Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт

**Отчёт**

по МДК.02.01. Технология разработки программного обеспечения

на тему: создание игры «Две лисы и двадцать кур»

Исполнители:Дувакин Андрей Андреевич,

Зырянова Елизавета Петровна, студенты группы ИСпП-21-2

Преподаватель: Коржина В. С.

**Техническое задание**

На поле указанной формы находятся две лисы и 20 кур. Куры могут перемещаться на один шаг вверх, влево или вправо, но не назад и не по диагонали. Лисы также могут перемещаться на один шаг (вверх, вниз, влево и вправо).

Лиса может съесть курицу: если в горизонтальном или вертикальном направлении за курицей на один шаг следует свободное поле, то лиса перепрыгивает через курицу и берет ее.

Лисы обязаны есть, и когда у них есть выбор, они обязаны осуществлять "наиболее длинное поедание". Если два приема пищи имеют одинаковую длину, осуществляется один из них - по выбору лисы.Составить программу, которая играет за лис. Игрок перемещает кур. Партнеры играю по очереди, причем куры начинают.

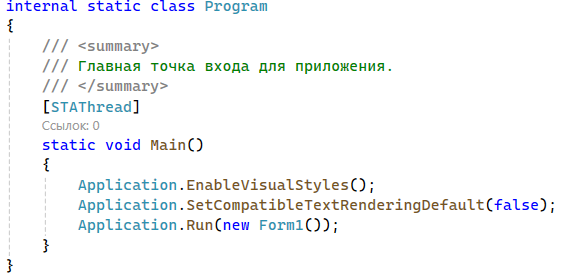
Они выигрывают партию, если девяти из них удается занять 9 полей, образующий верхний квадрат поля.Начальное положение кур и лис изображено на рисунке. Лисы выигрывают, если им удается съесть 12 кур, т.к. тогда оставшихся кур недостаточно, чтобы занять 9 верхних полей  
Детали на <https://studlearn.com/works/details/igra-dve-lisy-i-20-kur-s-247>

**Описание работы программы**

Приложение представляет собой игру на поле 7x7, где куры ("c") должны избегать лис ("f") и достичь верхней части поля для победы.

**Инициализация приложения (Листинг 1):**

Этот листинг содержит точку входа в приложение. Здесь устанавливаются параметры отображения приложения и создается экземпляр формы Form1, которая представляет основное окно игры.



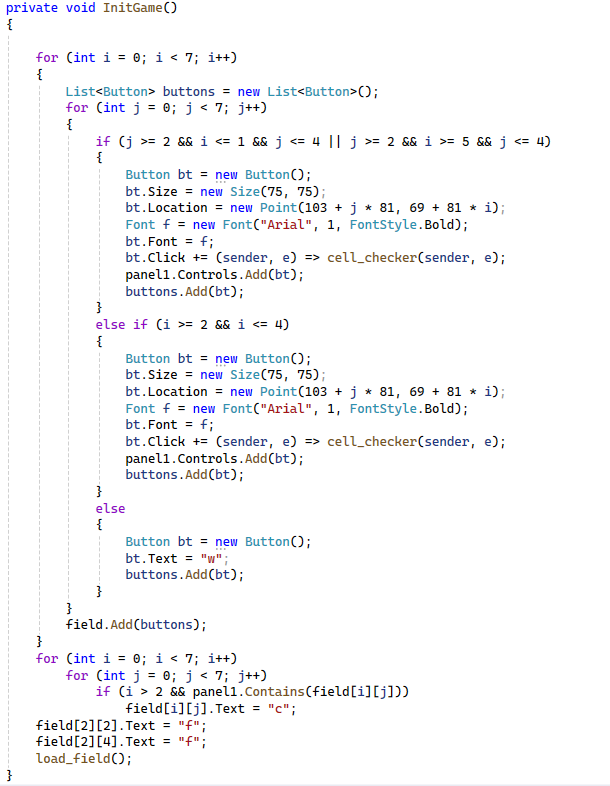
**Инициализация формы (Листинг 2):**

Инициализирует форму Form1, когда она создается. Вызывается конструктор формы, который включает инициализацию компонентов формы (InitializeComponent()) и метод InitGame(). Объявляются переменные для хранения кнопок поля (field), активной кнопки (active\_button) и пути к изображениям (path).



