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 дисциплины

«Теория автоматов»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А./

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

2021 г.

**Задание:**

Реализовать бота для игры “Морской бой”.

**Словесное описание алгоритма:**

В начале каждого сета массив ходов “BESTHOD” бота заполняется цифрами от 5 до 10, при этом, чем больше цифра в ячейке, тем приоритетнее она для бота (бот будет стрелять по 10 пока их не останется, затем по 9 и т.д.). Цифры распределены следующим образом: сначала бот будет стрелять по диагоналям, а именно по 4 штукам (1 – от [0, 3] до [3, 0], 2 - от [0, 7] до [7, 0], 3 - от [2, 9] до [9, 2], 4 - от [6, 9] до [9, 6]), остальные ячейки заполнены произвольно цифрами от 5 до 10.

В 1-ом состоянии бот выбирает первую попавшуюся наибольшую цифру из массива “BESTHOD” и стреляет в ячейку с такими же координатами, как и у ячейки с наибольшим числом. Если при очередном ходе бот не попадает по кораблю противника, то в ячейку массива “BESTHOD” с координатами хода записывается цифра “1”, то есть обозначает клетку, в которую уже стрелял.

Если бот попадает, то есть 2 варианта: либо бот “убивает” корабль противника, либо “ранит” его корабль.

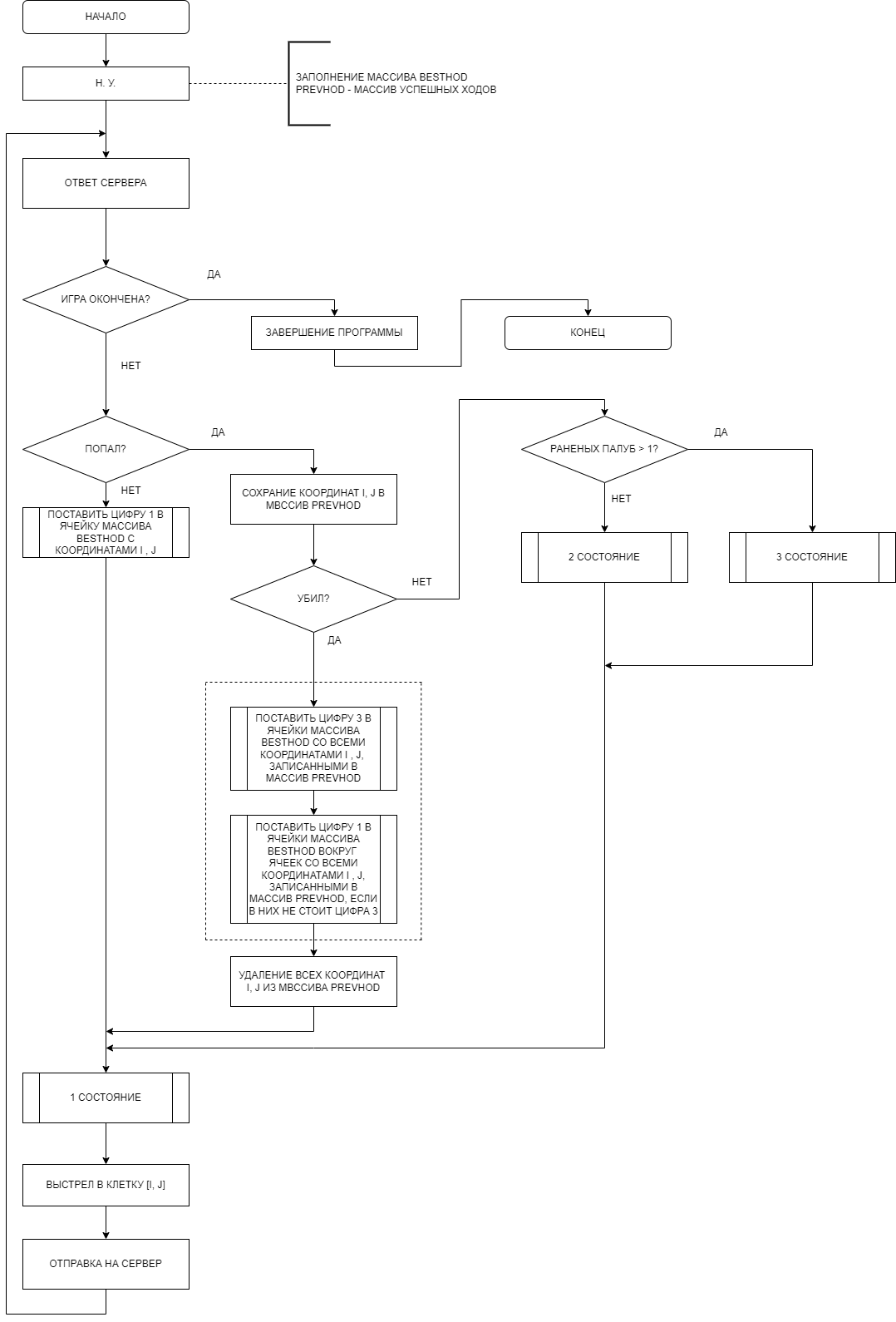
В первом случае в ячейку массива “BESTHOD” с координатами хода бот ставит цифру “3”, а в ячейки вокруг – цифру “1”.

