# Практическое занятие № 22. Работа с динамическими структурами

**1 Цель занятия**

Получить практические навыки работы с динамическими структурами

**2 Перечень оборудования и программного обеспечения**

Персональный компьютер

Microsoft Office (Word)

Microsoft Visual Studio

**3 Краткие теоретические сведения**

***Динамические структуры данных*** – это структуры данных, память под которые выделяется и освобождается по мере необходимости.

Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами. При этом не учитывается изменение содержимого самих элементов данных. Такая особенность динамических структур, как непостоянство их размера и характера отношений между элементами, приводит к тому, что на этапе создания машинного кода программа-компилятор не может выделить для всей структуры в целом участок памяти фиксированного размера, а также не может сопоставить с отдельными компонентами структуры конкретные адреса. Для решения проблемы адресации динамических структур данных используется метод, называемый динамическим распределением памяти, то есть память под отдельные элементы выделяется в момент, когда они "начинают существовать" в процессе выполнения программы, а не во время компиляции. Компилятор в этом случае выделяет фиксированный объем памяти для хранения адреса динамически размещаемого элемента, а не самого элемента.

Динамическая структура данных характеризуется тем что:

* она не имеет имени;
* ей выделяется память в процессе выполнения программы;
* количество элементов структуры может не фиксироваться;
* размерность структуры может меняться в процессе выполнения программы;
* в процессе выполнения программы может меняться характер взаимосвязи между элементами структуры.

Необходимость в динамических структурах данных обычно возникает в следующих случаях:

* используются переменные, имеющие довольно большой размер (например, массивы большой размерности), необходимые в одних частях программы и совершенно не нужные в других;
* в процессе работы программы нужен массив, список или иная структура, размер которой изменяется в широких пределах и трудно предсказуем;
* размер данных, обрабатываемых в программе, превышает объем сегмента данных.

Примером динамических структур являются ***коллекции***. Преимуществом коллекций является то, что их размер динамичен.

Еще одни плюсом коллекций является то, что они уже имеют внутреннюю реализацию различных структур данных, начиная со списка и заканчивая хеш-таблицей. Таким образом, в большинстве случаев не требуется реализовывать методы добавления/поиска/удаления элемента из структуры данных, т.к. эти методы уже реализованы внутри коллекции.

В C# коллекции содержатся в пространствах имен System.Collections и Systems.Collections.Generic. Коллекции подразделяются на универсальные и неуниверсальные. ***Универсальные коллекции*** могут быть типизированы и содержать объекты определенного типа. Это позволяет выигрывать в производительности, особенно, если коллекции хранят типы значений (int, bool, double и т.п.). В ***неуниверсальных коллекциях*** происходит упаковка хранимых объектов в тип Object, что требует их последующей распаковки при работе с ними.

Структура данных «список» в C# реализована как универсальная коллекция List, кроме того, есть аналог ArrayList - неуниверсальная коллекция. Для получения текущего количества элементов в коллекции можно использовать свойство Count.

Обозначим основные методы по работе с коллекциями List и ArrayList.

**Добавление элементов:**

Add(<элемент\_коллекции>) – добавляет элемент в коллекцию. В случае универсальной коллекции тип элемента должен совпадать с типом коллекции (пример был приведен выше);

AddRange(<коллекция>) – добавляет в текущую коллекцию совокупность элементов заданной коллекции. При этом, коллекция, указанная в качестве параметра, должна реализовывать интерфейс ICollection (стандартный интерфейс коллекции, замечание значимо при использовании пользовательской коллекции);

Insert(<место>, <элемент>) – вставляет в заданное место (индекс элемента) заданный элемент;

InsertRange(<место>, <коллекция>) – вставляет в заданное место в коллекции заданную коллекцию.

**Удаление элементов:**

Remove(<элемент>) – удаляет элемент из коллекции (первое вхождение элемента);

RemoveAt(<позиция>) – удаляет элемент на заданной позиции;

RemoveRange(<позиция>, <количество>) – удаляет заданное количество элементов, начиная с заданной позиции;

Clear() – удаляет все элементы из коллекции.

