**Лабораторная работа 3. 1 часть**

**Игровое приложение «Пятнашки»**

Создайте Приложение Windows Forms с именем FifteenGUI в решении с именем FifteenGame.

Задайте форме следующие свойства

|  |  |
| --- | --- |
| Text | Пятнашки |
| StartPosition | CenterScreen |

Переименуйте имя формы с Form1 на Fifteen.

Измените иконку приложения – свойство Icon. Для этого необходимо изображение с расширением .ico. Кроме этого иконку нужно прописать для exe-файла. Для этого в свойствах проекта выберите Application и в разделе Icon and manifest выберите файл с расширением .ico еще раз.

Положите на форму элемент MenuStrip. Задайте ему имя menu. Создайте пункт с текстом Начать игру (рис. 1). Переименуйте этот элемент его в startMenu.

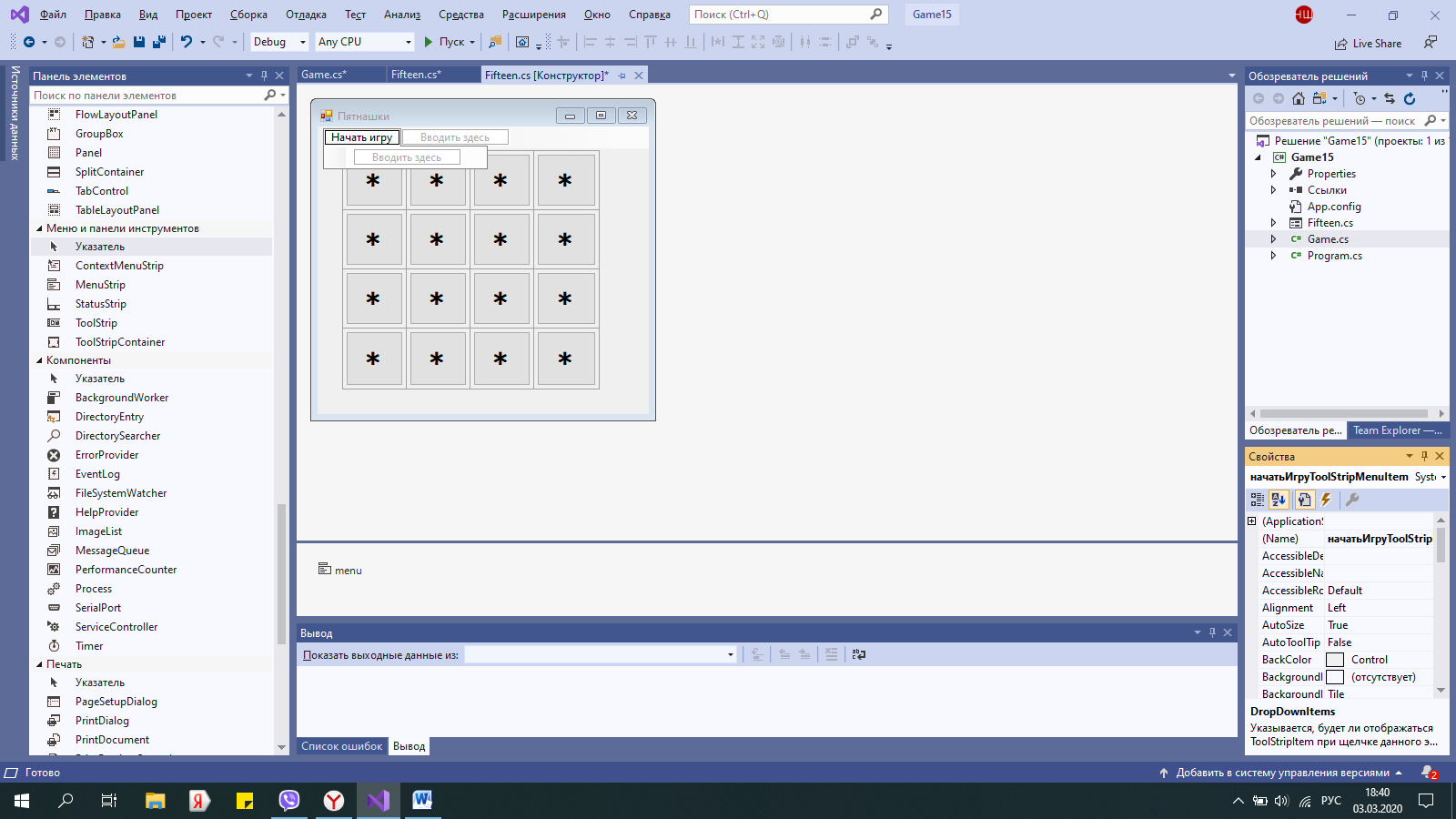


Рисунок 1. Добавление пункта меню

Разместите на форме элемент TableLayoutPanel. Задайте ему размер 4 строки на 4 столбца. Раскройте меню Правка строк и столбцов (рис. 2)

