Инструкция по написанию бота для Морского боя

1 Правила и терминология

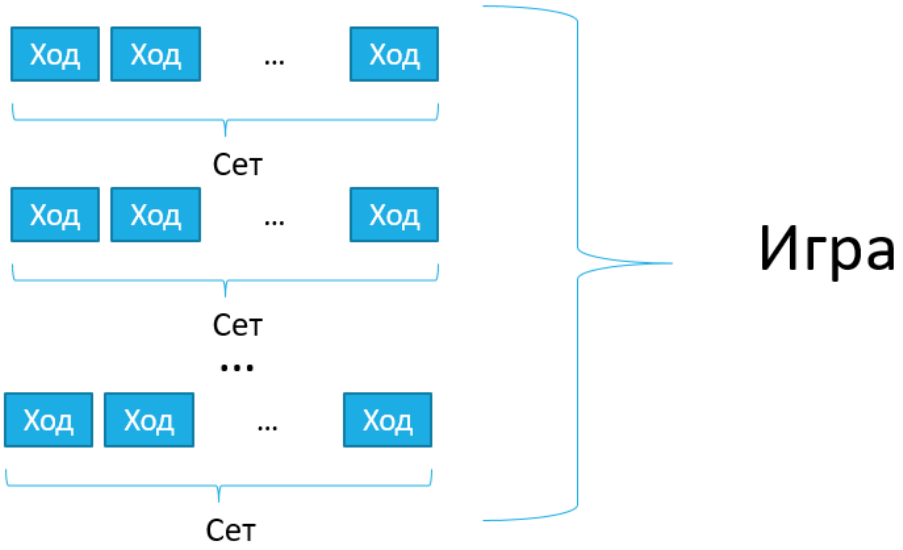
Используются стандартные правила морского боя: поле боя 10x10 клеток, на котором размешаются корабли - один 4-палубник, два 3-палубника, три 2-палубника, четыре 1-палубника. Корабли состоят из клеток и имеют форму строго прямой линии. Корабли не могут касаться друг друга, в том числе – по диагонали.

В самом начале игрок не знает ничего о расположении кораблей противника. Игроки по очереди стреляют в клетки на полях друг друга. Результатом выстрела могут быть «промах», «ранение» (попадание в клетку корабля, после которого у него остались «живые» клетки), «убийство» (попадание в клетку корабля, после которого у корабля не остается «живых» клеток). Цель игры – уничтожить все корабли противника.

Игроку запрещается стрелять снова в те клетки, в которые он уже стрелял. Не запрещается стрелять в клетки, граничащие с убитыми кораблями, хотя такие ходы тоже не имеют смысла.

Если результат выстрела – не «промах», следующий выстрел делает тот же самый игрок. Иначе ход переходит к противнику.

Наименьшая часть игры – ход (один выстрел). Последовательность ходов, в результате которой один из игроков побеждает, уничтожив все корабли противника – сет. Игра состоит из нескольких сетов. Для каждого сета игрок составляет новое поле с кораблями (которое может совпадать с предыдущими, а может и отличаться от них).



Для проведения игры определяется один параметр – количество сетов. Номер игрока, который производит первый выстрел в сете, чередуется. Один сет обязательно заканчивается победой одного из игроков.

Очевидно, что при нечетном количестве сетов в игре обязательно будет победитель. При четном количестве сетов игра может закончиться ничьей.

1.1 Правила турнира

Турнир проводится по тем же правилам, что и «Камень-Ножницы-Бумага».

Каждый бот играет с каждым другим ботом одну игру. Победитель получает 3 очка, проигравший – 0 очков. При ничье оба получают по 1 очку. Очки, полученные по результатам всех игр, суммируются. Чем больше сумма очков у бота, тем выше он находится в турнирной таблице. Если у двух и более ботов одинаковое количество очков, они разделяют одно место.

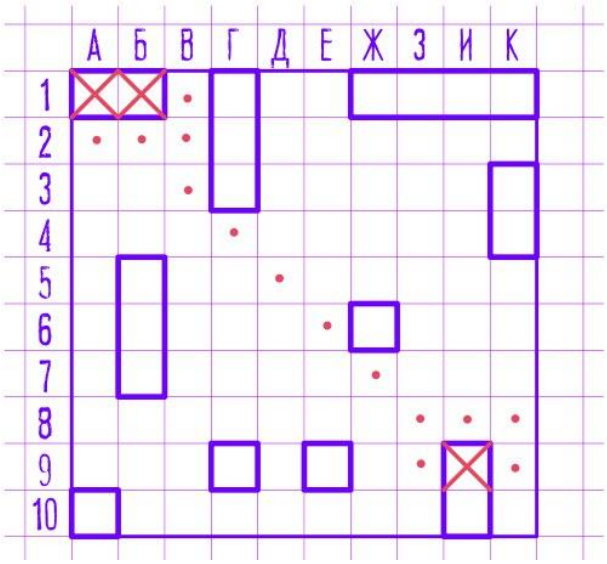
Бот запускается перед началом всех игр и завершает свою работу после окончания всех игр (либо может досрочно при возникновении ошибок в процессе турнира). То есть, между играми память бота не теряется.

Исключительные ситуации:

- если бот даёт некорректный ответ во время хода (неправильные координаты; неправильно построенная карта; выстрел по клетке, в которую бот уже стрелял) он считается проигравшим в игре, но из турнира не исключается;
- если бот неожиданно завершается во время игры, либо превышает установленные лимиты по времени выполнения функций (процедур, методов), он исключается из турнира.

