**Игра «Охота на лис»**

**Суть игры**

1. На поле случайным неизвестным для игрока образом расставляются «лисы», причём несколько из них могут находиться в одной и той же клетке.
2. Игрок целится с помощью стрелок и стреляет с помощью клавиши Enter. В ответ он получает количество «лис», которое пеленгуется из его нынешнего местоположения. Это число указывает, сколько лис расположено в одной вертикали и горизонтали с указанной клеткой.
3. Если местоположение игрока совпало с положением «лисы», она считается найденной.
4. Игра продолжается, пока не будут найдены все «лисы».

**План разработки игры**

Создать консольное приложение, моделирующее игру «Охота на лис».

1. Создать меню

* Игра
* Правила игры
* Выход

1. Перемещение по таблице

Пользователь сначала вводит размер поля (не больше 10х10) и количество лис (не больше (размер поля \* размер поля)). Перемещение только по таблице, но можно выйти из игры.

1. Игра

Количество выстрелов ограниченно, вычисляются по формуле:

размер поля \* (размер поля - 1) + 1

1. Ведется подсчет затраченного времени и количество выстрелов
2. При каждом выстреле выводится справка:

* Нашли ли вы лис;
* Сколько их там было;
* Сколько лис по пеленгу;
* Сколько лис осталось еще найти.
* Сколько выстрелов осталось

1. Таблица:

* По умолчанию ячейки зеленые;
* Ячейка, в которую стреляют – красная;
* Ячейки по пеленгу – желтые;
* Ячейки, в которых лис уже нашли – высвечивается «V».
* Ячейки, в которых лис нет – высвечивается «X».

**Спецификация структуры Cell**

Описание структуры: в структуре находятся два поля:

* int CounOfFoxes – количество лис в одной ячейке;
* ShotInCell FindFoxs – нашел ли игрок лису.

Перечисление ShotInCell:

* NoAction – игрок не проверял ячейку;
* Find – игрок нашел лис;
* NotFind – игрок не нашел лис.

**Спецификация структуры UserActions**

Описание структуры: в структуре реализована работа с действиями пользователя.

Методы:GetUserAction().

Перечисление Action (действия, которые может совершить пользователь):

* NoAction – не совершал действия
* Left – игрок нажал стрелочку влево
* Right – игрок нажал стрелочку вправо
* Exit – игрок нажал клавишу Escape
* Top – игрок нажал стрелочку вверх
* Bottom – игрок нажал стрелочку вниз
* Enter – игрок нажал клавишу Enter

**Описание методов структуры**

public static Action GetUserAction() – в зависимости от того какую кнопку нажал игрок, выполняется действие (движение влево, вправо, вверх, вниз, Enter и Esc);

**Спецификация структуры ViewTable**

Описание структуры: в структуре реализована работа с видом таблицы.

Поля: int size – размер поля ;

Методы:ViewTable, SetTableCursorPosition, DrawField, Peleng, CursorPosition

**Описание методов структуры**

public ViewTable(int Size) – конструктор;

* Size – размер поля;

public void CursorPosition(int cursorX, int cursorY) – перемещение курсора по таблице;

* cursorХ – координата по горизонтали
* cursory – координата по вертикали

private void SetTableCursorPosition(int row, int column) - реализована формула которая переводит обычные координаты в экранные;

* row – координата выбранной ячейки по горизонтали;
* column – координата выбранной ячейки по вертикали;

public void DrawField(Table tb) – рисует таблицу;

* tb – программная таблица;

public void Peleng(int targetRow, int targetColumn, Table tb) – рисует пеленг точки (желтым цветом) и саму точку в которую попали (красным цветом);

* targetRow – координата выбранной ячейки по горизонтали;
* targetColumn – координата выбранной ячейки по вертикали;
* tb – программная таблица;

**Спецификация структуры Table**

Описание структуры: в структуре реализованы вычисления и работа с таблицей

Поля:

* public int leftFoxes – количество лис надо найти;
* public int findNowFoxes – количество лис в определенной ячейке;
* public int cursorX – координата по горизонтали;
* public int cursory – координата по вертикали;
* public Cell[,] Foxes – двумерный массив ячеек таблицы;
* public int countFindFoxes – количество найденных лис;
* public int shots – заданное количество выстрелов;
* public int countShots – сделанное количество выстрелов;
* public int countPellingFoxes – количество лис по пеленгу.