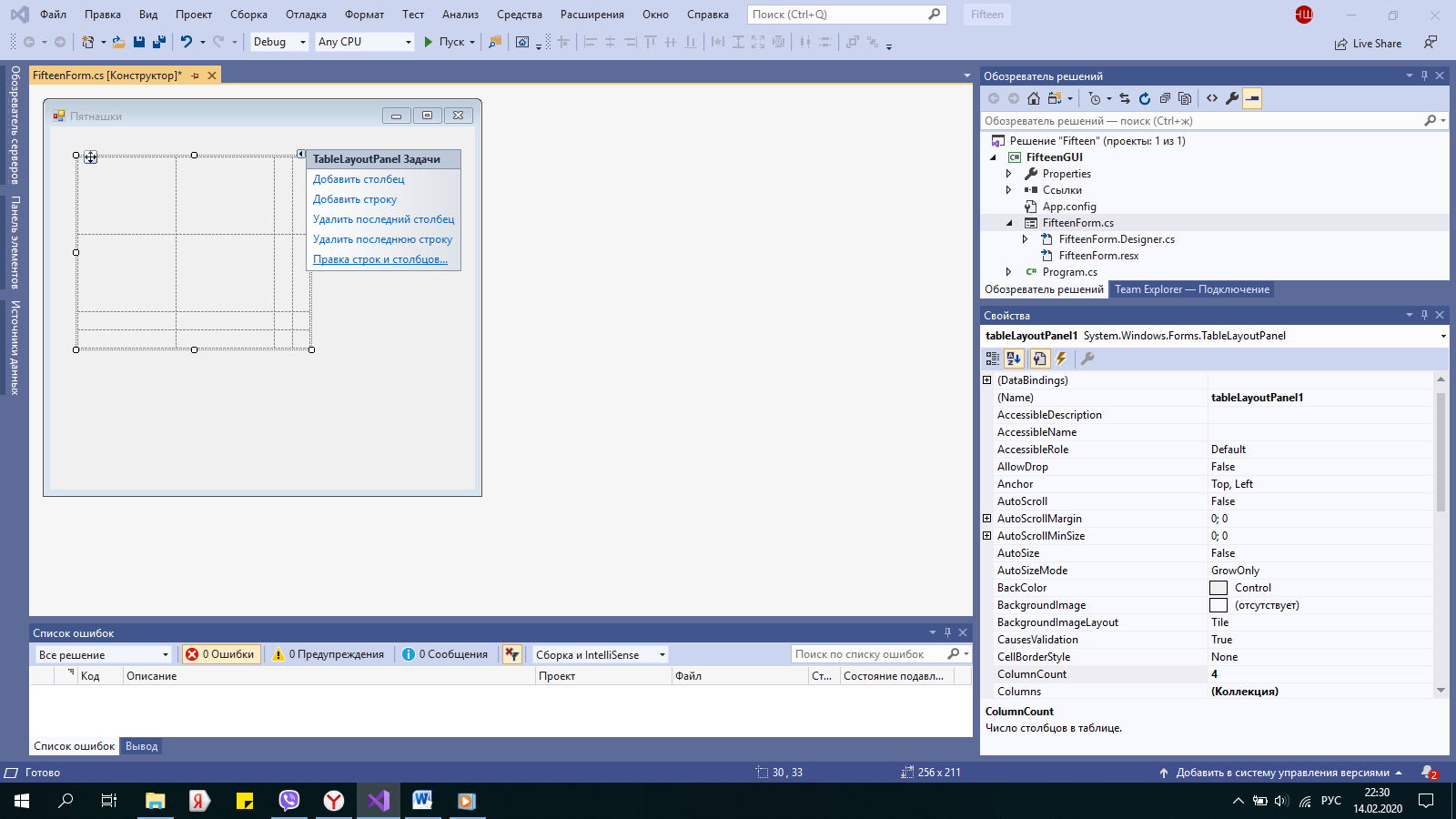


Рисунок 2. Меню TableLayoutPanel Задачи.