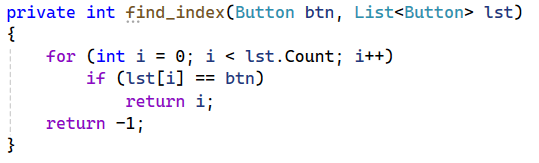
**Инициализация игры (Листинг 3):**

Метод InitGame() выполняет инициализацию игрового поля и размещает на нем начальные объекты. Создаются кнопки для игрового поля с помощью вложенных циклов и условий. Устанавливаются начальные позиции кур и лис на игровом поле. Вызывается метод load\_field(), чтобы отобразить начальное состояние игрового поля. Где w (wall) - стена, кнопки с такой меткой не будут отображаться, с (chicken) - это курицы, а а (fox) - это лисы.



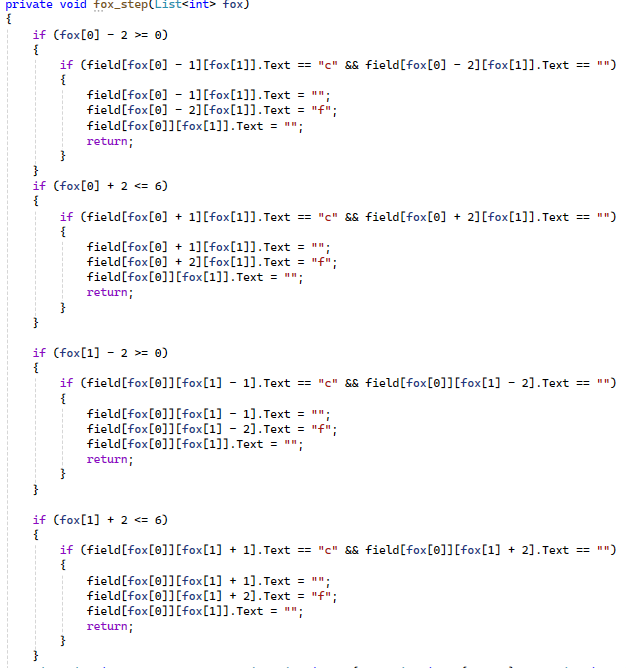
**Функция поиска индекса кнопки (Листинг 4):**

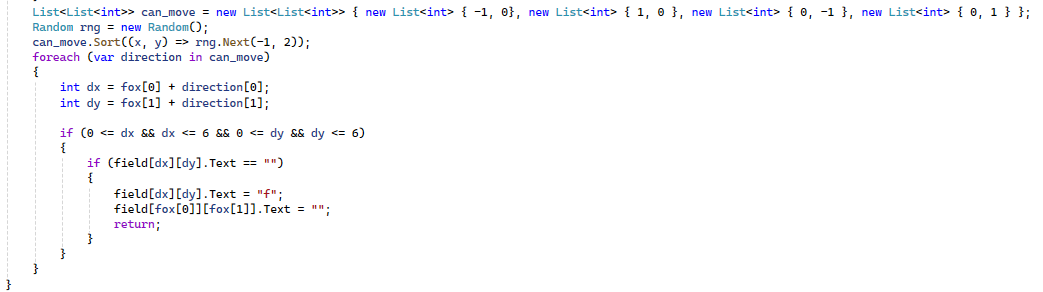
Метод find\_index(Button btn, List<Button> lst) принимает кнопку и список кнопок в качестве аргументов. Он выполняет поиск кнопки в списке и возвращает индекс кнопки в списке, если она найдена, и -1, если кнопка не найдена.



**Функция "хода" лисы (Листинг 5):**

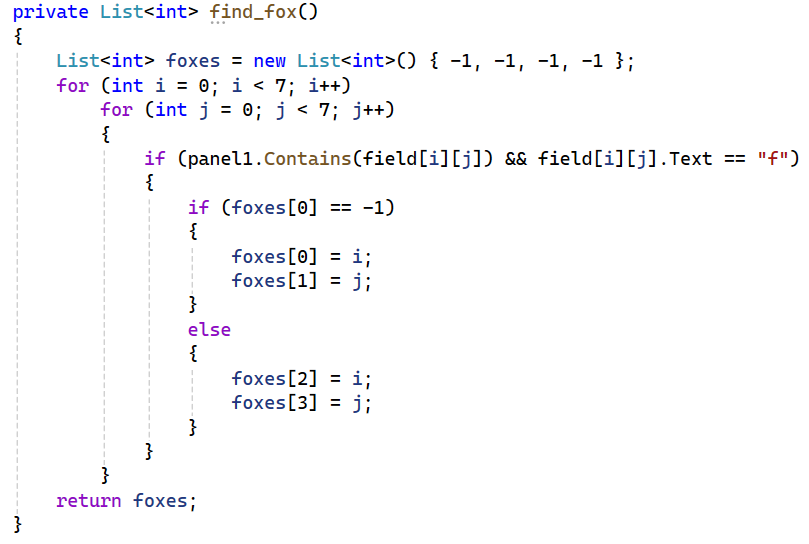
Метод fox\_step(List<int> fox) определяет логику движения лисы на игровом поле. Лиса пытается двигаться к ближайшей куре для съедения. Если кура находится в пределах двух клеток, лиса съедает ее. В противном случае лиса выбирает случайное направление движения.





