

# ЛАБОРАТОРНАЯ РАБОТА №6 (ЗАДАНИЕ 1)

Технологии искусственного интеллекта. Разработка экспертной системы.

## ЭКСПЕРТНЫЕ СИСТЕМЫ

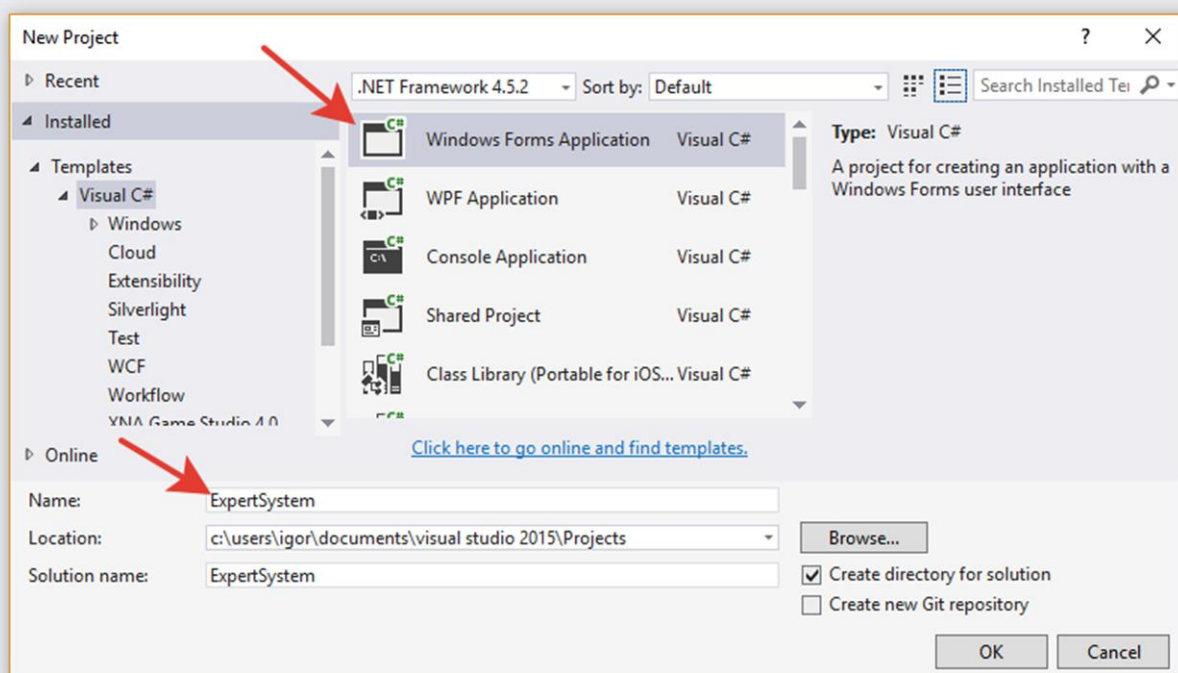
**Экспертные системы** это направление исследований в области искусственного интеллекта по созданию вычислительных систем, умеющих принимать решения, схожие с решениями экспертов в заданной предметной области.

**Экспертная система**, которую мы будем разрабатывать в рамках лабораторных работ 11 и 12, будет напоминать известный сайт «Акинатор». Она будет определять вид животного на основе серии вопросов, задаваемых пользователю.

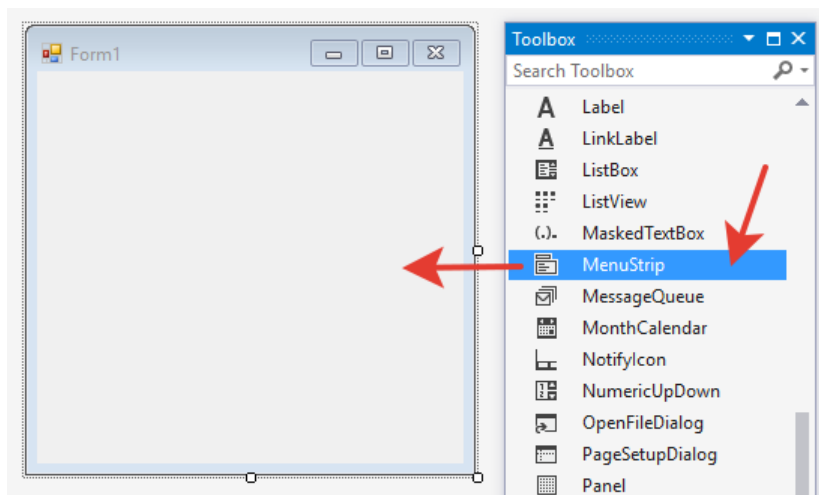
Лабораторная работа №11 посвящена созданию интерфейса программы.

## РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

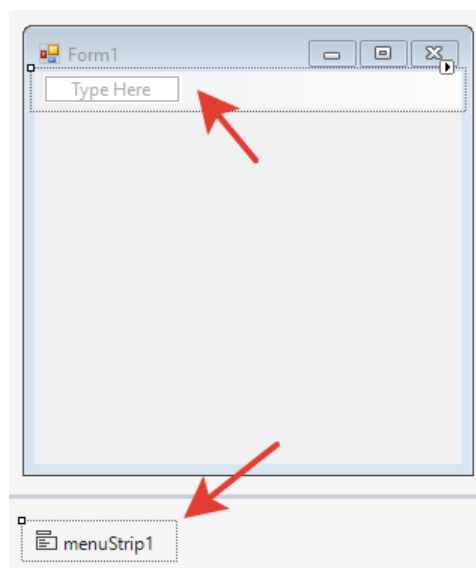
Запустите **Visual Studio** и создайте новый проект **Windows Forms**. Назовите новый проект **ExpertSystem**.



В окне конструктора форм добавьте на форму Form1 меню. Для этого перетащите элемент **MenuStrip** из панели инструментов в окно формы:

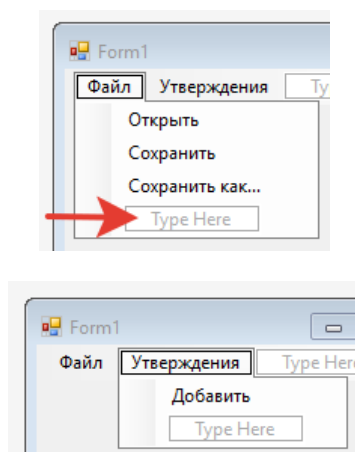


После добавления меню на форме появилась полоса редактирования. Также, в нижней части окна конструктора форм, открылась новая панель со списком невидимых объектов. Как ни странно, меню тоже относится к этой категории.