**Поиск элементов:**

boolContains(<элемент>) – возвращает, содержится ли заданный элемент в коллекции;

intIndexOf(<элемент>) – возвращает позицию, на которой содержится в коллекции заданный элемент (первое вхождение);

intIndexOf(<элемент>, <позиция>) – возвращает позицию, на которой содержится заданный элемент (первое вхождение). Поиск осуществляется по коллекции, начиная с заданной и заканчивая концом коллекции;

intIndexOf(<элемент>, <стартовая\_позиция>, <финишная\_позиция>) – возвращает позицию, на которой содержится заданный элемент (первое вхождение). Поиск осуществляется по коллекции, начиная с заданной стартовой позиции и заканчивая заданной финишной позицией;

intLastIndexOf(<элемент>) – возвращает позицию, на которой содержится в коллекции заданный элемент (последнее вхождение). Метод перегружен, его перегруженные версии копируют перегрузки метода IndexOf(), рассмотренного выше;

intBinarySearch(<элемент>) – возвращает позицию, на которой находится заданный элемент в коллекции. По коллекции осуществляется бинарный (двоичный) поиск. Коллекция должная быть отсортирована. Метод содержит несколько перегрузок, можно также задавать компаратор (метод сравнения) для элементов.

**Дополнительные методы для работы с коллекциями:**

Sort() – сортирует коллекцию, используя стандартный компаратор. Если тип элементов коллекции является пользовательским, то компаратор (сравнение элементов) нужно реализовать в соответствующем классе этого типа;

stringToString() – стандартный метод, переводящий коллекцию в строку;

Reverse() – переворачивает коллекцию, точнее, порядок элементов в коллекции. Первый элемент меняется с последним и т.д.;

Reverse(<стартовая\_позиция>, <количество\_элементов>) – перево­рачивает в коллекции заданное количество элементов, начиная с заданного номера элемента;

GetRange(<стартовая\_позиция>, <количество элементов>) – возвра­щает коллекцию, элементами которой является заданное количество элементов данной коллекции, начиная с заданной стартовой позиции;

GetType() – возвращает тип элементов коллекции;

CopyTo(<массив>) – копирует элементы коллекции в заданный массив;

CopyTo(<массив>, <номер\_позиции>) – копирует элементы коллекции в заданный массив, начиная с заданной позиции в массиве;

CopyTo(<массив>, <номер\_позиции>, <количество\_элементов>) – копирует заданное количество элементов коллекции в заданный массив, начиная с заданной позиции в массиве.

**Применение коллекций** List **и элемент управления** ListBox

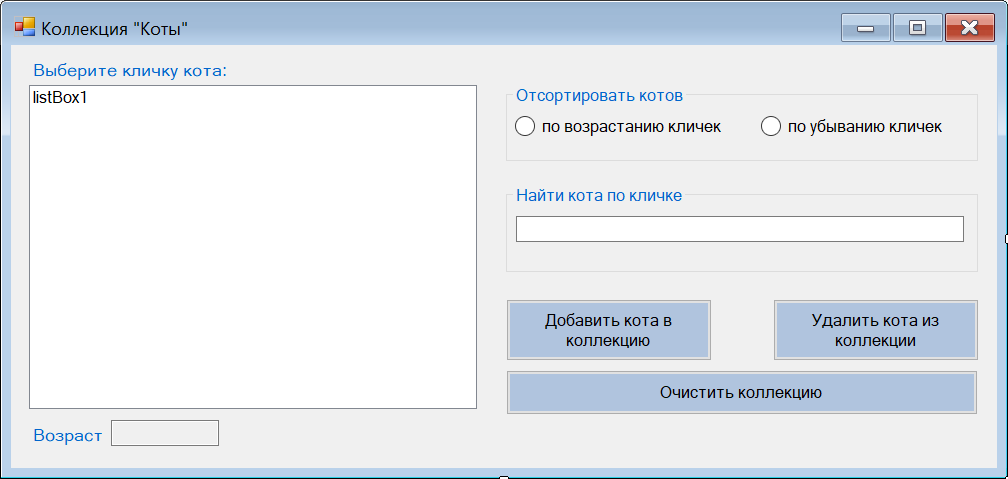
Элемент управления ListBox является примером динамического списка.

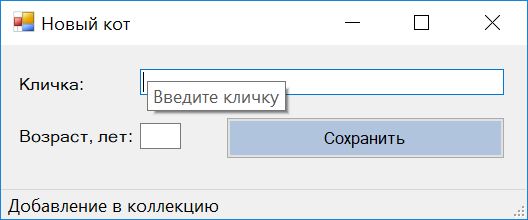
Элемент управления ListBox позволяет отображать несколько элементов, позволяя пользователям прокручивать длинный список.