Выделите все столбцы и укажите, чтобы все проценты у них были одинаковы (рис. 3). Аналогичное действие выполните и для строк.

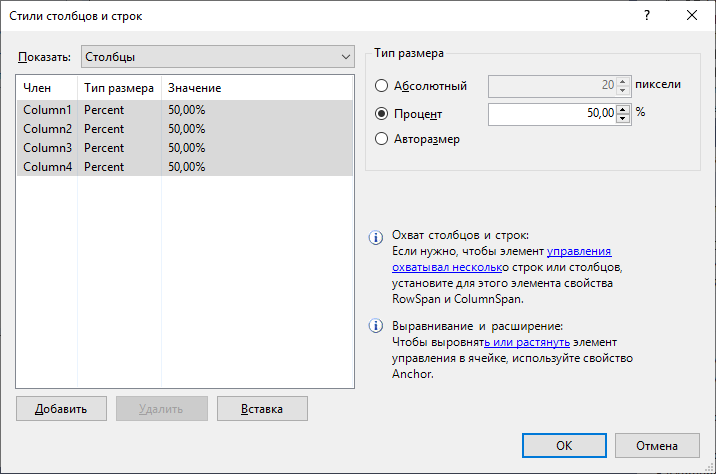


Рисунок 3. Меню стилей столбцов и строк.

Установите свойство Dock элемента TableLayoutPanel в значение Fill (по центру). Убедитесь, что свойство FormBorderStyle формы имеет значение Sizable.

В итоге должно получиться следующее размеченное поле (рис. 4).