Во втором случае бот переходит во 2-ое состояние: в ячейку массива “BESTHOD” с координатами хода ставит цифру “2”, а в ячейки слева, справа, сверху, снизу ставит цифру “11” (если до этого там не стояли цифры “1”), a также сохраняет координаты клетки, в которую стрелял на этом ходе, и возвращается в 1 состояние.

При повторном попадании по кораблю (даже после череды промахов) бот переходит в 3-е состояние: бот определяет направление расположения корабля с помощью хранящихся координат и меняет уже стоящие числа “11” на цифру “1” в зависимости от направления расположения корабля. Если координата i второго попадания больше или меньше координаты первого, то корабль расположен вертикально и выстрелы идут вниз. Бот уберёт числа “11” по бокам ячейки с первым попаданием и поставит ещё одно число “11” в i+1 координату данного хода. Для кораблей, расположенных горизонтально, алгоритм аналогичный: числа “11” заменятся на цифры “1” сверху и снизу клетки) и возвращается в 1 состояние. То есть у бота есть 2 варианта для выстрела: выше первых двух попаданий или ниже (аналогично для кораблей, расположенных горизонтально: левее или правее). Сначала бот выстрелит в ячейку с меньшими i (j), а потом, в случае промаха, в ячейку с большими i (j). Если верхняя (левая) ячейка - промах, то у бота остаётся один вариант: бот ставит число “11” ещё ниже (справа) и следующим ходом стреляет туда. Когда корабль потоплен, бот ставит цифру “1” вокруг данного корабля, так как хранил координаты попаданий, а в ячейки с координатами попаданий ставит “2”. После данной процедуры координаты клеток потопленного корабля удаляются из массива. Далее бот снова переходит в 1 состояние (см. выше).

**Схема алгоритма:**

**Основной алгоритм:**



По поводу основного алгоритма (выделенные символы): в 1 символе устанавливаются цифры “3” в ячейки, по которым были попадания, после того, как корабль потоплен. Во 2 символе у каждой клетки, в которой есть палуба корабля, в соседних ячейках ставятся цифры “1”, если это возможно (если там не было цифры “3”). Ситуация, когда в соседней клетке может быть цифра “3” возникает, когда корабль больше однопалубного:

до выстрела, потопившего корабль (“2” – это попадание, “11” - следующий выстрел) –

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 6 | 7 | 6 | 8 |
| 1 | 2 | 2 | 11 | 5 |
| 8 | 9 | 5 | 8 | 7 |

