := -1;

# vy := 9;

# end;

# 3: begin // установка по OY слева

# dh := 0;

# vx := 0;

# vy := -1;

# end;

# end;

# // поиск позиции для установки в полученном стролбце/строке

# j := 0;

# while (j <= 9) do

# begin

# f := False;

# k := 0;

# // по каждой клетке корабля

# while ((k <= 2) and (f = False)) do

# begin

# // если проход по x

# if vx = -1 then

# begin

# ix := j+k;

# iy := vy;

# ox := j; // опроная клетка

# oy := vy; // опроная клетка

# end

# else

# begin

# ix := vx;

# iy := j+k;

# ox := vx;

# oy := j+2; // +2 т.к расстановка начинается с левого нижнего угла. А здесь координаты левого верхнего

# end;

# // если клетка пустая

# if tmpMap[iy, ix] = 0 then

# inc(k) // проверка след. клетки

# else

# f := True; // клетка не пустая - установить корабль не можем

# end;

# // если была ошибка, переходим к след. опорной клетке

# if f = True then

# inc(j)

# else // если все клетки пустые - место для установки найдено

# j := 10; // выход из цикла

# end;

# placeToMap(tmpMap, oy, ox, dh, 3);

# end;

# // НАЧАЛО расстановки 2х-палубных

# 

# // сортировка временнЫх коэффициентов выстрелов по четвертям

# for i := 0 to 2 do

# for j := 0 to 2 do

# if tmpStats[j][2] > tmpStats[j+1][2] then

# begin

# h := tmpStats[j];

# tmpStats[j] := tmpStats[j+1];

# tmpStats[j+1] := h;

# end;

# 

# 

# // проходим по 3м секторам с наименьшими коэффициентами

# // пока не будет установлено 3 корабля

# i := 0;

# okNum := 0;

# while okNum <= 2 do

# begin

# // номер сектора

# case round(tmpStats[i, 3]) of

# 0: begin

# // начальные коориднаты сканирования области

# fx := 0;

# fy := 4;

# // конечные координаты сканирования области (на 1 меньше, т.к сканируется n+1)

# tx := 3;

# ty := 1;

# end;

# 1: begin

# fx := 5;

# fy := 4;

# tx := 8;

# ty := 1;

# end;

# 2: begin

# fx := 5;

# fy := 9;

# tx := 8;

# ty := 6;

# end;

# 3: begin

# fx := 0;

# fy := 9;

# tx := 3;

# ty := 6;

# end;

# end;

# // сканирование сектора

# for j := fy downto ty do

# begin

# for k := fx to tx do

# begin

# // если по OX можно поставить корабль

# if ((tmpMap[j, k] = 0) and (tmpMap[j, k+1] = 0)) then

# begin

# singleCord1 := fromXYtoSingle(j, k);

# singleCord2 := fromXYtoSingle(j, k+1);

# inc(kf3Num); // увеличить кол-во найденных позиций

# SetLength(tmpkf3, kf3Num);

# tmpkf3[kf3Num-1][0] := averageHistShots[singleCord1,3]+averageHistShots[singleCord2,3]; // коэффициент (ср. арифм. номеров выстрелов противника по соответствующим позициям)

# tmpkf3[kf3Num-1][1] := singleCord1; // опорная точка

# tmpkf3[kf3Num-1][2] := 1; // ориентация (горизонтально)

# end;

# // если по OY можно поставить корабль

# if ((tmpMap[j, k] = 0) and (tmpMap[j-1, k] = 0)) then

# begin

# singleCord1 := fromXYtoSingle(j, k);

# singleCord2 := fromXYtoSingle(j-1, k);

# inc(kf3Num); // увеличить кол-во найденных позиций

# SetLength(tmpkf3, kf3Num);

# tmpkf3[kf3Num-1][0] := averageHistShots[singleCord1,3]+averageHistShots[singleCord2,3]; // коэффициент (ср. арифм. номеров выстрелов противника по соответствующим позициям)