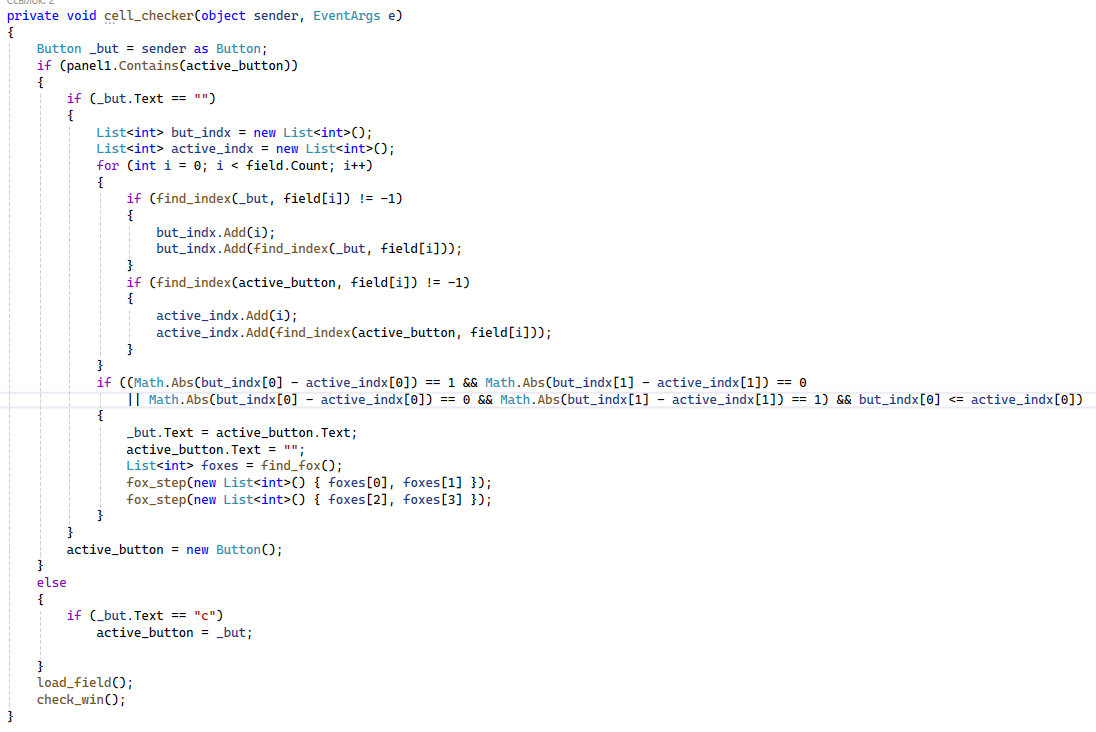
**Функция поиска лис в объектах поля (Листинг 6):**

Метод find\_fox() находит координаты лис на игровом поле. Он проходит по всем кнопкам поля и ищет лисы, сохраняя их координаты в списке. Возвращает список с координатами всех лис на поле.



**Функция обработки хода игрока (Листинг 7):**

Метод cell\_checker(object sender, EventArgs e) обрабатывает ходы игрока при нажатии на кнопки поля. При каждом нажатии на кнопку на игровом поле вызывается этот метод. Параметр sender представляет собой объект, который инициировал событие, в данном случае - кнопку. Метод проверяет, была ли нажата активная кнопка (хранится в переменной active\_button). Если кнопка была нажата, и нажата кнопка поля (не содержащая куры или лисы), то проверяется возможность перемещения куры на эту позицию. Если перемещение возможно, активная кнопка (кура) перемещается на выбранную позицию, и активной кнопке присваивается пустое значение. Затем происходит проверка наличия лис на поле и их ходов. Если кнопка не была нажата, и на ней находится кура, она становится активной кнопкой. После обработки хода игрока вызывается метод load\_field() для обновления отображения поля, а также метод check\_win() для проверки условия победы или поражения.