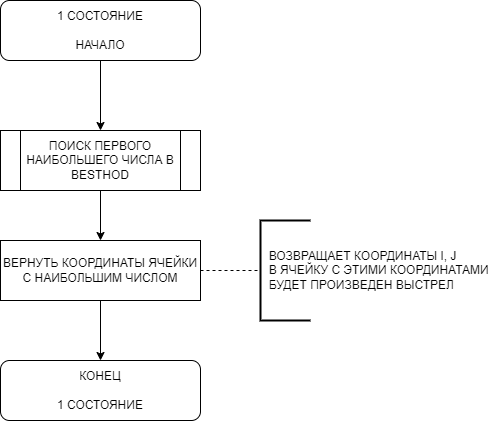
после того, как корабль потопили (цифры “3” уже поставлены, а цифры “1” еще нет) -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 6 | 7 | 6 | 8 |
| 1 | 3 | 3 | 3 | 5 |
| 8 | 9 | 5 | 8 | 7 |

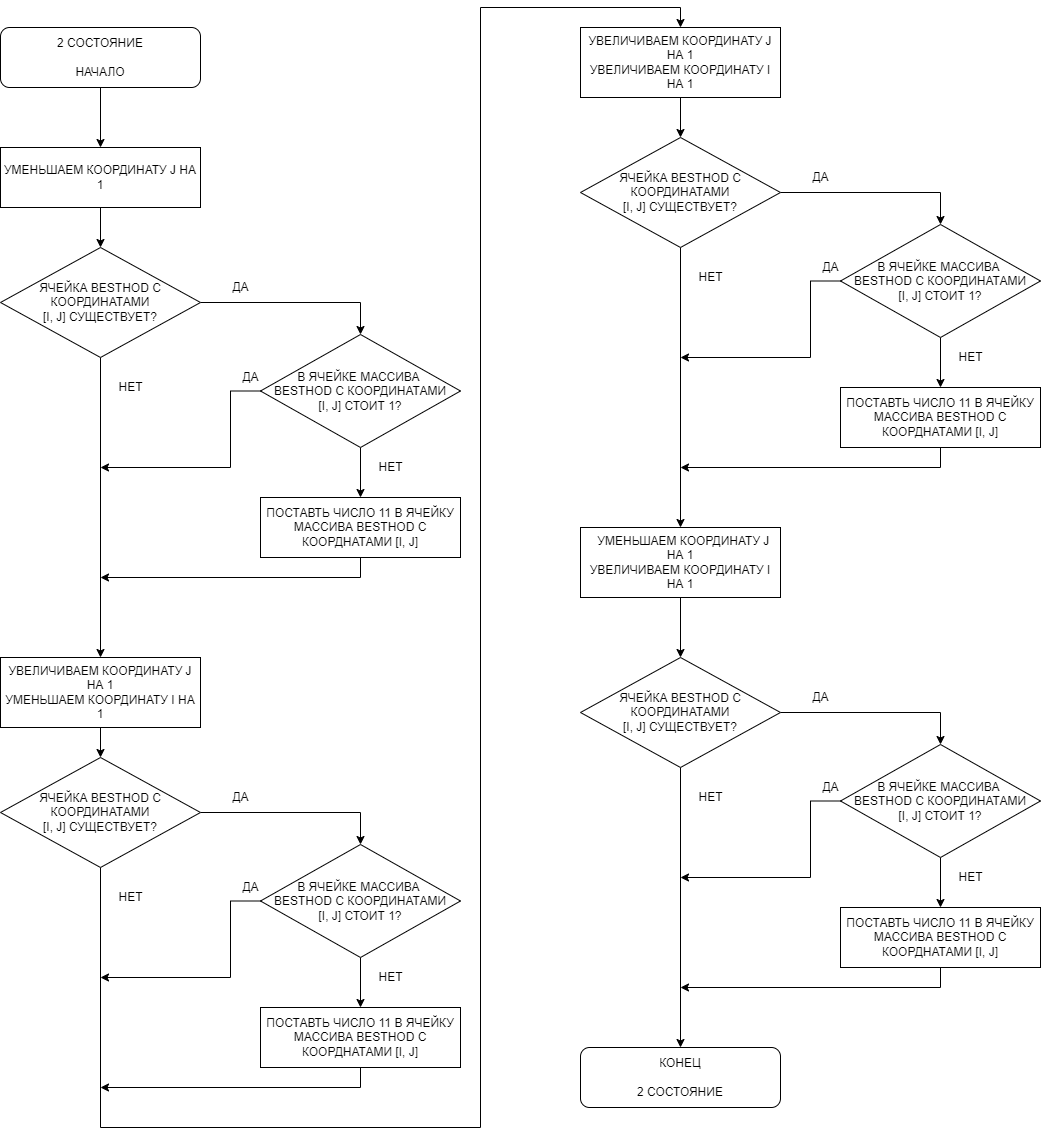
например, когда будем заполнять цифрами “1” соседние ячейки для выделенной клетки, получается, что в одной из соседних есть цифра “3”. Для следующей клетки цифры “3” уже будут стоять в 2-х соседних (слева и справа)-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 6 | 8 |
| 1 | 3 | 3 | 3 | 5 |
| 1 | 1 | 1 | 8 | 7 |

