МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
(ВятГУ)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Создание бота для игры «Морской бой»
Отчет по лабораторной работе №2 по дисциплине
«Теория автоматов»

Выполнил студент группы ИВТ-21_____/Рзаев А.Э./
Проверил доцент кафедры ЭВМ_____/Мельцов В.Ю./

Киров 2016

1 Описание задания

Написать модуль, реализующий бота, который играет в игру «Морской бой».

2 Разработка алгоритма

Сначала алгоритм простреливает клетки по диагоналям с шагом 3 по шаблону на рисунке 1, тем самым будет найден четырехпалубник. Затем по шаблону на рисунке 2 простреливаются клетки, при этом будут найдены все двух- и трехпалубники. После этого оставшиеся клетки простреливаются по порядку, пока не закончатся.

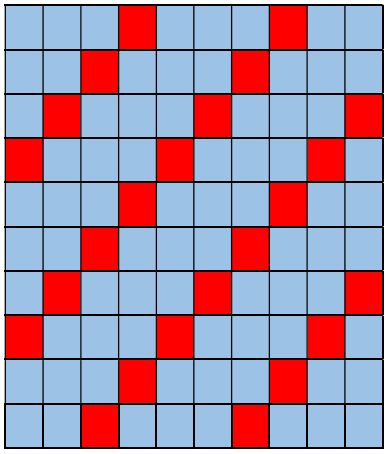

Рисунок 1 – шаблон поиска четырехпалубника

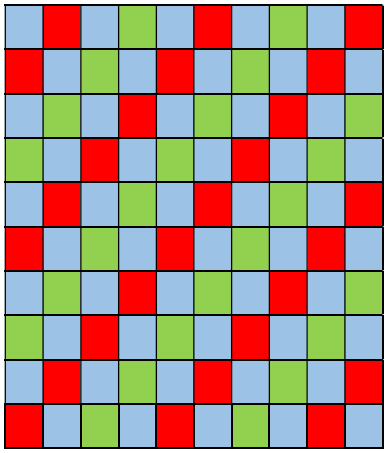

Рисунок 2 – шаблон поиска двух- и трехпалубников

3 Схема алгоритма

Схема алгоритма, по которому работает бот представлена на рисунке 3.