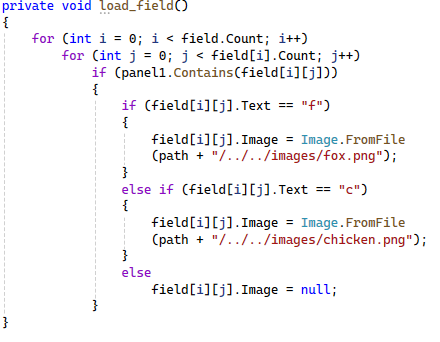


**Функция проверки условия для победы или поражения (Листинг 8):**

Метод check\_win() проверяет условие победы или поражения после каждого хода. Победа достигается, если все куры ("c") находятся в указанных позициях, занимая верхний лепесток креста поля. Поражение наступает, если количество оставшихся кур на поле меньше 9. Если условие победы выполнено, отображается сообщение о победе и создается новое окно игры. Если условие поражения выполнено, отображается сообщение о поражении и создается новое окно игры.

**Функция отрисовки изображений объект (Листинг 9):**

Метод load\_field() отвечает за отображение изображений на игровом поле в соответствии с их типом. Перебираются все элементы игрового поля, и для каждого элемента проверяется его тип (кура, лиса или пустая клетка). Если элемент - лиса, устанавливается изображение лисы. Если элемент - кура, устанавливается изображение куры. Если элемент - пустая клетка, изображение очищается. Это обновляет отображение игрового поля после каждого хода.



**Ссылка на репозиторий**

**Блок-схема**

Сюда вставляем блок-схему

**Приложение А**

**Листинг 1 (Инициализация приложения)**

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

**Листинг 2 (Инициализация формы)**

public Form1()

{

InitializeComponent();

InitGame();

}

List<List<Button>> field = new List<List<Button>>();

Button active\_button = new Button();

string path = System.IO.Path.GetDirectoryName(Application.ExecutablePath);

**Листинг 3 (Функция создания игрового поля и первоначальной расстановки объектов)**

private void InitGame()

{

for (int i = 0; i < 7; i++)

{

List<Button> buttons = new List<Button>();

for (int j = 0; j < 7; j++)

{

if (j >= 2 && i <= 1 && j <= 4 || j >= 2 && i >= 5 && j <= 4)

{

Button bt = new Button();

bt.Size = new Size(75, 75);

bt.Location = new Point(103 + j \* 81, 69 + 81 \* i);

Font f = new Font("Arial", 1, FontStyle.Bold);

bt.Font = f;

bt.Click += (sender, e) => cell\_checker(sender, e);

panel1.Controls.Add(bt);

buttons.Add(bt);

}

else if (i >= 2 && i <= 4)

{

Button bt = new Button();

bt.Size = new Size(75, 75);

bt.Location = new Point(103 + j \* 81, 69 + 81 \* i);

Font f = new Font("Arial", 1, FontStyle.Bold);

bt.Font = f;

bt.Click += (sender, e) => cell\_checker(sender, e);

panel1.Controls.Add(bt);

buttons.Add(bt);

}

else

{

Button bt = new Button();

bt.Text = "w";

buttons.Add(bt);

}

}

field.Add(buttons);

}

for (int i = 0; i < 7; i++)

for (int j = 0; j < 7; j++)

if (i > 2 && panel1.Contains(field[i][j]))

field[i][j].Text = "c";

field[2][2].Text = "f";

field[2][4].Text = "f";

load\_field();

}

**Листинг 4 (Функция поиска индекса кнопки)**

private int find\_index(Button btn, List<Button> lst)

{

for (int i = 0; i < lst.Count; i++)

if (lst[i] == btn)

return i;

return -1;

}

**Листинг 5 (Функция «хода» лисы)**

private void fox\_step(List<int> fox)

{

if (fox[0] - 2 >= 0)

{

if (field[fox[0] - 1][fox[1]].Text == "c" && field[fox[0] - 2][fox[1]].Text == "")

{

field[fox[0] - 1][fox[1]].Text = "";

field[fox[0] - 2][fox[1]].Text = "f";

field[fox[0]][fox[1]].Text = "";

return;

}

}

if (fox[0] + 2 <= 6)

{

if (field[fox[0] + 1][fox[1]].Text == "c" && field[fox[0] + 2][fox[1]].Text == "")

{

field[fox[0] + 1][fox[1]].Text = "";

field[fox[0] + 2][fox[1]].Text = "f";

field[fox[0]][fox[1]].Text = "";

return;

}

}

if (fox[1] - 2 >= 0)

{

if (field[fox[0]][fox[1] - 1].Text == "c" && field[fox[0]][fox[1] - 2].Text == "")