**Задание.** Создать класс элементов коллекции «Коты: кличка, возраст». Определить заданную коллекцию в классе Form. Разработать методы для работы с элементами коллекции.

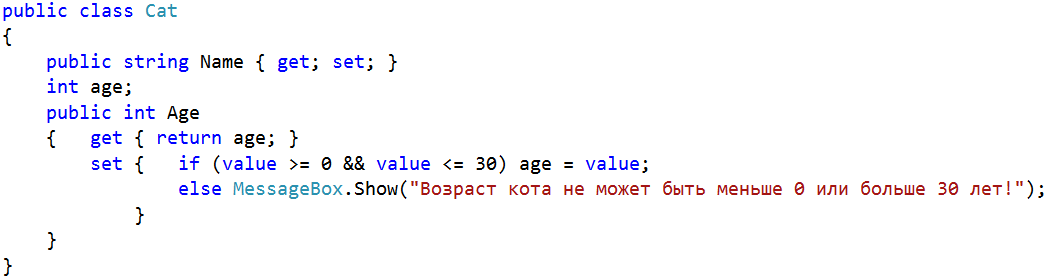
**Решение.**

Формы Windows-приложения для работы с заданной коллекцией:

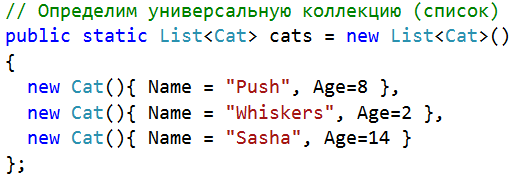




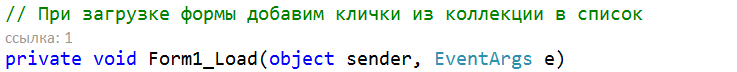
Код класса элементов коллекции (вставить после класса Form1):

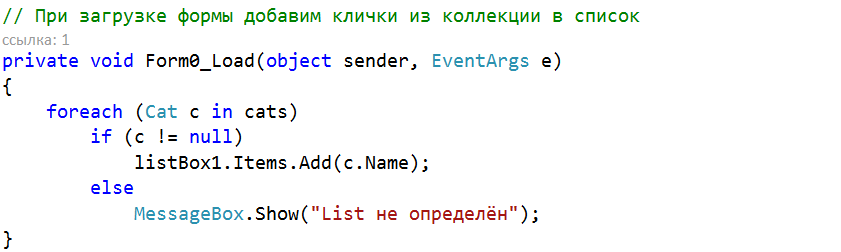


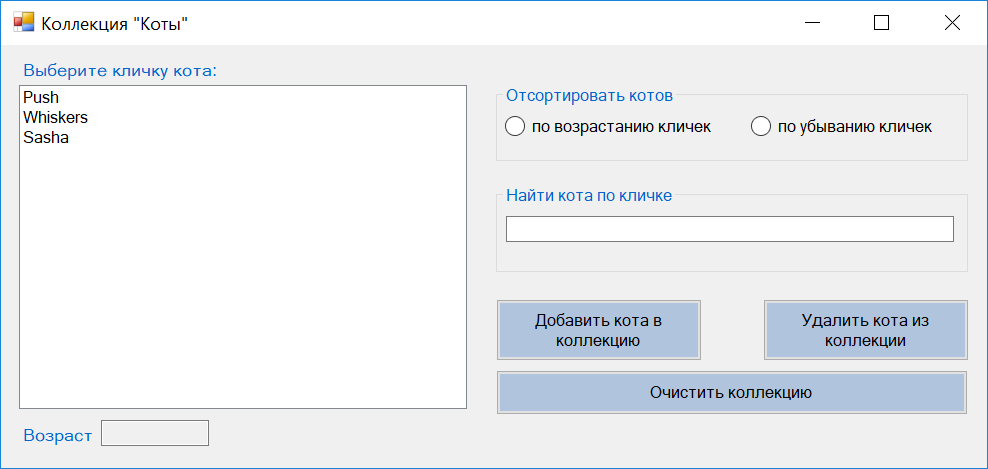
Код описания заданной коллекции в теле класса Form1:

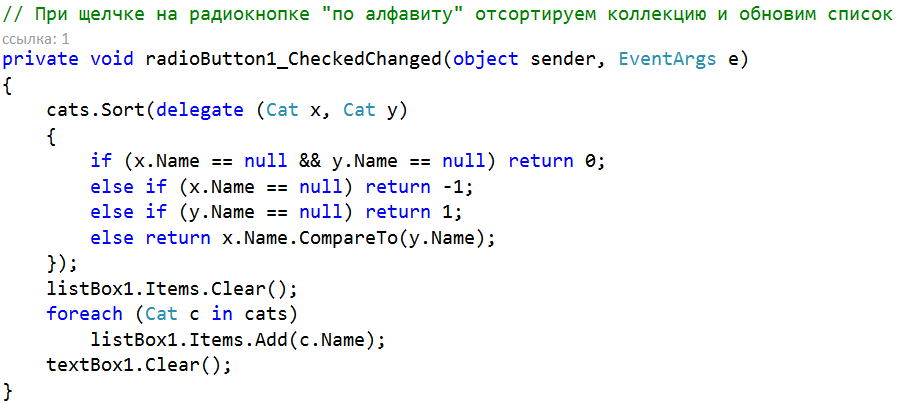


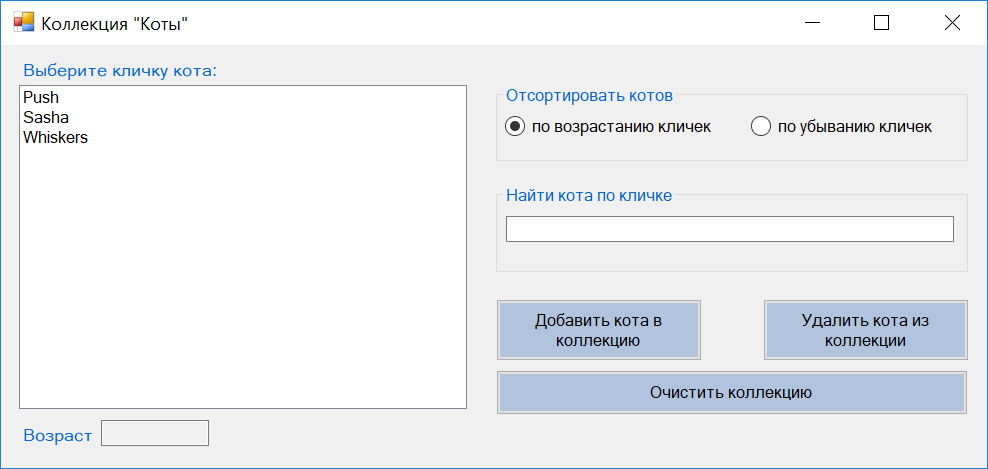
Методы класса Form1 для работы с элементами коллекции:

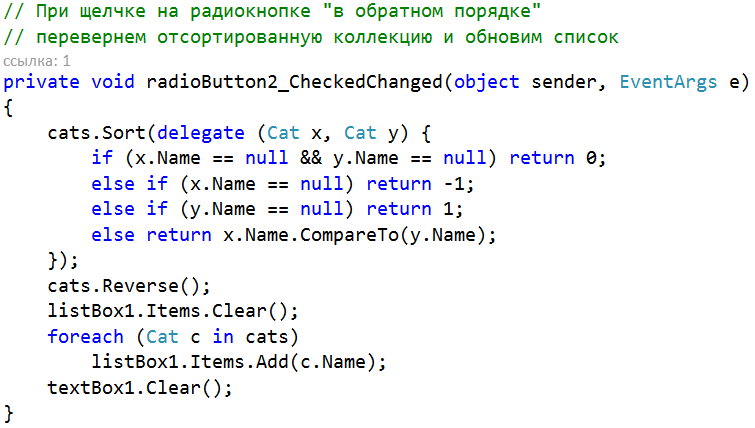


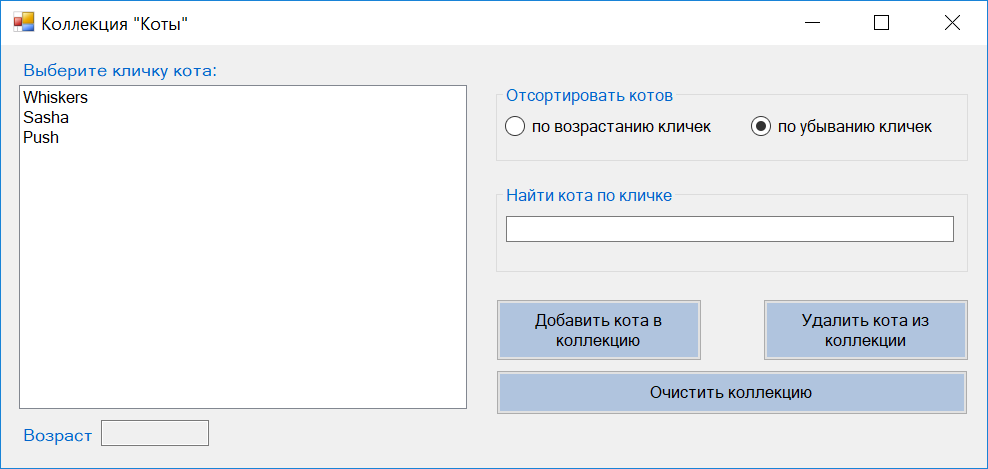


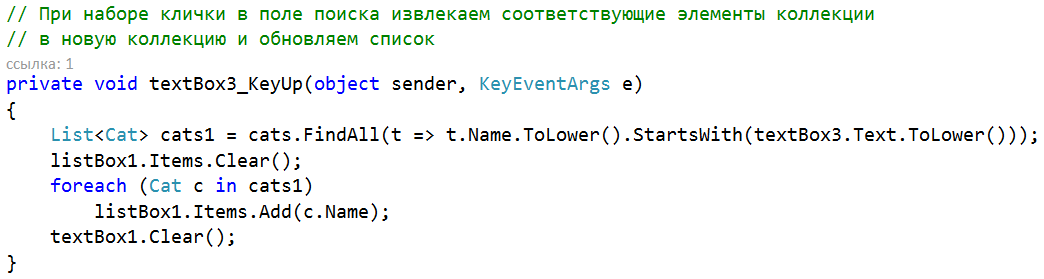


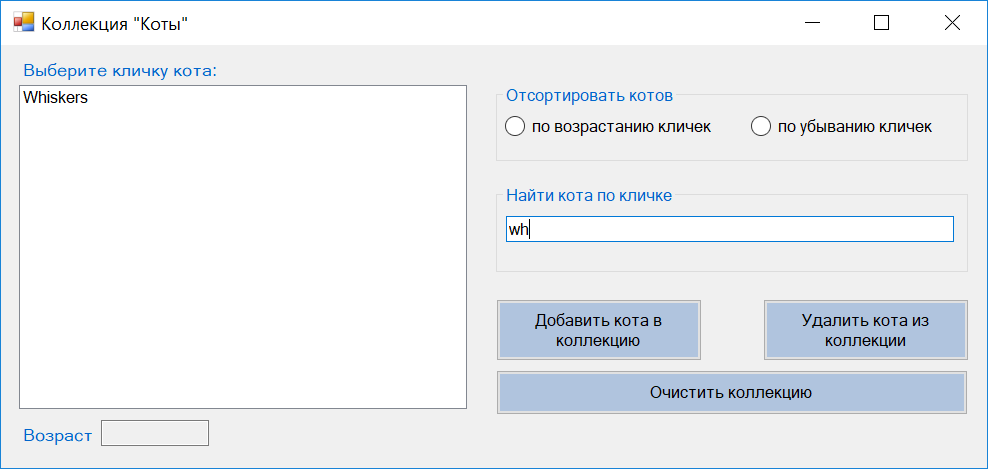


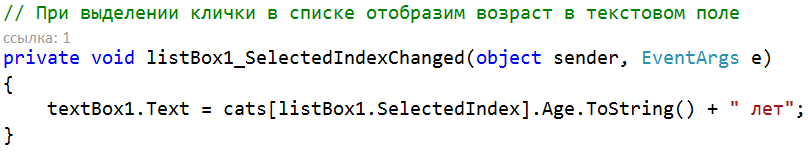


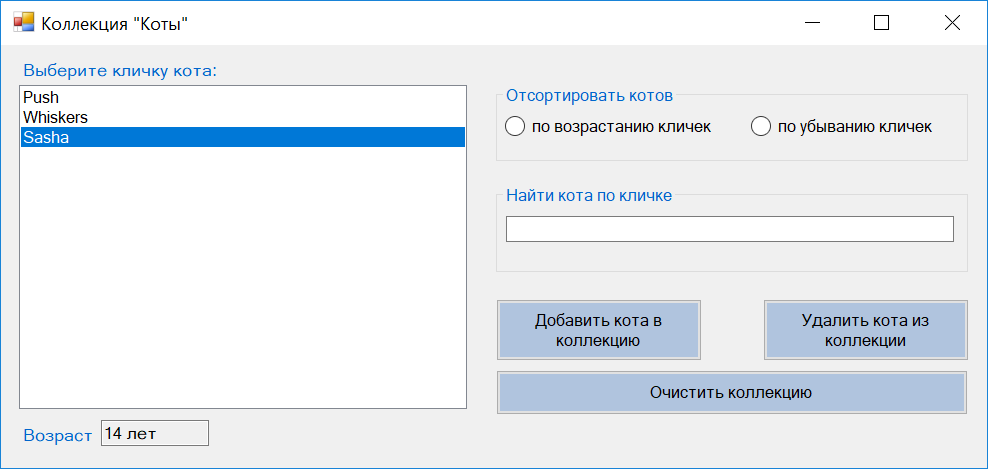


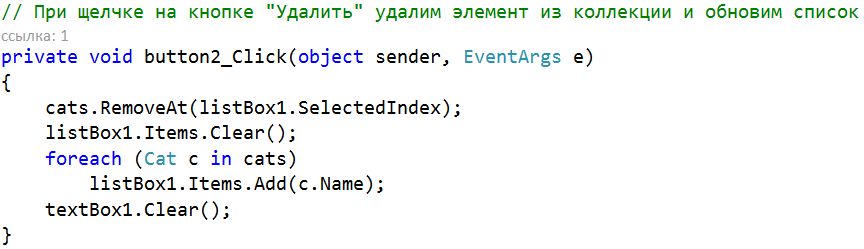


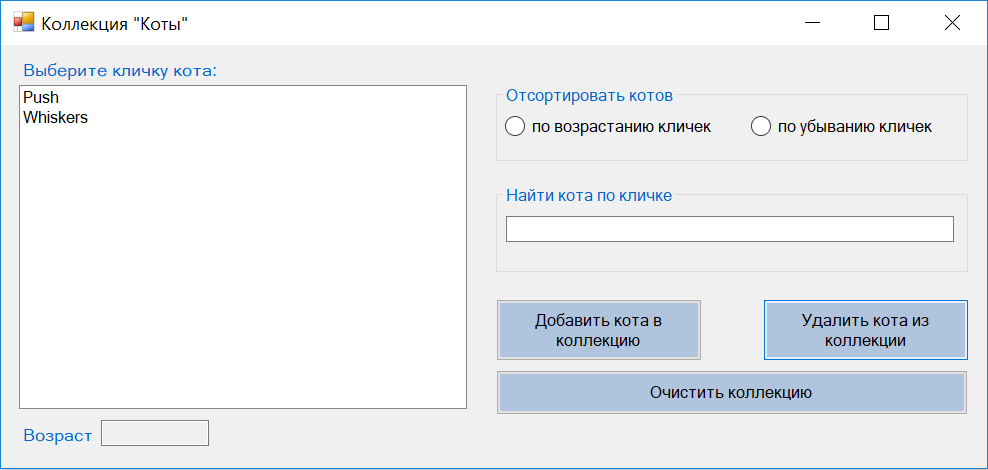


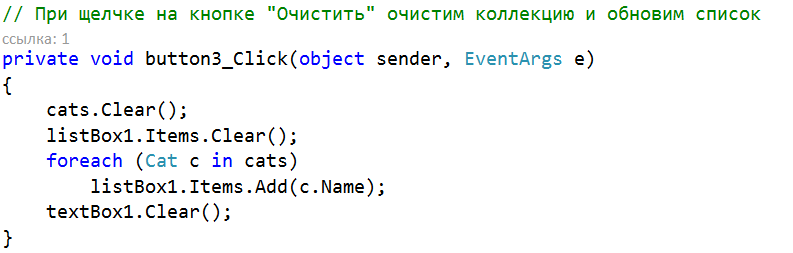


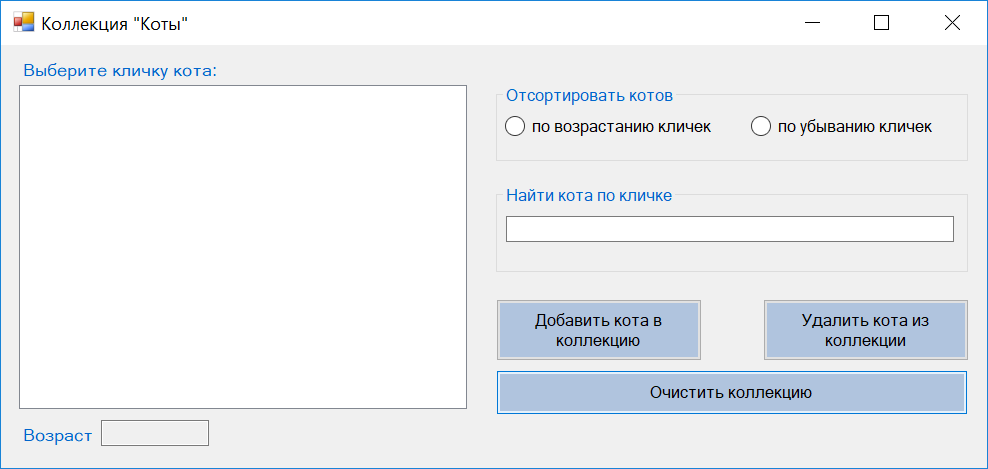