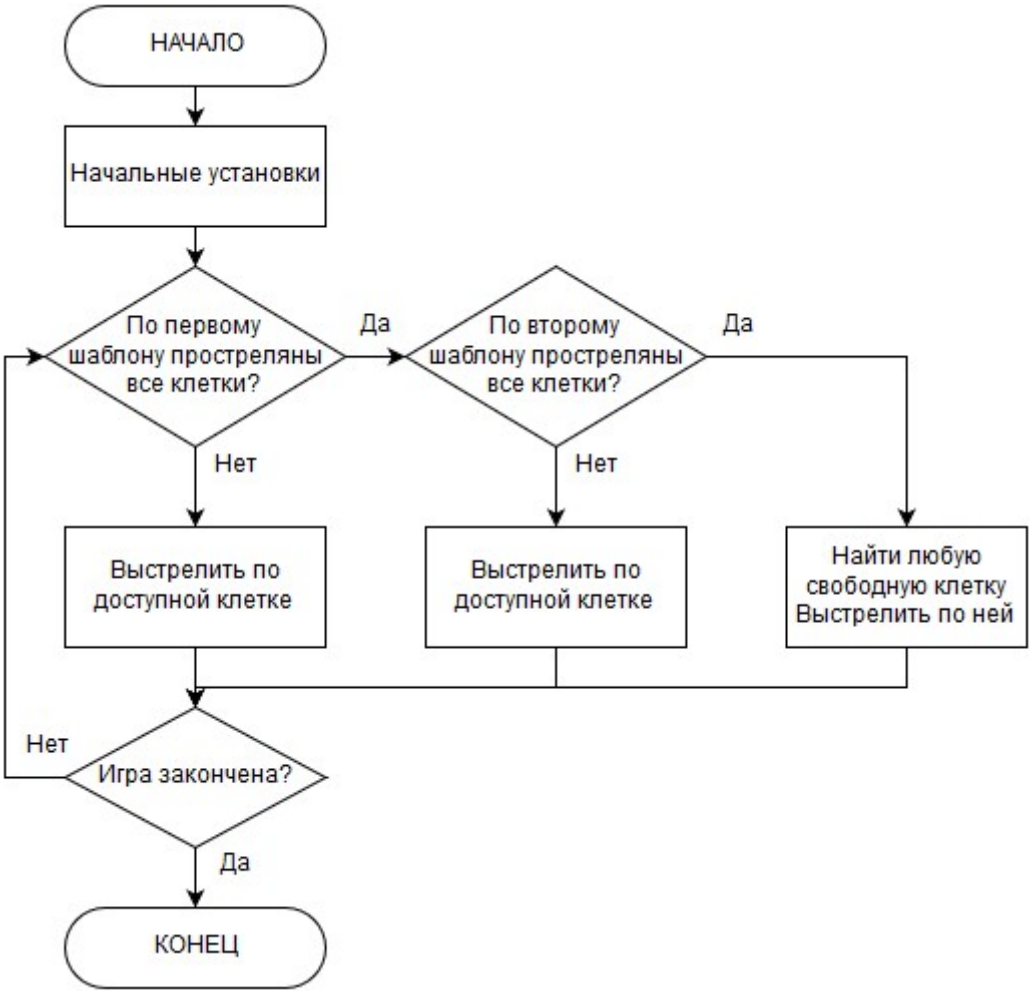

Рисунок 3 – Схема алгоритма работы бота

4 Листинг программы

```
unit SeaFightBotImpl;

{$mode objfpc}{$H+}

interface

type

    TByteArray = array of Byte;
    TByteArray2D = array of TByteArray;

const

    EMPTY  = 0;
    BLOCK  = 1;
    DAMAGE = 2;
    KILL   = 3;

    function getMap: TByteArray2D;
    function shoot: TByteArray;
    procedure shootResult(resultCode: Integer);
    procedure rivalShoot(point: TByteArray);
    procedure setParams(setsPerGame: Integer);
    procedure onCurrentSetEnd;
    procedure onCurrentGameEnd;

implementation

var
    rivalMap: TByteArray2D; // [X, Y]
```

```
    shoots4x, shoots32x: TByteArray2D; // [(X, Y)]

// pseudorandom generator
const RAND_MAX = 32767;
var
    seed: Longword;

procedure srand(s: Longword);
begin
    seed := s;
end;

function rand: Integer;
var
    next, res: Longword;
begin
    next := seed;

    next := next * 1103515245;
    next := next + 12345;
    res := (next div 65536) mod 2048;

    next := next * 1103515245;
    next := next + 12345;
    res := res shl 10;
    res := res xor ((next div 65536) mod 1024);

    next := next * 1103515245;
    next := next + 12345;
    res := res shl 10;
    res := res xor ((next div 65536) mod 1024);

    seed := next;

    rand := res and RAND_MAX;
end;
//

procedure swap(var a, b: TByteArray);
var
    t: TByteArray;
begin
    t := a;
    a := b;
    b := t;
end;

procedure shuffleShoots(shoots: TByteArray2D);
var
    i, a, b: Integer;
begin
    for i := 1 to Length(shoots) div 3 do
    begin
        a := rand() mod Length(shoots);
        b := rand() mod Length(shoots);
        swap(shoots[a], shoots[b]);
    end;
end;

function getNextShoot: TByteArray; // (X, Y)
var
    i, j: Integer;
    res: TByteArray;
    cur: TByteArray;
    shoots1x: TByteArray2D;
    cnt: Integer;
begin
```

```
    res := TByteArray.Create(-1, -1);

    // find 4x ship
    for i := 0 to Length(shoots4x) - 1 do
    begin
        cur := shoots4x[i]; // [X, Y]
        if rivalMap[cur[0]][cur[1]] = EMPTY then
        begin
            result := cur;
            exit;
        end;
    end;

    // find 3x and 2x ships
    for i := 0 to Length(shoots32x) - 1 do
    begin
        cur := shoots32x[i]; // [X, Y]
        if rivalMap[cur[0]][cur[1]] = EMPTY then
        begin
            result := cur;
            exit;
        end;
    end;

    // find 1x ships
    for i := 0 to 9 do
        for j := 0 to 9 do
            if rivalMap[i][j] = EMPTY then
            begin
                result := TByteArray.Create(i, j);
                exit;
            end;
end;

function getMap: TByteArray2D;
begin
    rivalMap := TByteArray2D.Create(
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
    );