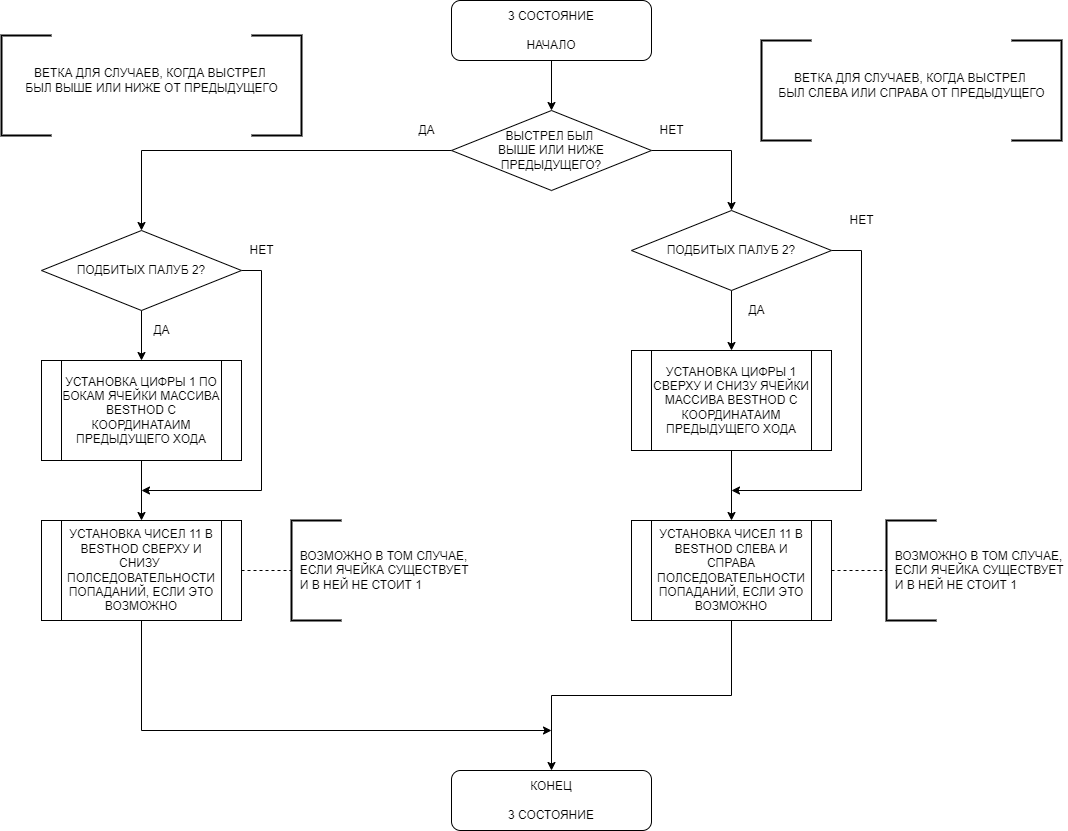
**1 состояние:**



**2 состояние:**



**3 состояние:**



По поводу 3 состояния: выстрел будет один. По левой ветке будет выстрел сначала выше последовательности. В случае попадания выстрел снова будет выше. Если же был промах, то следующий выстрел будет ниже последовательности. Для правой ветки алгоритм аналогичный, только бот будет стрелять сначала по клеткам левее последовательности до первого промаха, а затем по правым. Данная процедура не возвращает координаты для выстрела, она только меняет значения в массиве BESTHOD, а далее вызывается процедура «СОСТОЯНИЕ 1». Она берет первое встреченное число “11” и возвращает координаты этой клетки, по ним и будет выстрел. Для левой ветки числа будут стоять выше и ниже последовательности, а для правой левее и правее последовательности. Одновременно (друг за другом) обе ветви выполняться не могут.

**Листинг программы:**

**type**

TMapRow = **array of** integer;

TMap = **array of** TMapRow;

TCoordinates = **array of** integer;

TCoord = **record**

i: integer;

j: integer;

**end**;

**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)

**var**

setnumber: integer = 0;

ii, jj: integer;

nom: integer = 1;

damaged: **array** [1..10] **of** TCoord;

Besthod: **array**[0..9, 0..9] **of** byte;

hodi: integer;

hodj: integer;

flag, f: boolean;

fll: boolean;

k: integer;