Методы:MoveCursore, EnterFoxes

**Описание методов структуры**

public Table(int size, int countOfFoxes) **–** конструктор**;**

* size – размер поля;
* countOfFoxes – количество лис;

public void MoveGameCursore (int dx, int dy, ViewTable tb) – движение курсора по игровому полю;

* dx – сдвиг курсора по горизонтали;
* dy – сдвиг курсора по вертикали;
* tb – пользовательская таблица (видимое игровое поле);

public GameState EnterFoxes(ViewTable tb, ref int ch) – определяется нашел ли пользователь лису, сколько выстрелов уже было сделано, какую информацию надо выводить на экран, количество лис по полену определенной точки, количество уж найденных лис, количество лис, которых еще надо найти;

* tb – пользовательская таблица (видимое игровое поле);
* result – если result: GameState.Miss – не нашли лису, GameState.Hit – нашли лису, GameState.HitFirstCell – попали в клетку где находится игрок, GameState. Win – игрок победил, GameState.Lose – игорок проиграл;

**Спецификация структуры ProgramGame**

Описание структуры: в структуре реализованы вычисления и работа с таблицей

Поля:

* public static int countOfFoxes – количество всех лис;
* public static GameState stateOfGame – состояние игры ;
* public static int start – начало отсчета времени;
* public static bool showPeleng – показывать пеленг или нет;
* public static int sizeOfTable – размер поля;
* public const int minSizeTable = 2 – минимальный размер поля;
* public const int maxSizeTable = 10 – максимальный размер поля;
* public const int minCountOfFoxs = 2 – минимальное количество лис;

Методы:GameSetUp, RandomArray, Game, GameExit, GameWinMoveYesNo

Перечисление GameState – состояние игры.

* NoAction – не происходит дейсвия,
* Win – пользователь выиграл игру,
* Lose – пользователь проиграл игру,
* Hit – пользователь нашел лис в ячейке,
* Miss – пользователь не нашел лис в ячейке,
* HitFirstCell – пользователь выстрелил в первую клетку

**Описание методов структуры**

public static ProgramSate GameSetUp(ref ViewTable view, ref Table playingField) – найстройки для начала игры;

* view – видимое игровое поле;
* playingField – программная таблица;

public static ProgramSate Game(string strExit, string strWinGame, ref ProgMenu pm, ref ViewTable view, ref MenuConsole mn, ref Table playingField) – управление движением по таблице в зависимости от того, что нажал пользователь, перемещение между окнами;

* strExit – рисунок окна выхода;
* strWinGame – рисунок окна Победы/Проигрыша;
* pm – программное меню;
* view – видимое игровое поле;
* mn – меню отображаемое на консоли;
* playingField – программное игровое поле.

public static void RandomArray(ref Table playingField) – двумерный массив заполняется случайными числами;

* playingField – программная таблица;

private static ProgramSate GameExit(string strExit, ref ProgMenu pm, ref MenuConsole mn, ref Table playingField, ViewTable view) – управление в окне выхода из игры: можно выйти в меню или продолжить игру.

* pm – программное меню;
* mn – меню отображаемое на консоли;
* playingField – программное игровое поле.
* view – видимое игровое поле;

public static ProgramSate GameWinMoveYesNo(ref ProgMenu pm, ref MenuConsole mn, ref Table playingField) – управление в окне Победы/Поражение: можно выйти в меню или начать новую игру.

* pm – программное меню;
* mn – меню отображаемое на консоли;
* playingField – программное игровое поле.

**Спецификация структуры ConsoleGame**

Описание структуры: вывод информации об игре.

Методы:GameInfo, WinGame

**Описание методов структуры**

public static void GameInfo(ViewTable view, Table playingField, GameState stateOfGame)– вывод информации на экран при нажатии Enter во время игры;

* view – видимое игровое поле;
* playingField – программное игровое поле;
* stateOfGame – пользовательская таблица (видимое игровое поле);

public static ProgramSate WinGame(string strWin, ref ProgMenu pm, ref MenuConsole mn, ref Table playingField, GameState stateOfGame) – вывод информации на экран, если пользователь победил/проиграл.