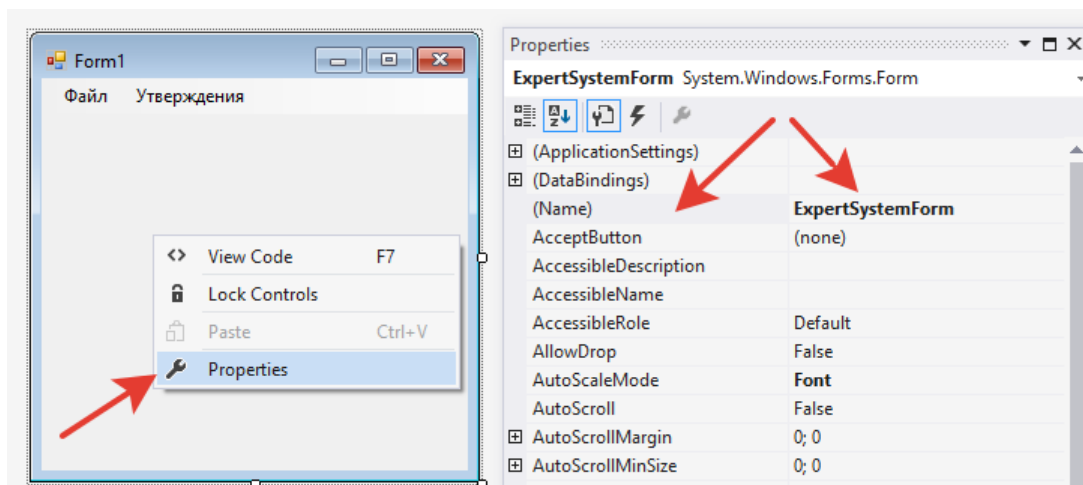


Для изменения свойств меню необходимо выбрать объект menuStrip1 в нижней части окна и, нажав на нем правой кнопкой мыши, выбрать пункт всплывающего меню «Свойства» (**Properties**). Переименуем наш объект из menuStrip1 в **mainMenu**.

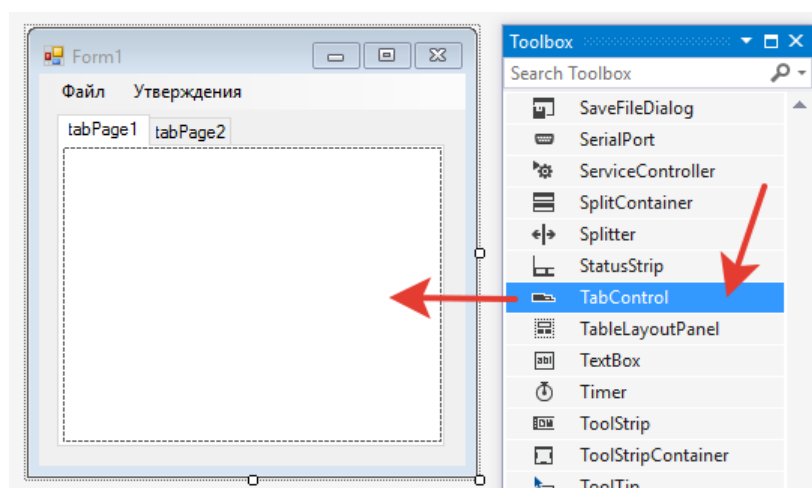
Для добавления нового пункта меню достаточно нажать в пустом прямоугольном поле и ввести имя пункта. Создайте меню следующего вида:



Переименуем нашу форму. Нажмите правой кнопкой мыши в центре формы **Form1** и выберите пункт меню «Свойства». Измените параметр «Имя» (**Name**) на **ExpertSystemForm**.

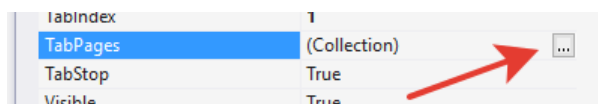


Добавим на форму еще один элемент управления – панель вкладок **TabControl**. По аналогии с меню, найдите его на панели инструментов и перетащите на форму:

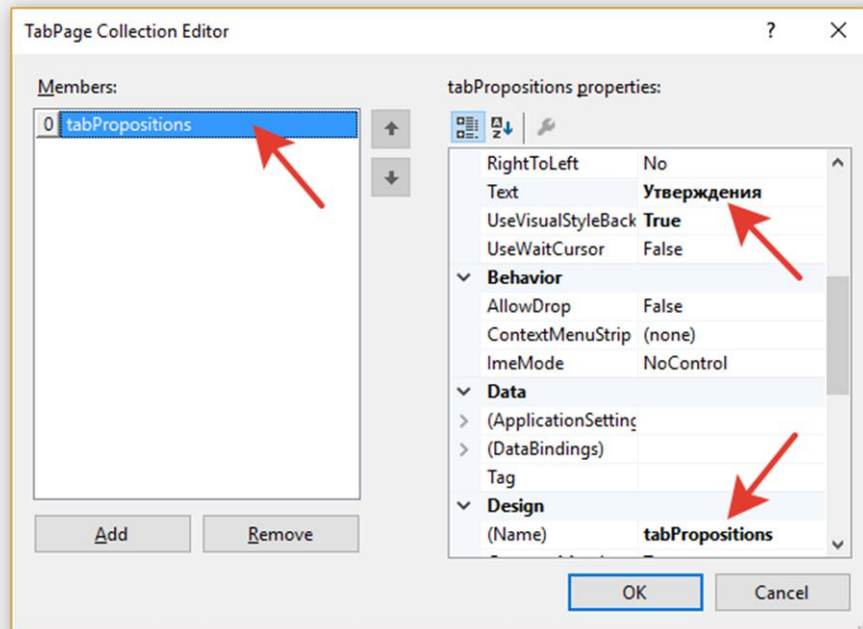


В свойствах элемента **TabControl** установите параметр **Dock** в значение **Fill**. Теперь панель вкладок будет автоматически заполнять все доступное пространство формы. Если параметр **Dock** не удастся найти, то проверьте, возможно, вы просматриваете свойства одной из вкладок, а не самого элемента управления.