i, j, ori, orj: integer;

**procedure** setParameters(setCount: integer);

**begin**

**end**;

**procedure** onGameStart();

**begin**

**end**;

**procedure** onSetStart();

**begin**

nom := 1;

ori := 0;

orj := 0;

**for** i := 1 **to** 10 **do**

**begin**

damaged[i].i := 0;

damaged[i].j := 0;

**end**;

hodnumber := 1;

f := false;

flag := false;

besthod[0, 0] := 5; besthod[0, 1] := 8; besthod[0, 2] := 9; besthod[0, 3] := 10; besthod[0, 4] := 7; besthod[0, 5] := 5; besthod[0, 6] := 7; besthod[0, 7] := 10; besthod[0, 8] := 5; besthod[0, 9] := 9;

besthod[1, 0] := 5; besthod[1, 1] := 5; besthod[1, 2] := 10; besthod[1, 3] := 7; besthod[1, 4] := 6; besthod[1, 5] := 6; besthod[1, 6] := 10; besthod[1, 7] := 9; besthod[1, 8] := 8; besthod[1, 9] := 7;

besthod[2, 0] := 7; besthod[2, 1] := 10; besthod[2, 2] := 6; besthod[2, 3] := 5; besthod[2, 4] := 5; besthod[2, 5] := 10; besthod[2, 6] := 6; besthod[2, 7] := 7; besthod[2, 8] := 6; besthod[2, 9] := 10;

besthod[3, 0] := 10; besthod[3, 1] := 9; besthod[3, 2] := 6; besthod[3, 3] := 5; besthod[3, 4] := 9; besthod[3, 5] := 8; besthod[3, 6] := 5; besthod[3, 7] := 5; besthod[3, 8] := 10; besthod[3, 9] := 5;

besthod[4, 0] := 9; besthod[4, 1] := 5; besthod[4, 2] := 7; besthod[4, 3] := 10; besthod[4, 4] := 8; besthod[4, 5] := 5; besthod[4, 6] := 7; besthod[4, 7] := 10; besthod[4, 8] := 9; besthod[4, 9] := 5;