# tmpkf3[kf3Num-1][1] := singleCord1; // опорная точка

# tmpkf3[kf3Num-1][2] := 0; // ориентация (вертикально)

# end;

# end;

# end;

# if kf3Num > 0 then

# begin

# // сортировка коэффициентов найденных позиций

# if kf3Num > 1 then

# for j := 0 to kf3Num-2 do

# for k := 0 to kf3Num-2 do

# if tmpkf3[k][0] > tmpkf3[k+1][0] then

# begin

# b := tmpkf3[k];

# tmpkf3[k] := tmpkf3[k+1];

# tmpkf3[k+1] := b;

# end;

# 

# // установка корабля в позиции с бОльшими коэффициентами (в которую стреляли позже всего (номера выстрелов больше))

# oxy := fromSingleToXY(round(tmpkf3[kf3Num-1, 1]));

# placeToMap(tmpMap, oxy[0], oxy[1], round(tmpkf3[kf3Num-1, 2]), 2);

# inc(okNum);

# end;

# if i = 3 then

# i := 0

# else

# inc(i);

# //очистить массив

# SetLength(tmpkf3, 0);

# kf3Num := 0;

# end;

# // Начало установки 1-палубных

# 

# // проходим по 4м секторам по возрастанию коэффициентов

# // пока не будет установлено 4 корабля

# i := 0;

# okNum := 0;

# while okNum <= 3 do

# begin

# // если кол-во сыгранных сетов > 5

# if countPlayedSets > 5 then

# begin

# // проверить, есть ли свободные позиции с нулевым коэффициентом

# for k := 0 to 99 do

# begin

# // если коэффициент равен 0 (за 5 сетов туда ни разу не стралял соперник)

# if averageHistShots[k,3] = 0 then

# begin

# oxy := fromSingleToXY(k);

# // если можно поставить корабль

# if tmpMap[oxy[0], oxy[1]] = 0 then

# begin

# inc(kf1NumZero); // увеличить кол-во позиций с нулевыми коэффициентами

# SetLength(tmpkf1Zero, kf1NumZero);

# tmpkf1Zero[kf1NumZero-1] := k;

# end;

# end;

# end;

# end;

# // если нет позиций с нулевыми коэффициентами или кол-во сыгранных сетов <= 5

# if ((countPlayedSets <= 5) or ( (countPlayedSets > 5) and (kf1NumZero = 0)) ) then

# begin

# // номер сектора

# case round(tmpStats[i, 3]) of

# 0: begin

# // начальные коориднаты сканирования области

# fx := 0;

# fy := 4;

# // конечные координаты сканирования области

# tx := 4;

# ty := 0;

# end;

# 1: begin

# fx := 5;

# fy := 4;

# tx := 9;

# ty := 0;

# end;

# 2: begin

# fx := 5;

# fy := 9;

# tx := 9;

# ty := 5;

# end;

# 3: begin

# fx := 0;

# fy := 9;

# tx := 4;

# ty := 5;

# end;

# end;

# // сканирование сектора

# for j := fy downto ty do

# begin

# for k := fx to tx do

# begin

# // если можно поставить корабль

# if (tmpMap[j, k] = 0) then

# begin

# singleCord1 := fromXYtoSingle(j, k);

# inc(kf1Num); // увеличить кол-во найденных позиций

# SetLength(tmpkf1, kf1Num);

# tmpkf1[kf1Num-1][0] := averageHistShots[singleCord1,3]; // коэффициент (ср. арифм. номеров выстрелов противника по позиции)