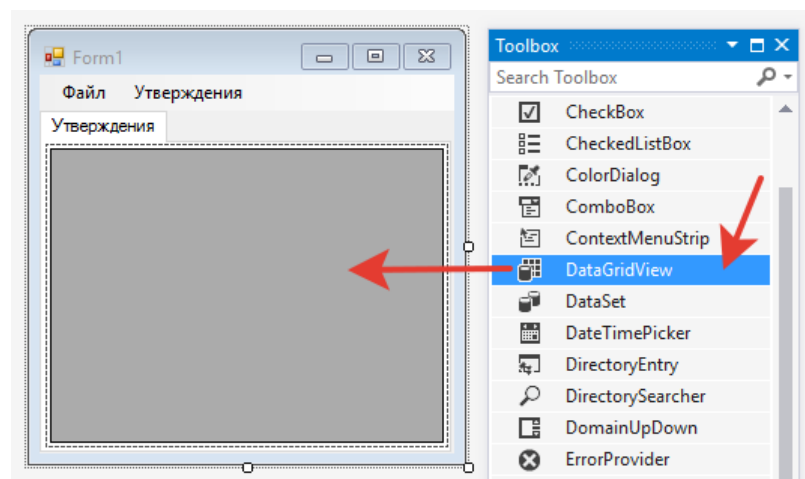
Найдите параметр **TabPages**. Щелкните мышкой справа от названия параметра. Еще раз нажмите на появившуюся кнопку с тремя точками:



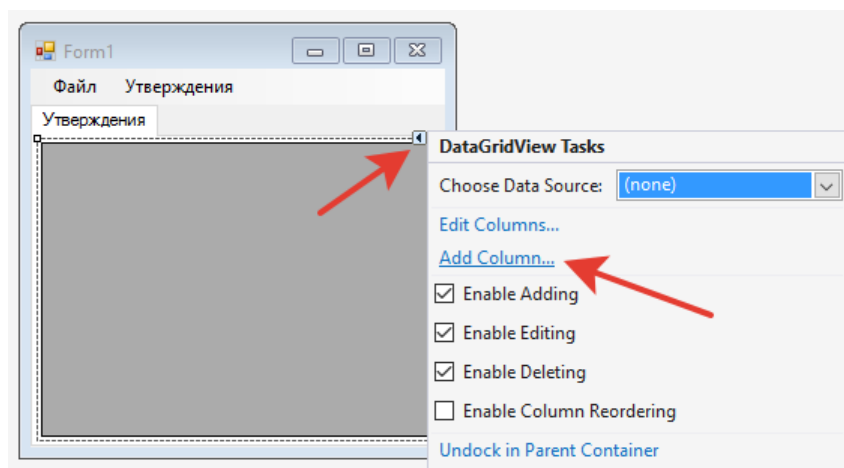
Откроется окно редактора вкладок. Удалите одну вкладку. Измените имя оставшейся вкладки на **tapPropositions**, а отображаемый текст на «Утверждения».



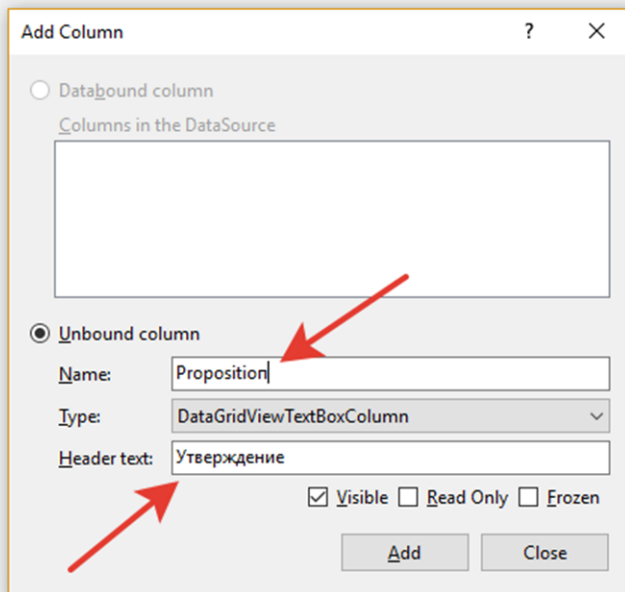
Добавим на вкладку «**Утверждения**» компонент для отображения таблиц **DataGridView**. Перетащите его из панели инструментов на середину открытой вкладки. Установите свойство **Dock** в значение **Fill**. Измените имя объекта таблицы с **dataGridView1** на **propositionsGrid**.



Добавим в таблицу **propositionsGrid** два новых столбца. Нажмите на маленькую стрелку в правой-верхней части таблицы и выберите пункт меню «**Добавить столбец**» (**Add Column**):

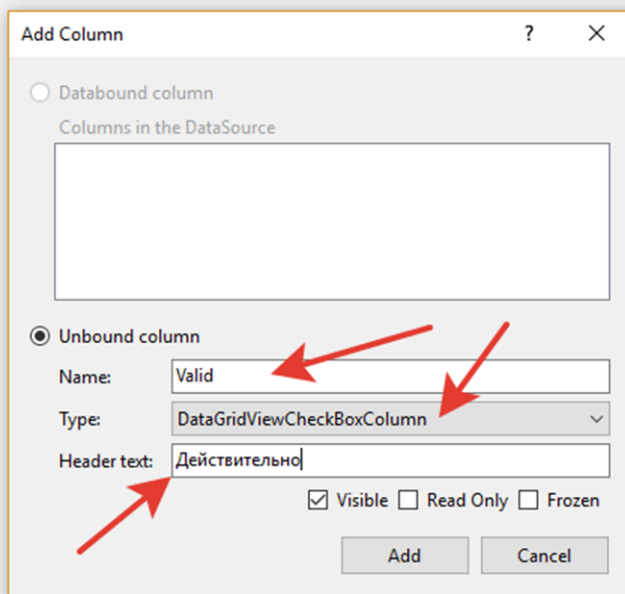


В открывшемся диалоговом окне укажите имя столбца (Name) как «**Proposition**», а отображаемый текст (Header text) как «**Утверждение**»:



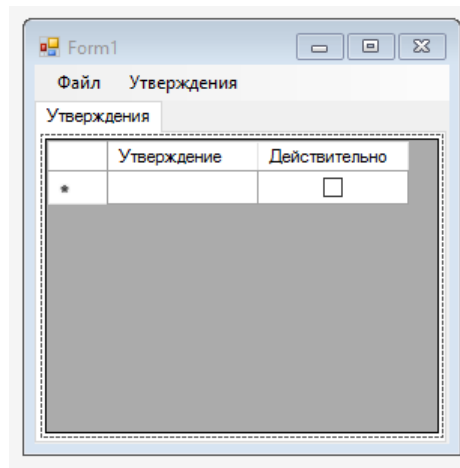
The screenshot shows the 'Add Column' dialog box with the 'Unbound column' option selected. The 'Name' field contains 'Proposition', the 'Type' is set to 'DataGridViewTextBoxColumn', and the 'Header text' is 'Утверждение'. Red arrows point to the 'Name' and 'Header text' fields. The 'Visible' checkbox is checked, and the 'Add' button is highlighted.