1.2 Система координат

Координаты клетки задаются парой (x, y) начиная от левого верхнего угла поля; счет клеток идет от 0 до 9. Ось X направлена слева направо, ось Y – сверху вниз. Например, на рисунке ниже ход A2 будет записан как (0, 1); ход B8 – как (2, 7).



2 Шаблон для бота

Здесь и далее для примера будет использован язык Pascal.

На следующей странице показан шаблон бота для Морского боя.

```
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

function getMap: TByteArray2D;
begin
end;

function shoot: TByteArray;
begin
end;

procedure shootResult(resultCode: integer);
begin
end;

procedure rivalShoot(point: TByteArray);
begin
end;

procedure setParams(setsPerGame: integer);
begin
end;

procedure onCurrentSetEnd;
begin
end;

procedure onCurrentGameEnd;
begin
end;

end.
```

2.1 Функция getMap

Вызывается в начале каждого сета. Бот должен вернуть карту в виде двумерного массива байт размером 10x10. Каждая клетка массива *result[Y][X]* должна содержать 0, если она пуста (константа *EMPTY*) или 1, если это клетка корабля (константа *BLOCK*).

Время выполнения – не более 5 секунд.

2.2 Функция shoot

Вызывается каждый раз, когда бот должен сделать выстрел. Возвращаемое значение – одномерный массив байт длиной 2, в нулевой клетке которого записана координата X, в первой – Y.

Время выполнения – не более 1 секунды.

2.3 Процедура shootResult

Вызывается после каждого выстрела рассматриваемого бота. Имеет один аргумент – *resultCode*, описывающий результат выстрела. 0 – «Промах» (константа *EMPTY*), 2 – «Ранение» (константа *DAMAGE*), 3 – «Убийство» (константа *KILL*).

В том случае, если результат вызова *shoot* не был корректен (координаты за пределами поля; выстрел по клетке, в которую уже бот стрелял; неверный размер массива), *shootResult* вызвана не будет.

Время выполнения – не более 1 секунды.

2.4 Процедура rivalShoot

Вызывается после каждого выстрела противника. Имеет один аргумент – point – двумерный массив байт, содержащий координаты (X, Y) выстрела противника.

В том случае, если выстрел противника не был корректен (координаты за пределами поля; выстрел по клетке, в которую уже бот стрелял; неверный размер массива), *rivalShoot* вызвана не будет.

Время выполнения – не более 1 секунды.

2.5 Процедура setParams

Вызывается перед началом каждой игры. Через неё боту передается единственный параметр – *setsPerGame*, определяющий количество сетов в игре. Несмотря на то, что *setParams* вызывается перед каждой игрой, значение *setsPerGame* постоянно для турнира.

Время выполнения – не более 5 секунд.

2.6 Процедура onCurrentSetEnd

Вызывается при завершении каждого сета. Если рассматриваемый бот или бот противника совершил ошибку во время сета (например, некорректный выстрел), *onCurrentSetEnd* вызвана не будет.

Время выполнения – не более 5 секунд.

2.7 Процедура onCurrentGameEnd

Вызывается при завершении каждой игры. Если рассматриваемый бот совершил ошибку во время сета (например, некорректный выстрел), *onCurrentGameEnd* вызвана не будет. Если бот противника допустил подобную ошибку, *onCurrentGameEnd* у рассматриваемого бота будет вызвана.

Исходя из названных особенностей методов *onCurrentSetEnd* и *onCurrentGameEnd*, действия, необходимые для выполнения перед каждым сетом, лучше всего делать в *getMap*, а необходимые перед каждой игрой – в *setParams*.

Однако, вы можете использовать их по своему усмотрению, если это окажется полезным для вашего алгоритма.

Время выполнения – не более 5 секунд.

3 Пример бота

Ниже показан пример самого простого бота, который всегда использует одну и ту же карту и случайным образом обстреливает поле противника. С помощью массива флагов *visited* гарантируется, что стрелять в одни и те же клетки он не будет, несмотря на случайность выбора координат выстрела.

```
unit SeaFightBotImpl;

{$mode objfpc}{$H+}

interface

type

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

const

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

var

    visited: array[0..99] of Boolean;
    leftCells: Integer;
    firstGame: Boolean = true;

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

implementation

function getMap: TByteArray2D;
var
    i: Integer;
begin
    // Инициализация
    leftCells := 100; // Количество непосещенных клеток противника
    for i := 0 to 99 do begin
        visited[i] := false; // Флаги о посещении клеток
    end;

    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
    pos, counter, cur: Integer;
begin
    // Выбирается номер случайной непосещенной клетки из оставшихся
    pos := random(leftCells);

    // Эта клетка ищется на поле
    // Поле нумеруется с нуля слева направо, сверзу вниз
    // counter - нумерация только непосещенных клеток
    // cur - нумерация всех клеток
    counter := -1;
    cur := -1;
    while counter <> pos do begin
        inc(cur);
        // cur div 10 - координата Y, cur mod 10 - координата X.
        if (visited[(cur div 10) * 10 + cur mod 10] = false) then inc(counter);
    end;

    visited[(cur div 10) * 10 + cur mod 10] := true;
    result := TByteArray.Create(cur mod 10, cur div 10);
end;

procedure shootResult(resultCode: integer);
begin
    dec(leftCells);
end;

procedure rivalShoot(point: TByteArray);
begin
end;

procedure setParams(setsPerGame: integer);
var
    i: Integer;
begin
    if firstGame then begin
        randomize;
        firstGame := false;
    end;
end;

procedure onCurrentSetEnd;
begin
end;

procedure onCurrentGameEnd;
begin
end;

end.
```