МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №3 по дисциплине

«Теория автоматов»

Выполнила студент группы ИВТб-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Птахова А.М

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Мельцов В.Ю

Киров 2021

**Цель работы**: научиться разрабатывать алгоритм для игры в «Морской бой» и реализовывать его программно.

**Задание**: разработать алгоритм работы бота для игры «Морской бой» и программно его реализовать.

Начальные условия:

M - массив 10х10. Он нужен для отметки кораблей, выстрелов и зон, в которые не стрелять. Первоначально все ячейки имеют значение 0. При выстреле значение ячейки может стать равным -1(при промахе) или 1(при попадании в корабль). При потоплении корабля его контур также получает значения (-1) – зона, по которой не стрелять.

Damage – переменная, отвечающая за наличие или отсутствие раненных

**Описание алгоритма**

1. Стрельба для поиска

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |

Создается дополнительный массив, каждая ячейка которого имеет номер, обозначающий номер хода, и переменная для отсчитывания ходов. После каждого выстрела увеличиваем переменную на 1 и сопоставляем её значение с массивом, беря его ячейку.

1. Попадание

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X |  |  | X | X | X |
| X | ранение | X |  | X | ранение | X |
|  | X |  |  | X | X | X |

Клетки из сторон Клетки из области

При попадании в корабль отмечаем раненную клетку 1, а клетки из области, находящиеся на диагоналях отмечаем, как промах. Присваиваем damage значение истины, находим следующую клетку из сторон, запоминаем координаты раненной клетки.

Для того, чтобы найти следующую клетку, создаем массив направлений, который проверяет клетки вокруг заданной ячейки. Массив содержит шаги на следующие клетки. Для обхода клеток из области задается шаг: шаг, равный 2 – для обхода клеток, расположенных на диагоналях; шаг, равный 1 – для обхода всех клеток из области.

1. Потопление корабля

Как только корабль потоплен, переходим к пункту 1 и не забываем обвести контур, для исключения его из области выстрелов. Damage присваиваем значение ложь.

**Листинг программы**

function CheckCell(map:Tmap;cell:TCoordinates;code:integer):boolean;

begin

    if (cell[0] >= 0) and (cell[0] < 10) and

        (cell[1] >= 0) and (cell[1] < 10) then

    begin

        if map[cell[0],cell[1]]=code then

        begin

            Exit(true);

        end;

    end;

    Exit(false);

end;

function getSideCell(cell:TCoordinates;side:integer):TCoordinates;

begin

    Exit(TCoordinates.Create(cell[0]+sides[0][side],cell[1]+sides[1][side]));

end;

function getAreaCell(cell:TCoordinates;side:integer):TCoordinates;

begin

    Exit(TCoordinates.Create(cell[0]+area[0][side],cell[1]+area[1][side]));

end;

function CheckEndShip(map:Tmap;cell:TCoordinates):boolean;

var i:integer;

    tcell:TCoordinates;

begin

    for i:=0 to 7 do

    begin

        tcell:=getAreaCell(cell,i);

        if CheckCell(map,tcell,1) then

        begin

            Exit(true);

        end;

    end;

    Exit(false);

end;

function FindSide(map:Tmap;cell:TCoordinates):integer;

var i,j:integer;

    tcell:TCoordinates;

begin

        //Если нужно поменять направление на противоположное(корабль подбит больше 1 раза и мы зашли в тупик)

        for i:=0 to 3 do

        begin

            tcell:=getSideCell(cell,i);

            if CheckCell(map,tcell,1) then

            begin

                Exit((i+2)mod 4);

            end;

        end;

        //Выбираем первоначальное направление

        for i:=0 to 3 do

        begin

            tcell:=getSideCell(cell,i);

            if CheckCell(map,tcell,0) then

            begin

                Exit(i);

            end;

        end;

end;

// procedure finishship();

// begin

// end;

function getNullMap():Tmap;

begin

    Exit(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)