    shuffleShoots(shoots4x);
    shuffleShoots(shoots32x);

    result := TByteArray2D.Create(
        TByteArray.Create(0, 1, 0, 0, 0, 0, 1, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
        TByteArray.Create(1, 1, 1, 0, 0, 0, 1, 0, 0, 1),
        TByteArray.Create(0, 0, 0, 0, 1, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 1, 0, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
        TByteArray.Create(1, 1, 0, 0, 0, 1, 0, 0, 1, 1),
        TByteArray.Create(0, 0, 0, 0, 0, 1, 0, 0, 0, 0),
        TByteArray.Create(0, 0, 0, 0, 0, 1, 0, 1, 0, 0),
        TByteArray.Create(0, 0, 1, 0, 0, 1, 0, 0, 0, 0)
    );
end;

function shoot: TByteArray;
```

```
var
    currentShoot: TByteArray;
    currentX, currentY: Integer;
begin
    currentShoot := getNextShoot;

    currentX := currentShoot[0];
    currentY := currentShoot[1];

    rivalMap[currentX][currentY] := BLOCK;

    result := TByteArray.Create(currentX, currentY);
end;

procedure shootResult(resultCode: Integer);
begin
end;

procedure rivalShoot(point: TByteArray);
begin
end;

procedure setParams(setsPerGame: Integer);
begin
    srand(9931);

    shoots4x := TByteArray2D.Create(
        TByteArray.Create(0, 3), TByteArray.Create(1, 2),
        TByteArray.Create(2, 1), TByteArray.Create(3, 0),
        TByteArray.Create(0, 7), TByteArray.Create(1, 6),
        TByteArray.Create(2, 5), TByteArray.Create(3, 4),
        TByteArray.Create(4, 3), TByteArray.Create(5, 2),
        TByteArray.Create(6, 1), TByteArray.Create(7, 0),
        TByteArray.Create(2, 7), TByteArray.Create(3, 6),
        TByteArray.Create(4, 5), TByteArray.Create(5, 4),
        TByteArray.Create(6, 3), TByteArray.Create(7, 2),
        TByteArray.Create(8, 1), TByteArray.Create(9, 0),
        TByteArray.Create(6, 7), TByteArray.Create(7, 6),
        TByteArray.Create(8, 5), TByteArray.Create(9, 4)
    );

    shoots32x := TByteArray2D.Create(
        TByteArray.Create(0, 1), TByteArray.Create(1, 0),
        TByteArray.Create(0, 5), TByteArray.Create(1, 4),
        TByteArray.Create(2, 3), TByteArray.Create(3, 2),
        TByteArray.Create(4, 1), TByteArray.Create(5, 0),
        TByteArray.Create(0, 9), TByteArray.Create(1, 8),
        TByteArray.Create(2, 7), TByteArray.Create(3, 6),
        TByteArray.Create(4, 5), TByteArray.Create(5, 4),
        TByteArray.Create(6, 3), TByteArray.Create(7, 2),
        TByteArray.Create(8, 1), TByteArray.Create(9, 0),
        TByteArray.Create(4, 9), TByteArray.Create(5, 8),
        TByteArray.Create(6, 7), TByteArray.Create(7, 6),
        TByteArray.Create(8, 5), TByteArray.Create(9, 4),
        TByteArray.Create(8, 9), TByteArray.Create(9, 8)
    );
end;

procedure onCurrentSetEnd;
begin
end;

procedure onCurrentGameEnd;
begin
end;

end.
```