{

field[fox[0]][fox[1] - 1].Text = "";

field[fox[0]][fox[1] - 2].Text = "f";

field[fox[0]][fox[1]].Text = "";

return;

}

}

if (fox[1] + 2 <= 6)

{

if (field[fox[0]][fox[1] + 1].Text == "c" && field[fox[0]][fox[1] + 2].Text == "")

{

field[fox[0]][fox[1] + 1].Text = "";

field[fox[0]][fox[1] + 2].Text = "f";

field[fox[0]][fox[1]].Text = "";

return;

}

}

List<List<int>> can\_move = new List<List<int>> { new List<int> { -1, 0}, new List<int> { 1, 0 }, new List<int> { 0, -1 }, new List<int> { 0, 1 } };

Random rng = new Random();

can\_move.Sort((x, y) => rng.Next(-1, 2));

foreach (var direction in can\_move)

{

int dx = fox[0] + direction[0];

int dy = fox[1] + direction[1];

if (0 <= dx && dx <= 6 && 0 <= dy && dy <= 6)

{

if (field[dx][dy].Text == "")

{

field[dx][dy].Text = "f";

field[fox[0]][fox[1]].Text = "";

return;

}

}

}

}

**Листинг 6 (Функция поиска лис в объектах поля)**

private List<int> find\_fox()

{

List<int> foxes = new List<int>() { -1, -1, -1, -1 };

for (int i = 0; i < 7; i++)

for (int j = 0; j < 7; j++)

{

if (panel1.Contains(field[i][j]) && field[i][j].Text == "f")

{

if (foxes[0] == -1)

{

foxes[0] = i;

foxes[1] = j;

}

else

{

foxes[2] = i;

foxes[3] = j;

}

}

}

return foxes;

}

**Листинг 7 (Функция обработки хода игрока)**

private void cell\_checker(object sender, EventArgs e)

{

Button \_but = sender as Button;

if (panel1.Contains(active\_button))

{

if (\_but.Text == "")

{

List<int> but\_indx = new List<int>();

List<int> active\_indx = new List<int>();

for (int i = 0; i < field.Count; i++)

{

if (find\_index(\_but, field[i]) != -1)

{

but\_indx.Add(i);

but\_indx.Add(find\_index(\_but, field[i]));

}

if (find\_index(active\_button, field[i]) != -1)

{

active\_indx.Add(i);

active\_indx.Add(find\_index(active\_button, field[i]));

}

}

if ((Math.Abs(but\_indx[0] - active\_indx[0]) == 1 && Math.Abs(but\_indx[1] - active\_indx[1]) == 0 || Math.Abs(but\_indx[0] - active\_indx[0]) == 0 && Math.Abs(but\_indx[1] - active\_indx[1]) == 1) && but\_indx[0] <= active\_indx[0])

{

\_but.Text = active\_button.Text;

active\_button.Text = "";

List<int> foxes = find\_fox();

fox\_step(new List<int>() { foxes[0], foxes[1] });

fox\_step(new List<int>() { foxes[2], foxes[3] });

}

}

active\_button = new Button();

}

else

{

if (\_but.Text == "c")

active\_button = \_but;

}

load\_field();

check\_win();

}

**Листинг 8 (Функция проверки условия для победы или поражения)**

private void check\_win()

{

if (field[0][2].Text == "c" && field[0][3].Text == "c" && field[0][4].Text == "c" && field[1][2].Text == "c" && field[1][3].Text == "c" && field[1][4].Text == "c" && field[2][2].Text == "c" && field[2][3].Text == "c" && field[2][4].Text == "c")

{

MessageBox.Show("Поздравляем, вы победили");

Form1 form1 = new Form1();

form1.Show();

this.Visible = false;

}

int chicken\_count = 0;

for (int i = 0; i < 7; i++)

for (int j = 0; j < 7; j++)

if (field[i][j].Text == "c")

chicken\_count++;

if (chicken\_count < 9)

{

MessageBox.Show("К сожалению, вы проиграли");

Form1 form1 = new Form1();

form1.Show();

this.Visible = false;

}

}

**Листинг 9(Функция отрисовки изображений объект)**

private void load\_field()

{

for (int i = 0; i < field.Count; i++)

for (int j = 0; j < field[i].Count; j++)

if (panel1.Contains(field[i][j]))

{

if (field[i][j].Text == "f")

{

field[i][j].Image = Image.FromFile

(path + "/../../images/fox.png");

}

else if (field[i][j].Text == "c")

{

field[i][j].Image = Image.FromFile

(path + "/../../images/chicken.png");

}

else

field[i][j].Image = null;

}

}