    ));

end;

procedure getMoves(var steps:TMoves;var al:Tmap);

var i,j:integer;

begin

    for i:=0 to 9 do

    begin

        for j:=0 to 9 do

        begin

          steps[al[i,j]]:=TCoordinates.Create(i,j);

        end;

    end;

end;

function getAl():TMap;

begin

    Exit(TMap.Create(

         TMapRow.Create(51, 26, 52, 4, 53, 32, 54, 12, 55, 42),

        TMapRow.Create(25, 56, 3, 57, 31, 58, 11, 59, 41, 60),

        TMapRow.Create(61, 2, 62, 30, 63, 10, 64, 40, 65, 20),

        TMapRow.Create(1, 66, 29, 67, 9, 68, 39, 69, 19, 70),

        TMapRow.Create(71, 28, 72, 8, 73, 38, 74, 18, 75, 48),

        TMapRow.Create(27, 76, 7, 77, 37, 78, 17, 79, 47, 80),

        TMapRow.Create(81, 6, 82, 36, 83, 16, 84, 46, 85, 24),

        TMapRow.Create(5, 86, 35, 87, 15, 88, 45, 89, 23, 90),

        TMapRow.Create(91, 34, 92, 14, 93, 44, 94, 22, 95, 50),

        TMapRow.Create(33, 96, 13, 97, 43, 98, 21, 99, 49, 100))

    );

end;

procedure setParameters(setCount: integer);

begin

end;

procedure onGameStart();

begin

    area:=TMap.Create(

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

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

    );

    sides:=TMap.Create(

        TMapRow.Create(0, -1, 0, 1),

        TMapRow.Create(-1, 0,  1, 0)

    );

end;

procedure onSetStart();

begin

    lastShotIdx := 0;

    move:=0;

    al:=getAl;

    getMoves(moves,al);

    statusMap:=getNullMap();

    damage:=false;

end;

function getMap(): TMap;

begin

    result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

    );

end;

procedure MarkDamageCell(var map:Tmap;cell:TCoordinates;step:integer);

var i:integer;

    tcell:TCoordinates;

begin

    i:=0;

    while(i<8)do

    begin

        tcell:=getAreaCell(cell,i);

        if CheckCell(map,tcell,0) then

        begin

             map[tcell[0],tcell[1]]:=-1;

        end;

        i:=i+step;

    end;

end;

function shoot(): TCoordinates;

var tcell:TCoordinates;

begin

     if damage=true then

    begin

        tcell:=getSideCell(crd,side);

        if not CheckCell(statusMap,tcell,0) then

        begin

            crd:=cellFirstDamage;

            side:=FindSide(statusMap,crd);

        end;

        crd:=getSideCell(crd,side);

    end

    else

    begin

        repeat

            Inc(move);

            crd:=moves[move];

        until (statusMap[crd[0],crd[1]]=0);

    end;

    shoot:=crd;

end;

procedure ResultEmpty();

begin

    statusMap[crd[0],crd[1]]:=-1;

    if(damage) then

    begin

        if CheckEndShip(statusMap,crd) then

        begin

        end;

        crd:=cellFirstDamage;

        side:=FindSide(statusMap,cellFirstDamage);

    end;

end;

procedure ResultDamage();

begin

    statusMap[crd[0],crd[1]]:=1;

    MarkDamageCell(statusMap,crd,2);

    if not damage then

    begin

        damage:=true;

        side:=FindSide(statusMap,crd);

        cellFirstDamage:=crd;

    end;

end;

procedure Resultkill();

begin

    //cellFirstDamage:=crd;

    statusMap[crd[0],crd[1]]:=1;

    MarkDamageCell(statusMap,crd,1);

    if(damage)then

    begin

        MarkDamageCell(statusMap,cellFirstDamage,1);

    end;

    damage:=false;

end;

procedure shotResult(resultCode: integer);

begin

if resultCode = SHOT\_RESULT\_EMPTY then

    begin

        ResultEmpty();

    end

    else if resultCode = SHOT\_RESULT\_DAMAGE then

    begin

        ResultDamage();

    end else if resultCode = SHOT\_RESULT\_KILL then

    begin

        ResultKill();

    end;

end;

**Вывод:**

В ходе выполнения лабораторной работы были получены навыки разработки алгоритма для игры «Морской бой».