# tmpkf1[kf1Num-1][1] := singleCord1; // опорная точка

# end;

# end;

# end;

# 

# if kf1Num > 0 then

# begin

# // сортировка коэффициентов найденных позиций

# if kf1Num > 1 then

# for j := 0 to kf1Num-2 do

# for k := 0 to kf1Num-2 do

# if tmpkf1[k][0] > tmpkf1[k+1][0] then

# begin

# t := tmpkf1[k];

# tmpkf1[k] := tmpkf1[k+1];

# tmpkf1[k+1] := t;

# end;

# 

# // установка корабля в позиции с бОльшими коэффициентами (в которую стреляли позже всего (номера выстрелов больше))

# oxy := fromSingleToXY(round(tmpkf1[kf1Num-1, 1]));

# placeToMap(tmpMap, oxy[0], oxy[1], 1, 1);

# inc(okNum);

# end;

# if i = 3 then

# i := 0

# else

# inc(i);

# end

# else // если кол-во сетов больше 5 и есть позиции с нулевыми коэффициентами

# begin

# oxy := fromSingleToXY(tmpkf1Zero[0]);

# placeToMap(tmpMap, oxy[0], oxy[1], 1, 1);

# inc(okNum);

# end;

# // очистить массив

# SetLength(tmpkf1, 0);

# SetLength(tmpkf1Zero, 0);

# kf1Num := 0;

# kf1NumZero := 0;

# end;

# // очистка обводки у кораблей

# for i := 0 to 9 do

# begin

# for j := 0 to 9 do

# begin

# if tmpMap[i,j] = 2 then

# tmpMap[i,j] := 0;

# end;

# end;

# placement := tmpMap;

# end;

# procedure setParameters(setCount: integer);

# begin

# end;

# procedure onGameStart();

# var i, j, n, k:integer;

# begin

# // Массивы выстрелов для первого сета

# // формирование массива выстрелов. обстрел 4

# n := 2;

# k := 0;

# SetLength(shots\_map[0], 24);

# for i := 9 downto 0 do

# begin

# for j := 0 to 9-n do

# begin

# if (j mod 4) = 0 then

# begin

# shots\_map[0][k] := TCoordinates.Create(j+n, i);

# inc(k);

# end;

# end;

# inc(n);

# if n = 4 then

# n := 0;

# end;

# 

# // формирование массива выстрелов. обстрел 3,2

# n := 1;

# k := 0;

# SetLength(shots\_map[1], 26);

# for i := 0 to 9 do

# begin

# for j := 0 to 9-n do

# begin

# if (j mod 4) = 0 then

# begin

# shots\_map[1][k] := TCoordinates.Create(j+n, i);

# inc(k);

# end;

# end;

# dec(n);

# if n = -1 then

# n := 3;

# end;

# // формирование массива выстрелов. обстрел 1

# SetLength(shots\_map[2], 50);

# k := 0;

# for i := 0 to 9 do

# begin

# for j := 0 to 9 do

# begin

# if ((j mod 2) = (i mod 2)) then

# begin

# shots\_map[2][k] := TCoordinates.Create(i, j);

# inc(k);

# end;

# end;

# end;

# SetLength(histCoords, numOpTemplates);

# for i:= 0 to numOpTemplates - 1 do

# begin

# for j := 0 to 19 do

# begin

# histCoords[i, j] := -1;

# end;

# end;

# for j:= 0 to 3 do

# begin

# averageStats[j, 0] := 0;

# averageStats[j, 1] := 0;

# averageStats[j, 2] := 0;

# averageStats[j, 3] := j;

# end;

# for i := 0 to 99 do

# begin

# averageHistShots[i, 0] := 0;

# averageHistShots[i, 1] := 0;

# averageHistShots[i, 2] := 0;

# averageHistShots[i, 3] := 0;

# end;

# end;

# procedure onSetStart();

# var

# i,j:integer;

# begin

# inc(sn);

# // инифиализация поля соперника

# // 0 не стреляли

# // 1 - попадание

# // 2 - обводка убитых и промах

# opMap := TMap.Create(

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

# );

# 

# for j:= 0 to 3 do

# begin

# stats[j, 0] := 0; // попаданий в сектор

# stats[j, 1] := 0; // вес сектора

# stats[j, 2] := j; // номер сектора

# end;

# // оставшиеся корабли сеперника

# SetLength(opShips, 10);

# opShips[0] := 4;

# opShips[1] := 3;

# opShips[2] := 3;

# opShips[3] := 2;

# opShips[4] := 2;

# opShips[5] := 2;

# opShips[6] := 1;

# opShips[7] := 1;

# opShips[8] := 1;

# opShips[9] := 1;

# nhb := 0;

# for i := 0 to 19 do

# begin

# curOpCoords[i] := -1;

# end;

# iNumShotsTmpl[0] := 0;

# iNumShotsTmpl[1] := 0;

# iNumShotsTmpl[2] := 0;

# shot\_num := 0;

# ishots[0] := 24;

# ishots[1] := 26;

# ishots[2] := 50;

# loseShotsCount := 0;

# au := False;

# countInj := 0;

# allCountInj := 0;

# shot\_op\_num := 0;

# opShotsMap1 := TMap.Create(

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

# );

# for i := 0 to 99 do

# begin

# histShots1[i, 0] := 0;

# histShots1[i, 1] := 0;

# end;

# end;

# function getMap(): TMap;

# begin

# if firstSet then

# begin

# // Расстановка для первого сета

# map := TMap.Create(

# TMapRow.Create(0, 1, 0, 0, 0, 0, 1, 1, 1, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 1, 1),

# TMapRow.Create(0, 0, 0, 1, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(1, 0, 0, 0, 0, 0, 0, 0, 0, 1),

# TMapRow.Create(1, 0, 0, 0, 0, 0, 1, 0, 0, 1),

# TMapRow.Create(1, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 1, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