Нажмите кнопку «**Добавить**» (**Add**). Аналогично добавьте столбец «**Действительно**» (**Valid**), однако в этот раз дополнительно измените его тип на селектор (**DataGridViewCheckBoxColumn**):

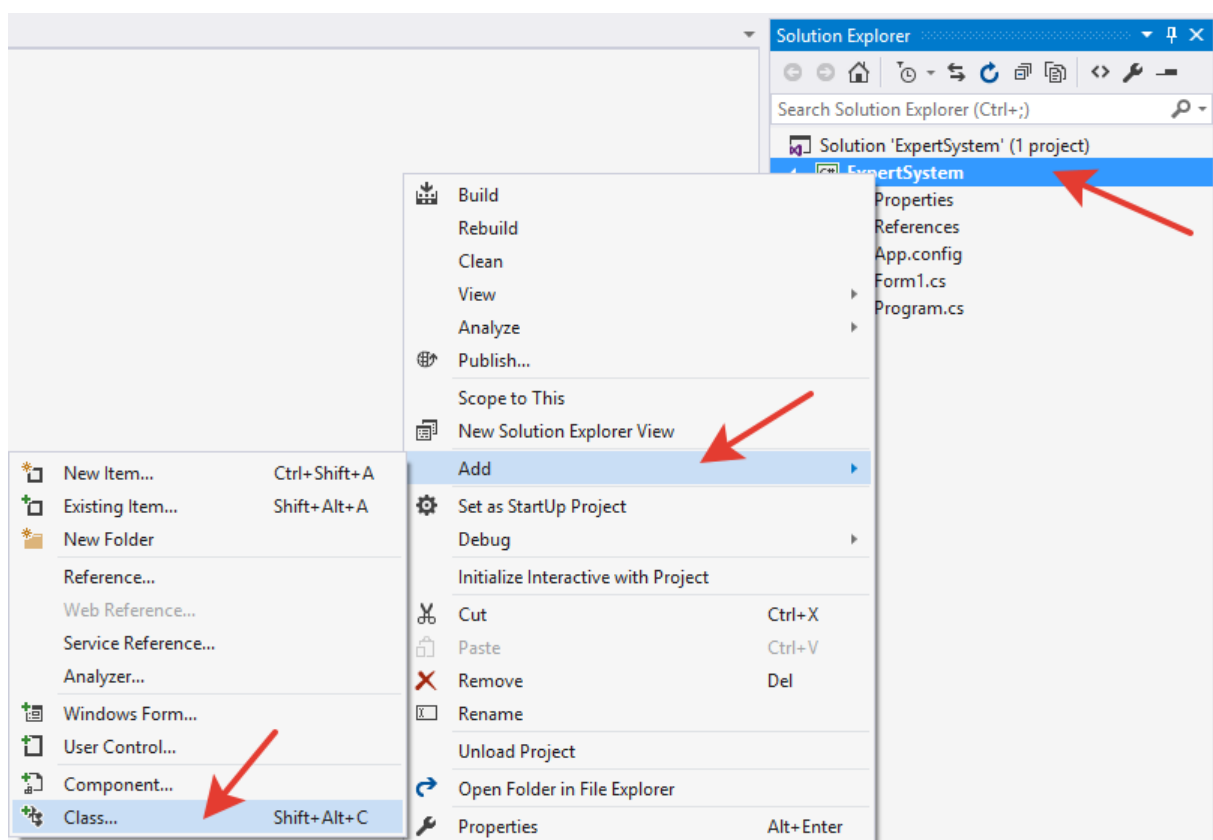


The screenshot shows the 'Add Column' dialog box with the 'Unbound column' option selected. The 'Name' field contains 'Valid', the 'Type' is set to 'DataGridViewCheckBoxColumn', and the 'Header text' is 'Действительно'. Red arrows point to the 'Name', 'Type', and 'Header text' fields. The 'Visible' checkbox is checked, and the 'Add' button is highlighted.

В результате должно получиться:



Добавим в проект несколько классов. Для этого щелкните правой кнопкой мыши на узле проекта на панели обозревателя решений. Выберите пункты всплывающего меню «Добавить» – «Класс...». Добавьте в проект классы **ExpertSystemData** и **Proposition**.



Класс **Proposition** инкапсулирует логическое утверждение. Например: «Это ползает», «Имеет хвост», «Имеет перья». Текст утверждения будет храниться в поле **Caption**, истинность утверждения в поле **Valid**.

```
[Serializable]
public class Proposition {
    public string Caption;
    public bool Valid;
}
```

Обратите внимание на атрибут **Serializable** перед объявлением класса **Proposition**. В языке C# класс с таким атрибутом может быть **сериализован** средствами языка. **Сериализация** представляет процесс преобразования какого-либо объекта в поток байтов. После преобразования мы можем этот поток байтов или записать на диск или сохранить его временно в памяти. При необходимости можно выполнить обратный процесс - **десериализацию**, то есть получить из потока байтов ранее сохраненный объект.

Отдельные утверждения будут храниться в объекте класса **ExpertSystemData**, в списке Propositions:

```
[Serializable]
public class ExpertSystemData {
    public List<Proposition> Propositions = new List<Proposition>();
}
```

Класс **ExpertSystemData** также является **сериализуемым**.

Добавим обработчики событий выбора пунктов меню. В конструкторе форм откройте наше меню и дважды щелкните на пункте меню «**Открыть**». В класс **ExpertSystemForm** будет добавлен метод **открытьToolStripMenuItem\_Click**. Переименуем его во что-то более стандартное. В окне кода нажмите правой кнопкой мыши на названии метода и выберите пункт всплывающего меню «**Переименовать**» (**Rename**). Измените имя на **openMenu\_Click**.