besthod[5, 0] := 8; besthod[5, 1] := 7; besthod[5, 2] := 10; besthod[5, 3] := 6; besthod[5, 4] := 6; besthod[5, 5] := 9; besthod[5, 6] := 10; besthod[5, 7] := 8; besthod[5, 8] := 7; besthod[5, 9] := 6;

besthod[6, 0] := 3; besthod[6, 1] := 10; besthod[6, 2] := 8; besthod[6, 3] := 8; besthod[6, 4] := 7; besthod[6, 5] := 10; besthod[6, 6] := 9; besthod[6, 7] := 6; besthod[6, 8] := 5; besthod[6, 9] := 10;

besthod[7, 0] := 10; besthod[7, 1] := 5; besthod[7, 2] := 6; besthod[7, 3] := 7; besthod[7, 4] := 10; besthod[7, 5] := 8; besthod[7, 6] := 8; besthod[7, 7] := 9; besthod[7, 8] := 10; besthod[7, 9] := 8;

besthod[8, 0] := 7; besthod[8, 1] := 9; besthod[8, 2] := 7; besthod[8, 3] := 10; besthod[8, 4] := 6; besthod[8, 5] := 6; besthod[8, 6] := 5; besthod[8, 7] := 10; besthod[8, 8] := 8; besthod[8, 9] := 5;

besthod[9, 0] := 5; besthod[9, 1] := 8; besthod[9, 2] := 10; besthod[9, 3] := 5; besthod[9, 4] := 8; besthod[9, 5] := 7; besthod[9, 6] := 10; besthod[9, 7] := 7; besthod[9, 8] := 6; besthod[9, 9] := 7;

setnumber := setnumber + 1;

**if** setnumber > 6 **then**

setnumber := 1;

**end**;

**function** getMap(): TMap;

**begin**

**if** setnumber = 1 **then begin**

result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

**end**;

**if** setnumber = 2 **then begin**

result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

**end**;

**if** setnumber = 3 **then begin**

result := TMap.Create(

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

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

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

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

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

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

**end**;

**if** setnumber = 4 **then begin**

result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

**end**;

**if** setnumber = 5 **then begin**

result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

**end**;

**if** setnumber = 6 **then begin**

result := TMap.Create(

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

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

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

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

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

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

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

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

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

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

**end**;

**end**;

**function** shoot(): TCoordinates;

**var**

i, j, maxi, maxj: integer;

**begin**

**begin**

maxi := 0;

maxj := 0;

fll := false;

**for** i := 0 **to** 9 **do**

**begin**

**for** j := 0 **to** 9 **do**

**begin**

**if** Besthod[i, j] > Besthod[maxi, maxj] **then begin**

maxi := i;

maxj := j;

**end**;

**end**;

**end**;

hodi := maxi;

hodj := maxj;

**end**;

result := TCoordinates.Create(hodi, hodj);

**end**;

**procedure** shotResult(resultCode: integer);

**begin**

**if** resultcode = 0 **then**

**begin**

ii := hodi;

jj := hodj;

Besthod[ii, jj] := 1;

**end**;

**if** resultcode = 2 **then**

**begin**

**if** f = true **then**

nom := nom + 1;

ii := hodi;

jj := hodj;

damaged[nom].i := hodi;

damaged[nom].j := hodj;

Besthod[hodi, hodj] := 2;

**if** nom = 1 **then begin**

ori := hodi;

orj := hodj;

jj := jj - 1;

**if** (ii >= 0) **and** (ii <= 9) **and** (jj >= 0) **and** (jj <= 9) **and** (Besthod[ii, jj] <> 1) **then**

Besthod[ii, jj] := 9;

jj := jj + 1;

ii := ii - 1;

**if** (ii >= 0) **and** (ii <= 9) **and** (jj >= 0) **and** (jj <= 9) **and** (Besthod[ii, jj] <> 1) **then**

Besthod[ii, jj] := 9;

jj := jj + 1;

ii := ii + 1;