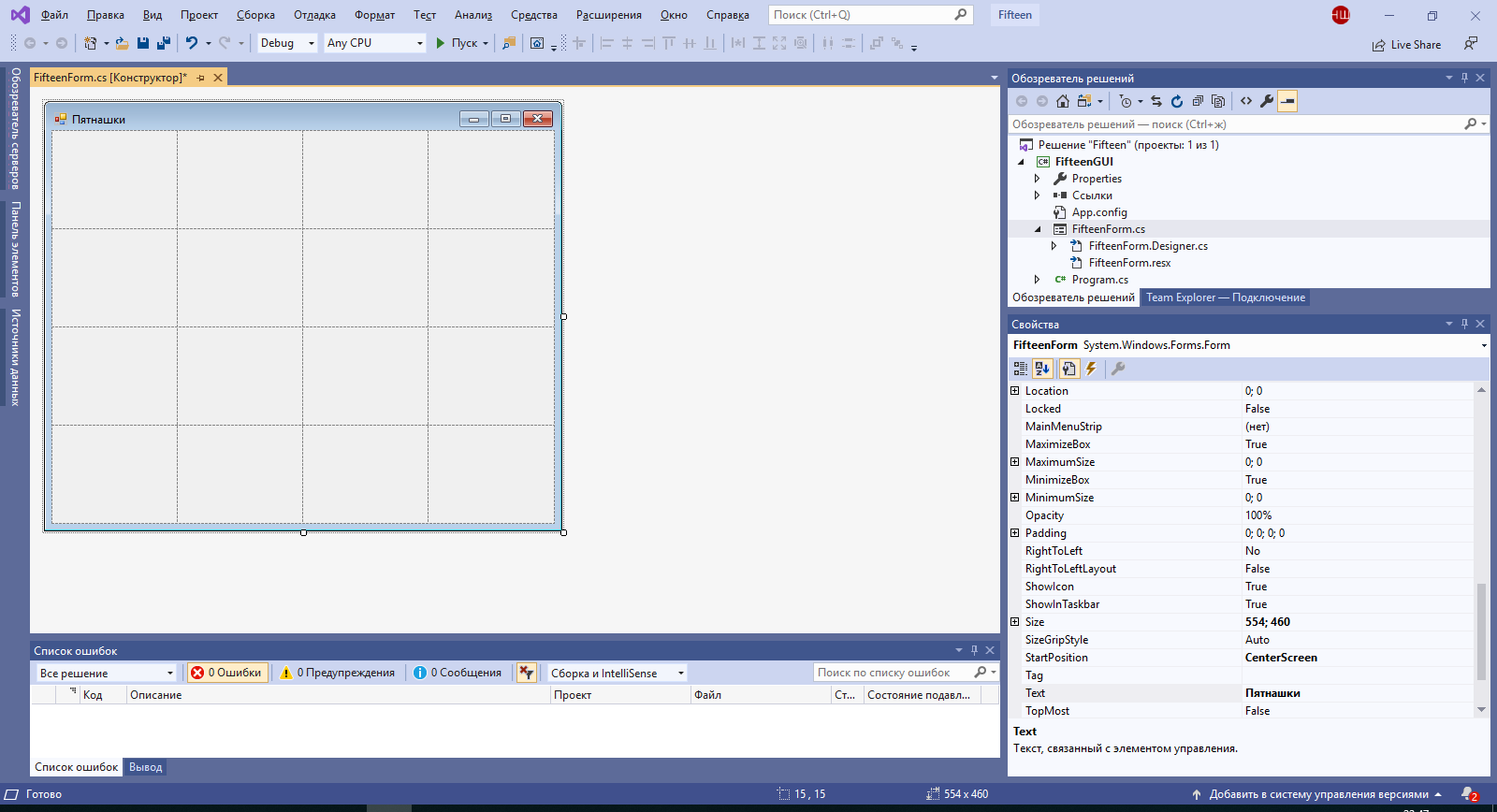


Рисунок 4. Размещение TableLayoutPanel на FitteenForm.

Поместите в ячейку с номером (0,0) кнопку. Установите кнопке следующие свойства: Dock в значение Fill, Name в значение button0, Text в значение '\*'. Измените значения свойства Font по своему усмотрению. Задайте кнопкам свойство Tag в соответствии с номером в имени (от 0 до 15).

Скопируйте и вставьте полученную кнопку во все ячейки таблицы. Нумерация в именах кнопок будет проставлена автоматически.

**Запуск**. Запустите проект и попробуйте изменить размеры формы. Размеры содержимого изменяются вместе с размерами формы (спасибо элементу TableLayoutPanel).

Создайте метод, который по номеру возвращает кнопку (одну из шестнадцати созданных):

private Button GetButton(int index)

Добавьте к проекту класс с именем Game. В нем будет реализована логика игры.

Моделью игрового поля является массив. Будем использовать двумерный массив:

int[,] field;

int size;

Опишите конструктор с параметром, в котором задайте значение полю size и выделите память под массив field.

Для обозначения координат будем использовать переменные x и y (названия координатных осей в математике). Например, кнопка с номером 7 в массиве будет иметь координаты: x=3, y=1 (рис. 5).

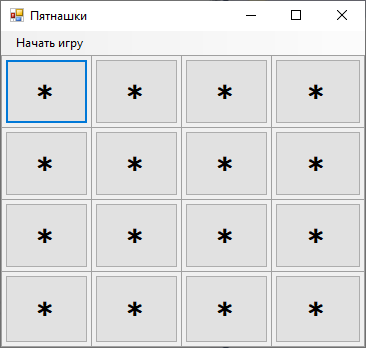


Рисунок 5. Игровое поле

Так как на форме кнопки нумеруются от 0 до 15, а модель поля двумерная, то необходим метод преобразования координат двумерного массива в номер кнопки:

private int CoordinatesToPosition(int x, int y)

и обратно:

private void PositionToCoordinates(int position, out int x,out int y)

Необходимы будут методы для создания начальной расстановки чисел на поле, метод обмена значениями, когда игрок делает ход, проверки на окончание игры.

Создайте метод Start () для начальной расстановки игры:

public void Start()

В этом методе нужно заполнить поле field значениями от 1 до 16. Для получения этих значений легче всего использовать вызов метода

CoordinatesToPosition(x, y)

где x, y –это параметры циклов для просмотра двумерного массива.

Для координат нуля (пустого поля в игре) объявите два поля:

int x0, y0;

Значения 16 замените на 0, используя введенные координаты.

Пригодится функция, которая по позиции нажатой кнопки должна будет вернуть значение, хранящееся в соответствующей ячейке поля:

public int GetNumber(int position)

{

int x, y;

PositionToCoordinates(position, out x, out y);

return field[x, y];

}

В эту функцию можно добавить проверку корректности значений координат x, y, если они некорректны, например меньше 0, возвращать 0.

Вернитесь к описанию класса Fifteen.

Опишите поле Game game;

В конструкторе формы после вызова метода InitializeComponent() вызовите конструктор для создания game.

Добавьте в класс формы метод обновления поля кнопок. В этом методе необходимо пробежать по всем номерам кнопок и обновить их текстовое поле в соответствии со значением в поле field класса Game:

private void RefreshButtonField()

{

for (int poition = 0; poition < 16; poition++)

GetButton(poition).Text = game.GetNumber(poition).ToString();

}

Создайте обработчик события нажатия на меню Начать игру. Вызовите в нем game.Start() и метод RefreshButtonField().