Аналогично, добавьте обработчики событий для пунктов «**Сохранить**», «**Сохранить как**» и «**Добавить**». Переименуйте добавленные методы соответственно в **saveMenu\_Click**, **saveAsMenu\_Click** и **addMenu\_Click**.

Откройте файл с кодом формы. Для этого можно в конструкторе форм нажать кнопку **F7** или выбрать узел формы на панели обозревателя решений и также нажать **F7**.

Приступим к добавлению функционала к нашей форме. Для начала подключим два пространства имен:

```
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
```

Пространство имен **System.IO** вам должно быть уже знакомо – в нем размещены классы для работы с файлами. Второе пространство имен содержит нужный нам класс **BinaryFormatter**, который мы будем использовать для **сериализации** данных экспертной системы.

Добавим в класс **ExpertSystemForm** два поля:

```
private ExpertSystemData data = new ExpertSystemData();
private string filename;
```

В поле **data** хранятся введенные утверждения, а в поле **filename** – имя текущего файла проекта.

Для вывода утверждений на форму напомним метод **UpdatePropositions**.

```
public void UpdatePropositions() {
    propositionsGrid.Rows.Clear();
    var count = data.Propositions.Count;
    if (count > 0) {
        propositionsGrid.RowCount = count;
        for (int i = 0; i < count; ++i) {
            var item = data.Propositions[i];
            propositionsGrid.Rows[i].Cells[0].Value = item.Caption;
            propositionsGrid.Rows[i].Cells[1].Value = item.Valid;
        }
    }
}
```

Внутри этого метода мы сначала очищаем таблицу утверждений:

```
propositionsGrid.Rows.Clear();
```

Затем устанавливаем количество строк в таблице:

```
var count = data.Propositions.Count;
if (count > 0) {
    propositionsGrid.RowCount = count;
```

И, наконец, в цикле добавляем утверждения:

```
for (int i = 0; i < count; ++i) {
```

```

        var item = data.Propositions[i];
        propositionsGrid.Rows[i].Cells[0].Value = item.Caption;
        propositionsGrid.Rows[i].Cells[1].Value = item.Valid;
    }

```

Изменим обработчик добавления утверждения **addMenu\_Click**. В этой лабораторной работе напомним простейший вариант с фиксированными данными. Итак, создадим новое утверждение:

```
var prop = new Proposition();
```

Установим для текста утверждения значение «Проверка» и присоединим произвольное случайное число от 0 до 99:

```
prop.Caption = "Проверка " + new Random().Next(100);
```

Истинность утверждения установим в **true**:

```
prop.Valid = true;
```

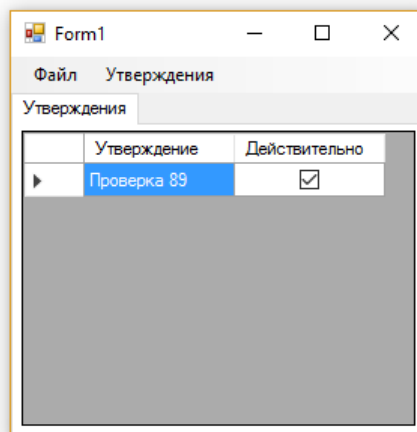
Добавим утверждение в список:

```
data.Propositions.Add(prop);
```

Обновим таблицу:

```
UpdatePropositions();
```

В результате, при выборе пункта меню «**Утверждения**» - «**Добавить**» мы получим следующий результат:



Изменим обработчики сохранения данных **saveMenu\_Click** и **saveAsMenu\_Click**. Код первого обработчика:

```

if (filename == null) {
    saveAsMenu_Click(sender, e);
    return;
}
var file = new FileStream(filename, FileMode.OpenOrCreate);
var formatter = new BinaryFormatter();
formatter.Serialize(file, data);
file.Close();

```

Если при нажатии кнопки **сохранить** оказывается, что файл для записи еще не был определен, то вызывается диалог «**Сохранить как**».

```

if (filename == null) {
    saveAsMenu_Click(sender, e);
    return;
}

```

В противном случае открывается файл на запись:



```
var file = new FileStream(filename, FileMode.OpenOrCreate);
```

Создается конвертер для сериализации в двоичный формат:

```
var formatter = new BinaryFormatter();
```

Производится сериализация с записью в файл:

```
formatter.Serialize(file, data);
```

После чего файл закрывается.

Метод **saveAsMenu\_Click** занимается созданием диалога для выбора места сохранения файла:

```
var dialog = new SaveFileDialog();
dialog.Title = "Сохранение проекта...";
dialog.Filter = "Файлы экспертных систем|.es|Все файлы|*.*";
if (dialog.ShowDialog() == DialogResult.OK) {
    filename = dialog.FileName;
    saveMenu_Click(sender, e);
}
```

Мы создаем новый объект стандартного диалога **SaveFileDialog**, инициализируем его, и, если пользователь нажал кнопку «**OK**», сохраняем путь к файлу в поле **filename**. Затем вызываем метод сохранения **saveMenu\_Click**, описанный выше.

Осталось определить метод **openMenu\_Click** для загрузки сохраненных данных. Для начала создадим объект диалога выбора файла:

```
var dialog = new OpenFileDialog();
dialog.Title = "Открыть проект...";
dialog.Filter = "Файлы экспертных систем|.es|Все файлы|*.*";
```

Если при открытии файла пользователь нажал кнопку «**OK**», то загружаем содержимое файла в объект **data**:

```
if (dialog.ShowDialog() == DialogResult.OK) {
    var file = new FileStream(dialog.FileName, FileMode.Open);
    var formatter = new BinaryFormatter();
    data = (ExpertSystemData)formatter.Deserialize(file);
    file.Close();
    UpdatePropositions();
}
```

Здесь опять используется класс **BinaryFormatter**, но теперь с его помощью выполняется операция загрузки, или десериализация:

```
var formatter = new BinaryFormatter();
data = (ExpertSystemData)formatter.Deserialize(file);
```

После загрузки данных нам необходимо вывести их на форму. Для этого достаточно вызвать метод **UpdatePropositions**.