# TMapRow.Create(1, 1, 1, 1, 0, 1, 1, 0, 0, 0)

# );

# end

# else

# map := placement();

# 

# getMap := map;

# end;

# function shoot(): TCoordinates;

# var

# sh:TCoordinates;

# begin

# // если в режиме добивания

# if finActivate = True then

# begin

# sh := finShot(lastGoodShotCoords[0], lastGoodShotCoords[1], finDirection);

# lastShotCoords := sh;

# shoot := sh;

# end

# else

# begin

# // если сет первый или режим стрельбы - по шаблону

# if ((firstSet = True) or (shootId = 0)) then

# begin

# // стрельба по шаблону

# sh := templateShot();

# lastShotCoords := sh;

# shoot := sh;

# end

# else if shootId = 1 then

# begin

# // режим стрельбы по позициям противника

# sh := templateOpShot();

# 

# lastShotCoords := sh;

# shoot := sh;

# end;

# end;

# 

# end;

# procedure shotResult(resultCode: integer);

# var i :integer;

# begin

# // если ранили

# if resultCode = 2 then

# begin

# finCoords[countInj] := fromXYtoSingle(lastShotCoords[0], lastShotCoords[1]); // добавить клетку в список раненых

# // увеличение кол-ва раненых

# inc(countInj);

# inc(allCountInj);

# lastGoodShotCoords := lastShotCoords;

# opMap[lastShotCoords[0], lastShotCoords[1]] := 1;

# finActivate := True; // вкл режим добивания

# // если не определено направление добивания

# if finDirection = -1 then

# begin

# // если добивание уже было => направление определено

# if tmpFinDirection <> -1 then

# begin

# finDirection := tmpFinDirection;

# end;

# end;

# curOpCoords[nhb] := fromXYtoSingle(lastShotCoords[0], lastShotCoords[1]);

# inc(nhb);

# end

# else if resultCode = 0 then // промах

# begin

# opMap[lastShotCoords[0], lastShotCoords[1]] := 2;

# // если не в режиме добивания

# if finActivate = False then

# begin

# // если стрельба по шаблону противника

# if shootId = 1 then

# begin

# inc(loseShotsCount); // увеличить кол-во промахов

# // если кол-во промахов больше установленного

# if loseShotsCount > numLoseShots then

# begin

# shootId := 0; // сменить режим стрельбы на стрелбу по моим шаблонам

# end;

# end;

# end;

# end

# else if resultCode = 3 then // убили

# begin

# inc(allCountInj);

# opMap[lastShotCoords[0], lastShotCoords[1]] := 1;

# finCoords[countInj] := fromXYtoSingle(lastShotCoords[0], lastShotCoords[1]); // добавить клетку в список раненых

# // удалить убитый корабль из списка кораблей противника

# for i := 0 to Length(opShips)-1 do

# begin

# if(opShips[i] = countInj+1) then

# begin

# delete(opShips, i, 1);

# break;

# end;

# end;

# countInj := 0;

# finActivate := False; // выкл режим добивания

# finDirection := -1;

# tmpFinDirection := -1;

# curOpCoords[nhb] := fromXYtoSingle(lastShotCoords[0], lastShotCoords[1]);

# inc(nhb);

# stroke();

# end;

# end;

# procedure onOpponentShot(cell: TCoordinates);

# var

# sid:integer;

# begin

# opShotsMap1[cell[0], cell[1]] := 1;

# inc(shot\_op\_num); // увеличение кол-во выстрелов противника

# histShots1[fromXYtoSingle(cell[0], cell[1]), 0] := fromXYtoSingle(cell[0], cell[1]);

# histShots1[fromXYtoSingle(cell[0], cell[1]), 1] := shot\_op\_num;

# sid := getSector(cell[0], cell[1]); // определение сектора выстрела

# stats[sid, 0] := stats[sid, 0] + 1; // кол-во попаданий в сектор +1

# stats[sid, 1] := stats[sid, 1] + 1 \* Power(0.9, shot\_op\_num-1); // вес сектора

# end;

# procedure onSetEnd();

# var i:integer;

# begin

# inc(countPlayedSets);

# // сохранение истории выстрелов

# for i := 0 to 99 do

# begin

# averageHistShots[i, 0] := histShots1[i, 0];

# averageHistShots[i, 1] := averageHistShots[i, 1] + histShots1[i, 1]; // увеличение суммы номеров выстрелов

# averageHistShots[i, 2] := averageHistShots[i, 2] + 1; // увеличение кол-ва сумм

# averageHistShots[i, 3] := averageHistShots[i, 1] / averageHistShots[i, 2]; // ср. арифметическое

# end;

# opShotsMap2 := opShotsMap1;

# 

# for i := 0 to 3 do

# begin

# averageStats[i, 0] := averageStats[i, 0] + stats[i, 1]; // увеличение суммы весов

# averageStats[i, 1] := averageStats[i, 1] + 1; // увеличение кол-ва сумм весов

# averageStats[i, 2] := averageStats[i, 0] / averageStats[i, 1]; // ср. арифметическое

# end;

# // устанавливаем режим стрельбы по шаблону соперника

# shootId := 1;

# firstSet := False;

# 

# // сохраняем расположения кораблей соперника в историю

# histCoords[nhc] := curOpCoords;

# 

# // увеличить счетчик сохраняемых шаблонов соперника

# if nhc = numOpTemplates-1 then

# nhc := 0

# else

# inc(nhc);

# // увеличить кол-во сохраненных шаблонов кораблей соперника

# if countHistCoords < numOpTemplates-1 then

# inc(countHistCoords);

# end;

# procedure onGameEnd();

# begin

# end;

# end.

# Вывод: в ходе выполнения лабораторной работы был реализован алгоритм игры «Морской бой» на языке Pascal.