**Запуск**. Запустите и протестируйте приложение.

При нажатии на меню появляется заполненное поле. Сделайте так, чтобы поле заполнялось при появлении формы. Заполнение нужно выполнить в событии Load формы. Получается, что в двух методах будет написано одно и то же. Это не хорошо. Выделите повторяющиеся действия в отдельный метод:

private void GameStart()

В методе RefreshButtonField сделайте кнопку с 0 невидимой. Используйте свойство .

**Запуск**. Запустите и проверьте автоматическое заполнение.

При нажатии на кнопку, находящуюся рядом с пустым полем, значение, записанное на ней, меняется местами с 0. Это выглядит так, как будто кнопки поменялись местами. Это действие касается механики игры.

В классе Game опишите метод

public void Shift(int position)

В нем по параметру position найдите координаты x, y той ячейки массива, которую нужно поменять с нулевой ячейкой.

Останется только поменять их местами:

field[x0, y0] = field[x, y];

field[x, y] = 0;

Не забудьте присвоить переменных x0, y0 новые значения.

Вызов описанного метода необходимо выполнить в обработчике события Click каждой кнопки. Создайте обработчик для кнопки с номером 0. В нем запишите:

int position = Convert.ToInt32(((Button)sender).Tag);

game.Shift(position);

RefreshButtonField();

Параметр sender – это объект, который вызвал событие. Нужно привести его к кнопке и взять свойство Tag. Свойство Tag отвечает за позицию кнопки.

**Запуск**. Запустите и убедитесь в том, что любая кнопка может поменяться местами с пустой. Это неправильное поведение.

Нужно перемещать на пустое место только соседнюю кнопку. Исправьте это поведение в методе Shift.

**Запуск**. Запустите и протестируйте приложение.

Далее нужно добавить перемешивание.

1 способ. КАК НЕ НАДО ДЕЛАТЬ! Перемешать массив значений. Но при этом с одинаковой вероятностью можно получить расстановку, которую можно собрать и расстановку, которую собрать нельзя!

2 способ. Можно перемешать исходное верно заполненное игровое поле, выполняя большое число ходов.

3 способ. Любителям Алексея Савватеева посвящается <https://www.youtube.com/watch?v=toCqrVmQPiY>, <https://www.youtube.com/watch?v=rZvQW4bBzfk>.

Реализация второго способа. Создайте в классе игры статическое поле rand (сразу же вызовите конструктор) класса Random и метод:

public void ShiftRandom()

Создайте в нем переменную a, которая принимает случайное значение от 0 до 3 включительно. Эта переменная нужна для того, чтобы можно было менять местами только соседние элементы, не произвольные.

Сохраните координаты пустой клетки в переменных x, y соответственно. Далее в зависимости от выбранного значения переменной a будет выполнен ход на одну из четырех соседних клеток, то есть либо переменная x изменит свое значение на 1 (вправо или влево), либо переменная y изменит свое значение на 1 (вниз или вверх). Последней строкой в данном методе нужно выполнить вызов:

Shift(CoordinatesToPosition(x,y));

Вызовите этот метод в классе формы, в методе GameStart, 100 или более раз.

**Запуск**. Запустите приложение. Вы должны получить ошибки.

Метод ShiftRandom вносит в программу новую ошибку. Уточните, в каком месте появляется эта ошибка, и исправьте ее.

**Запуск**. Запустите приложение. Оно должно работать без сбоев.

Осталось написать метод проверки завершения игры. В классе Game создайте метод:

public bool Check()

Вначале необходимо проверить, стоит ли пустая ячейка в правой нижней клетке. Если пустая ячейка стоит в другом месте, то функция точно возвращает false. В противном случае нужно проверить правильность расстановки кнопок на всем поле. Используя вложенные циклы по x и y, выполните следующую проверку: если клетка не последняя, она должна удовлетворять следующему условию:

if (field[x, y] == CoordinatesToPosition(x, y) + 1)

При невыполнении этого условия функция снова возвращает false. И в конце функция должна вернуть true.

Проверка выигрыша

if (game.Check())

MessageBox.Show("Победа");

должна выполняться после каждого хода, то есть после каждого нажатия на кнопку.

Задание.

1. Что должно быть выполнено в программе после выигрыша? Придумайте и реализуйте сценарий.
2. Введите счетчик ходов и придумайте как его разместить на форме.
3. Введите таймер и придумайте как его разместить на форме.