Form1

Файл Утверждения

Утверждения

	Утверждение	Действительно
▶	Проверка 70	<input checked="" type="checkbox"/>
	Проверка 49	<input checked="" type="checkbox"/>
	Проверка 91	<input checked="" type="checkbox"/>
	Проверка 64	<input checked="" type="checkbox"/>
	Проверка 97	<input checked="" type="checkbox"/>
	Проверка 22	<input checked="" type="checkbox"/>

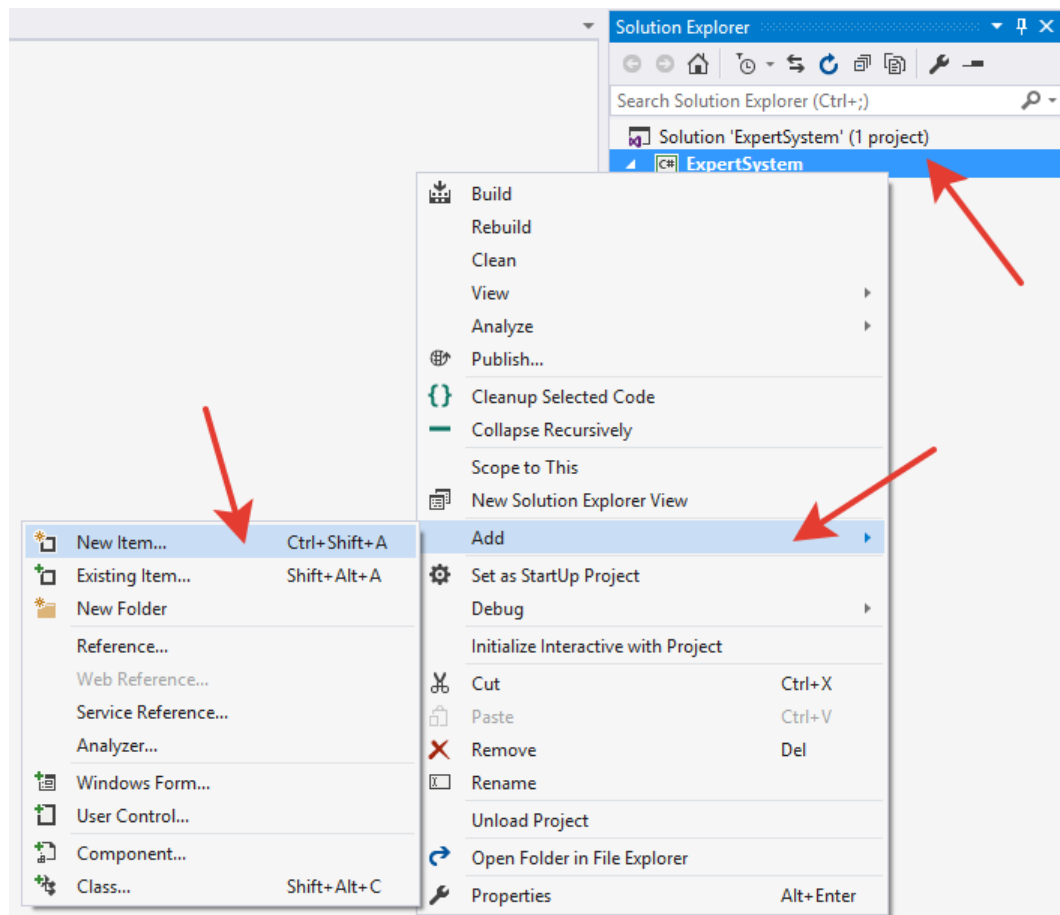
# ЛАБОРАТОРНАЯ РАБОТА №6 (ЗАДАНИЕ 2)

Технологии искусственного интеллекта. Разработка экспертной системы.

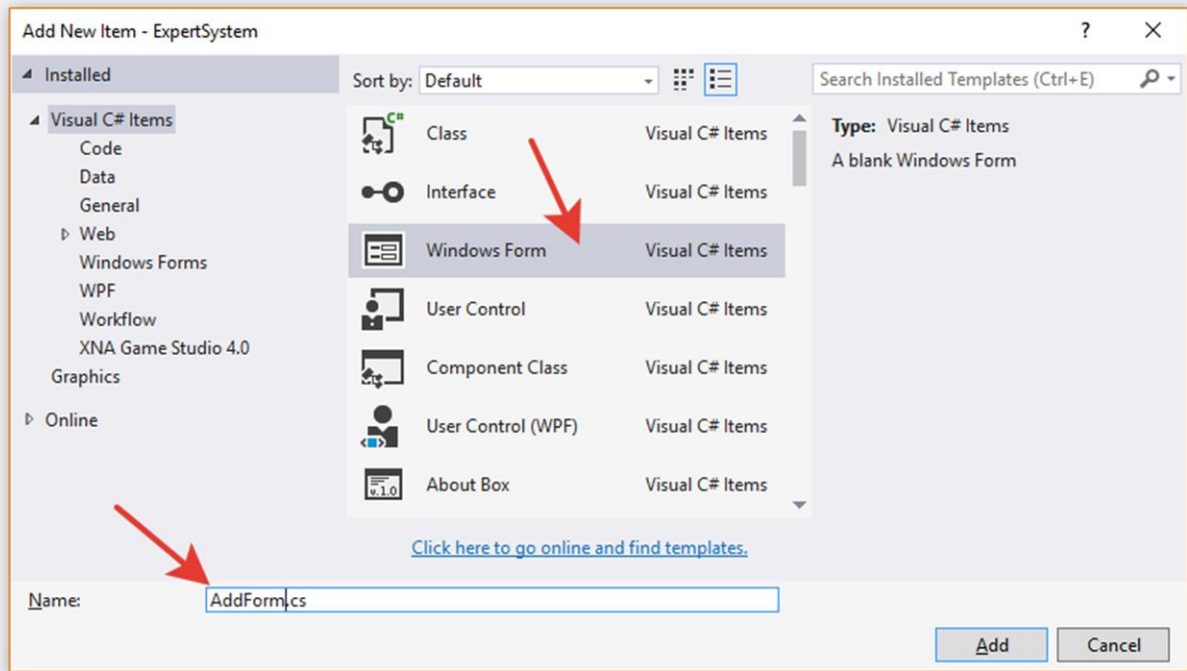
## ДОБАВЛЕНИЕ НОВОГО УТВЕРЖДЕНИЯ

Добавим в наш проект экспертной системы еще одну форму. Эта форма будет использоваться для ввода текста утверждения и добавления его в общий список.