**if** (ii >= 0) **and** (ii <= 9) **and** (jj >= 0) **and** (jj <= 9) **and** (Besthod[ii, jj] <> 1) **then**

Besthod[ii, jj] := 9;

jj := jj - 1;

ii := ii + 1;

**if** (ii >= 0) **and** (ii <= 9) **and** (jj >= 0) **and** (jj <= 9) **and** (Besthod[ii, jj] <> 1) **then**

Besthod[ii, jj] := 9;

**end**;

**if** nom > 1 **then begin**

**if** (hodi > ori) **then begin** //выстрелы вниз

**for** j := 0 **to** 9 **do**

**for** i := 0 **to** 9 **do**

**begin**

**if** (i <> ori - 1) **and** (j <> hodj) **and** (Besthod[i, j] = 9) **then**

Besthod[i, j] := 1;

**if** ((i = hodi + 1) **and** (j = orj)) **and** (Besthod[i, j] <> 1) **then**

Besthod[i, j] := 9;

**end**;

**end**;

**if** (hodi < ori) **then begin** //выстреллы вверх

**for** i := 0 **to** 9 **do**

**for** j := 0 **to** 9 **do**

**begin**

**if** ((i <> ori + 1) **and** (j <> orj) **and** (Besthod[i, j] = 9)) **then**

Besthod[i, j] := 1;

**if**(i = hodi - 1) **and** (j = hodj) **and** (Besthod[i, j] <> 1) **then**

Besthod[i, j] := 9;

**end**;

**end**;

**if** (hodj < orj) **then begin** //выстрелы влево

**for** i := 0 **to** 9 **do**

**for** j := 0 **to** 9 **do**

**begin**

**if** ((i <> hodi) **and** (j <> orj + 1) **and** (Besthod[i, j] = 9)) **then**

Besthod[i, j] := 1;

**if** ((i = hodi) **and** (j = hodj - 1)) **and** (Besthod[i, j] <> 1) **then**

Besthod[i, j] := 9;

**end**;

**end**;

**if** (hodj > orj) **then begin**

**for** i := 0 **to** 9 **do**

**for** j := 0 **to** 9 **do**

**begin**

**if** ((i <> hodi) **and** (j <> orj - 1) **and** (Besthod[i, j] = 9)) **then**

Besthod[i, j] := 1;

**if** ((i = hodi) **and** (j = hodj + 1)) **and** (Besthod[i, j] <> 1) **then**

Besthod[i, j] := 9;

**end**;

**end**;

**end**;

f := true;

**end**;

**if** resultcode = 3 **then**

**begin**

ii := hodi;

jj := hodj;

Besthod[ii, jj] := 2;

deletehod(ii, jj);

**if** (f = true) **then begin**

**for** k := 1 **to** nom **do**

**begin**

ii := damaged[k].i;

jj := damaged[k].j;

Bestmap[ii, jj] := 2;

deletehod(ii, jj);

**end**;

f := false;

nom := 1;

**end**;

**end**;

**end**;

**procedure** deletehod(i: integer; j: integer);

**begin** //заполнение массива ходов(куда нельзя стрелять)

i := i - 1; j := j - 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

j := j + 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

j := j + 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

i := i + 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

i := i + 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

j := j - 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

j := j - 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

i := i - 1;

**if** (i >= 0) **and** (i <= 9) **and** (j >= 0) **and** (j <= 9) **and** (Besthod[i, j] <> 2) **then**

Besthod[i, j] := 1;

**end**;

**procedure** onOpponentShot(cell: TCoordinates);

**begin**

**end**;

**procedure** onSetEnd();

**begin**

**end**;

**procedure** onGameEnd();

**begin**

**end**;

**end**.

**Вывод:**

В ходе лабораторной работы был реализован алгоритм работы бота для игры “Морской бой”.