* strWin – рисунок окна Победы/Поражения
* pm – программное меню;
* mn – меню отображаемое на консоли;
* playingField – программное игровое поле;
* stateOfGame – пользовательская таблица (видимое игровое поле);

**Спецификация структуры ProgMenu**

Описание структуры: в структуре реализована работа с движением по меню или по кнопкам Да/Нет.

Поля:

* public int cursorMenu – положение курсора в меню;
* public int cursorYesNo – положение курсора в окне Победы/Поражения на кнопках Да/Нет;
* public int cursorYesNoExit – положение курсора в окне Выхода на кнопках Да/Нет;

Методы:MoveMenu, MoveYesNo, MoveYesNoExit

Перечисления:

Menu – кнопки меню

* NoAction
* Game – игра
* Reference – справка
* Exit – выход

YesNo – кнопки Да/Нет

* NoAction
* Yes – кнопка Да
* No – кнопка Нет

**Описание методов структуры**

public void MoveMenu(MenuConsole mc, int dy) – движение курсора по меню;

* mc – Меню на консоле
* dy – смещение по вертикали среди пунктов меню;

public void MoveYesNo(MenuConsole mc, int dx)– движение курсора по кнопках Да/Нет в окне Победы/Поражения;

* mc – Меню на консоле
* dx – смещение по горизонтали среди кнопок «Да» и «Нет»;

public void MoveYesNoExit(MenuConsole mc, int dx) – движение курсора по кнопкам Да/Нет в окне Выхода

* mc – Меню на консоле
* dx – смещение по горизонтали среди кнопок «Да» и «Нет»;

**Спецификация структуры MenuControl**

Описание структуры: в структуре реализовано движение по меню, в зависимости от того, что нажал пользователь.

Методы:Menu

**Описание методов структуры**

public static ProgramSate Menu(string strMenu, ref ProgMenu pm, ref MenuConsole mn, ref Table playingField) – управление перемещением курсора по пунктам меню;

* strMenu – рисунок окна Меню
* pm – программное меню;
* mn – меню отображаемое на консоли;
* playingField – программное игровое поле.

**Спецификация структуры MenuConsole**

Описание структуры: в структуре реализовано движение по меню, в зависимости от того, что нажал пользователь.

Методы:ConsoleMenu, SelectItemMenu, ConsoleYesNo, SelectYesNo

**Описание методов структуры**

public void ConsoleMenu() – отображение кнопок меню на консоли;

public void SelectItemMenu(Menu mn) – отображение выбранной кнопки меню на консоли

* mn – меню отображаемое на консоли;

public void ConsoleYesNo(int x, int y) – отображение кнопок Да/Нет

* x – смещение по горизонтали
* y – смещение по вертикали

public void SelectYesNo(YesNo ch, int x, int y) – отображение выбранной кнопки Да/Нет

* ch – выбранный пункт меню
* x – смещение по горизонтали
* y – смещение по вертикали

**Спецификация структуры MenuControl**

Описание структуры: в структуре реализована работа с движением по меню или по кнопкам Да/Нет.

Методы:MoveMenu, MoveYesNo, MoveYesNoExit

**Описание методов структуры**

public void MoveMenu(MenuConsole mc, int dy) – движение курсора по меню;

* mc – Меню на консоле
* dy – смещение по вертикали среди пунктов меню;

public void MoveYesNo(MenuConsole mc, int dx)– движение курсора по кнопках Да/Нет в окне Победы/Поражения;

* mc – Меню на консоле
* dx – смещение по горизонтали среди кнопок «Да» и «Нет»;

public void MoveYesNoExit(MenuConsole mc, int dx) – движение курсора по кнопкам Да/Нет в окне Выхода

* mc – Меню на консоле
* dx – смещение по горизонтали среди кнопок «Да» и «Нет»;

**Спецификация структуры UserReference**

Описание структуры: вывод на консоль окна справки

Методы:Reference

**Описание методов структуры**

public static ProgramSate Reference(string strReference, ref ViewTable view, ref Table playingField) – отображение на консоль окна справки;

* strReference – картинка окна справки
* view – видимое игровое поле;
* playingField – программное игровое поле;

**Спецификация структуры ProgramReference**

Описание структуры: Выход из справки

Методы:ExitReference

**Описание методов структуры**

public static ProgramSate ExitReference() – реализует выход из справки;