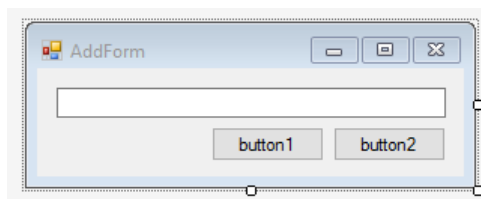
В окне обозревателя решений нажмите правой кнопкой мыши на узле проекта и выберите пункты выпадающего меню «Добавить» - «Новый элемент» (Add – New Item).



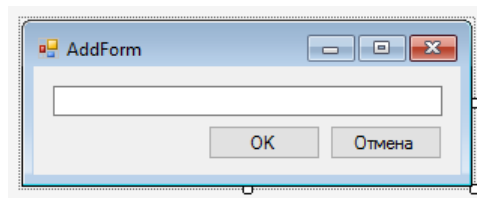
В открывшемся диалоговом окне выберите пункт списка «Форма windows» (Windows Form) и укажите название формы как **AddForm**:



Добавьте на форму поле для ввода текста (**TextBox**) и две кнопки (**Button**). Расположите их как показано на рисунке:



Зайдите в свойства кнопок и измените их текст на «**OK**» и «**Отмена**». Измените названия переменных кнопок соответственно на **buttonOK** и **buttonCancel**.



Для элемента **TextBox** измените имя переменной на **InputBox**. Модификатор видимости переключите с **Private** на **Public**:



Дважды щелкните на кнопках «**OK**» и «**Отмена**» для добавления обработчиков событий нажатия. Измените код обработчиков на следующий:

```
private void buttonOK_Click(object sender, EventArgs e) {
    DialogResult = DialogResult.OK;
    Close();
}

private void buttonCancel_Click(object sender, EventArgs e) {
    DialogResult = DialogResult.Cancel;
}
```

```

        Close();
    }

```

Каждый диалог содержит в коде класса поле **DialogResult**, содержащее тип нажатой при закрытии кнопки. Оба метода-обработчика устанавливают свое значение для этого поля и закрывают форму вызовом метода **Close()**.

Найдите класс **ExpertSystemData** и добавьте в него новое поле **rules**:

```
public List<int> rules = new List<int>();
```

Список **rules** будет хранить весовые коэффициенты, по которым будет рассчитываться один из вариантов ответов (исходов), наиболее подходящий под набор утверждений. Поскольку класс **ExpertSystemData** объявлен с атрибутом **Serializable**, то содержимое списка **rules** будет автоматически добавлено в файл при сохранении.

Вернемся к главной форме приложения. Нам необходимо добавить код по созданию нашего нового диалога в обработчик нажатия пункта меню «**Утверждения**» - «**Добавить**». В коде формы он называется **addMenu\_Click**, удалите его содержимое и вставьте следующий код:

```

private void addMenu_Click(object sender, EventArgs e) {
    var dialog = new AddForm();
    dialog.ShowDialog(this);
    if (dialog.DialogResult == DialogResult.OK) {
        var prop = new Proposition();
        prop.Caption = dialog.InputBox.Text;
        prop.Valid = false;
        data.Propositions.Add(prop);
        data.rules.Add(0);
        UpdatePropositions();
    }
}

```

Процесс вызова пользовательского диалога аналогичен процессу работы со стандартными диалогами по работе с файлами. Сначала мы создаем объект формы:

```
var dialog = new AddForm();
```

Затем отображаем диалог:

```
dialog.ShowDialog(this);
```

Если пользователь закрыл наш диалог нажав кнопку «**OK**», то мы можем добавить в систему новое утверждение:

```

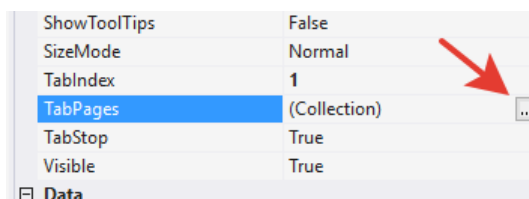
if (dialog.DialogResult == DialogResult.OK) {
    var prop = new Proposition();
}

```

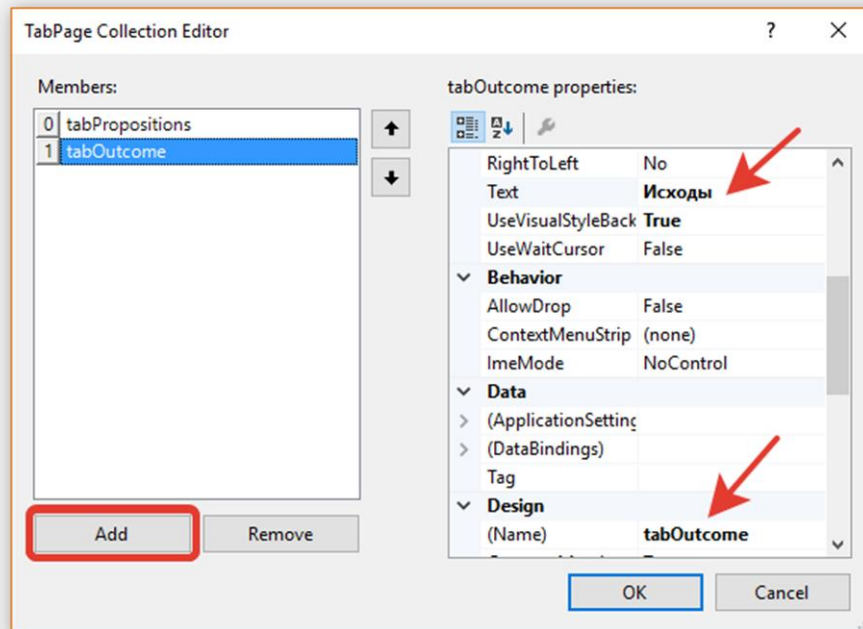
Введенный текст утверждения извлекается из диалога через открытое поле **InputBox**:

```
prop.Caption = dialog.InputBox.Text;
```

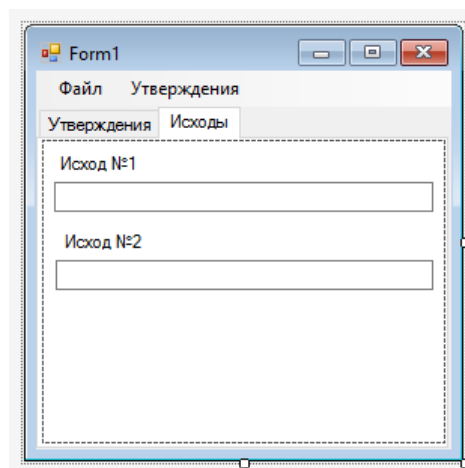
Добавим на основную форму еще одну закладку. Зайдите в свойства элемента **TabControl** и найдите параметр **TabPages**. Нажмите на кнопку с троеточием:



В открывшемся диалоговом окне добавьте новую закладку. Назовите эту закладку «**Исходы**» и измените имя ее переменной на **tabOutcome**:

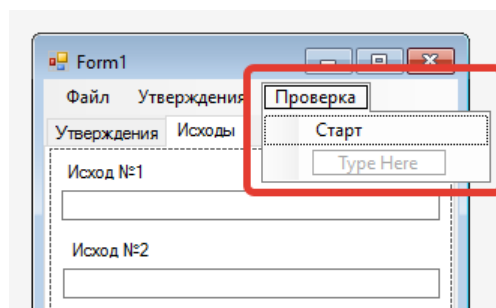


Откройте закладку в конструкторе форм и добавьте на нее две метки и два поля ввода текста (**TextBox**):



В свойствах полей ввода измените имена переменных на **Outcome1** и **Outcome2**.

В меню формы добавьте два новых пункта:



Дважды щелкните на пункте «**Старт**» для добавления обработчика события.

```
private void стартToolStripMenuItem_Click(object sender, EventArgs e) {
```

Выберите название метода и нажмите кнопку **F2**. Переименуйте этот обработчик в **startMenu\_Click**:

```
private void startMenu_Click(object sender, EventArgs e) {
```

Теперь напишем наш алгоритм по выбору одного из двух возможных вариантов исходов на основе истинности набора утверждений. Для начала опросим пользователя:

```
DialogResult result;
foreach (var prop in data.Propositions) {
    result = MessageBox.Show(prop.Caption, "Это утверждение верно?",
                             MessageBoxButtons.YesNo);
    prop.Valid = (result == DialogResult.Yes);
}
UpdatePropositions();
```

В цикле **foreach** мы проходим по всем утверждениям и для каждого спрашиваем пользователя, истинно оно или нет. Пусть у нас есть следующий набор утверждений:

Утверждение	Действительно
оно летает	<input type="checkbox"/>
оно бегает	<input type="checkbox"/>
оно ползает	<input type="checkbox"/>

Тогда запрос будет выглядеть следующим образом:

Теперь, на основе введенных значений и накопленных данных мы вычисляем результат:

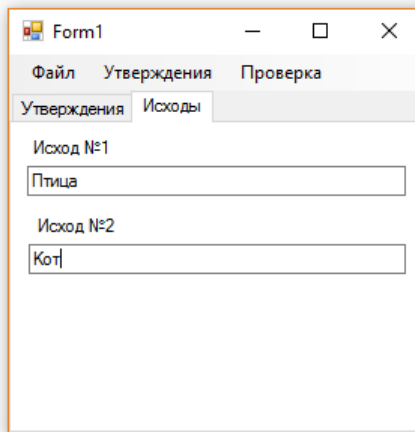
```
var decision = 0;
for (int i = 0; i < data.Propositions.Count; ++i) {
    var valid = Convert.ToInt32(data.Propositions[i].Valid);
    decision += valid * data.rules[i];
}
```

Поскольку при первом запуске системы у нас нет накопленных данных, то переменная **decision** будет равна 0.

Если переменная **decision** больше 0, то мы считаем, что выбран первый исход, иначе второй:

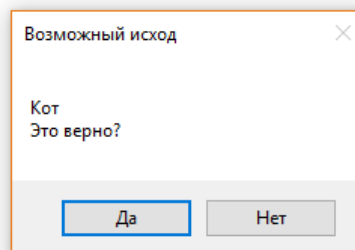
```
var text = decision > 0 ? Outcome1.Text : Outcome2.Text;
result = MessageBox.Show(text + "\nЭто верно?", "Возможный исход",
                          MessageBoxButtons.YesNo);
```

Для исходов «Птица» и «Кот» первый запуск будет всегда выдавать «Кот»:



Спросим у пользователя, правильно ли был найден ответ:

```
result = MessageBox.Show(text + "\nЭто верно?", "Возможный исход",
                          MessageBoxButtons.YesNo);
```



Если пользователь считает, что ответ не верный, то модифицируем весовые коэффициенты:

```
if (result == DialogResult.No) {
    if (decision > 0) {
        for (int i = 0; i < data.rules.Count; ++i) {
            var valid = Convert.ToInt32(data.Propositions[i].Valid);
            data.rules[i] -= valid;
        }
    } else {
        for (int i = 0; i < data.rules.Count; ++i) {
            var valid = Convert.ToInt32(data.Propositions[i].Valid);
            data.rules[i] += valid;
        }
    }
}
```

При следующем запуске проверки, при тех же настройках утверждений, ответ изменится на правильный. По сути, мы реализовали частный случай модели нейрона, согласно которой на входе мы имеем разные сигналы, а на выходе сумму этих сигналов, умноженную на весовые коэффициенты.

## ДРУГОЙ ПРИМЕР

Допустим, изначальная таблица правил заполнена нулями. У вас есть утверждения:

- Это летает
- Это ползает
- Это бежит.

Зададим таблицу утверждений для воробья:



Это летает	Да
Это ползает	Нет
Это бегают	Да

Что у нас вычислит программа?

Результат расчета будет равен нулю. Это значит, что экспертная система предположила, что это не птица. Естественно, предположение неверное, мы знаем, что воробей - это птица и говорим об этом программе. Она меняет весовые коэффициенты. Теперь они у нас будут:

Это летает	1
Это ползает	0
Это бегают	1

В следующий раз на тот же самый набор утверждений программа ответит "Это птица".

Теперь зададим параметры, допустим, кота:

Это летает	Нет
Это ползает	Нет
Это бегают	Да

Программа предположила, что это птица. Мы говорим ей, что это неверно, так как кот не может быть птицей. Получаем следующий набор правил:

Это летает	1
Это ползает	0
Это бегают	0

Теперь программа будет отвечать правильно для случая с котом и для случая с воробьем.

---

## ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

- Модифицируйте программу таким образом, чтобы она дополнительно сохраняла и загружала в файл значения текстовых полей «Исход №1» и «Исход №2»