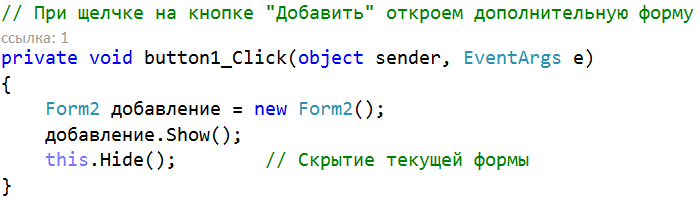


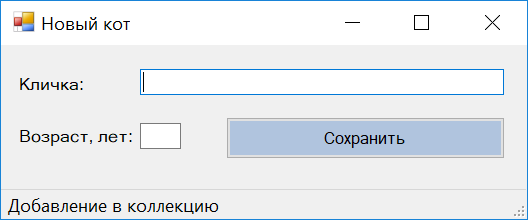


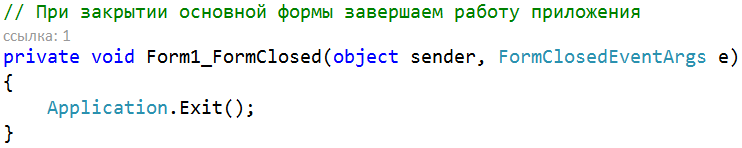




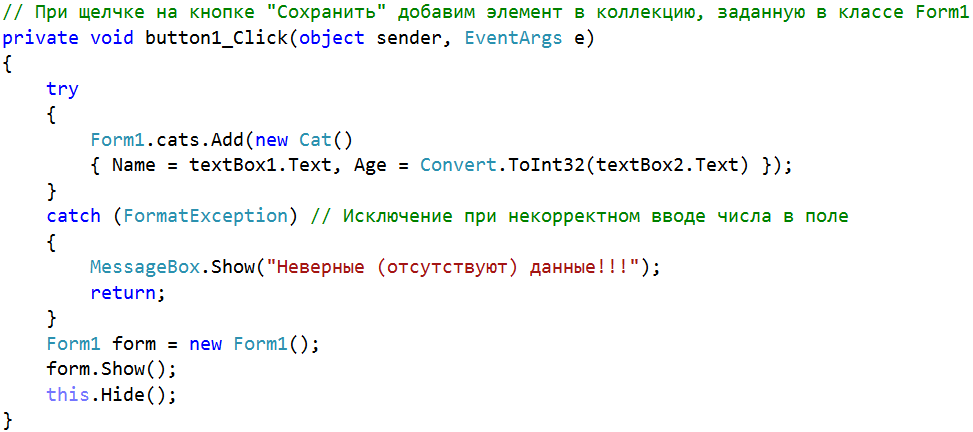


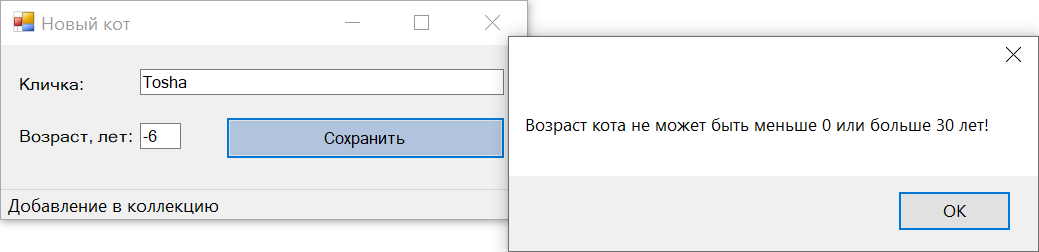


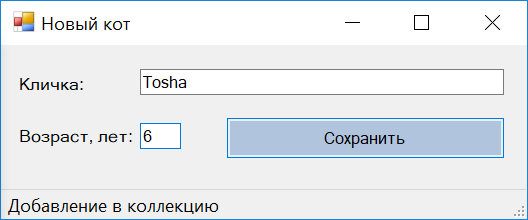


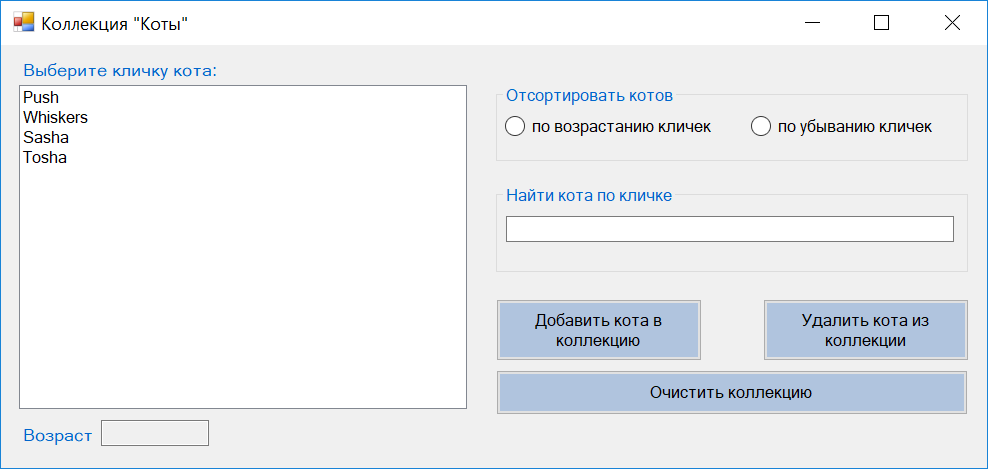


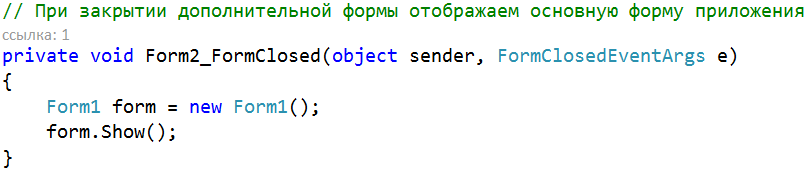
Методы класса Form2 для добавления элемента в коллекцию:











**4 Порядок выполнения работы**

4.1 Изучить теоретические сведения и задание к работе.

4.2 В соответствии с вариантом задания создать формы приложения Windows Forms.

4.3 В соответствии с вариантом задания создать класс элементов заданной коллекции.

4.4 Определить заданную коллекцию в классе Form.

4.5 В соответствии с вариантом задания разработать и отладить коды методов приложения Windows Forms для обработки (отображения, добавления, удаления, поиска, сортировки) элементов коллекции, используя исключения, выбрасываемые, если форматы или значения входных данных не соответствуют спецификациям параметров обрабатывающих их методов.

**5 Содержание отчета**

5.1 Название работы

5.2 Цель работы

5.3 Программные коды и скрины работы методов разработанного приложения Windows Forms в соответствии с вариантом задания

**6 Варианты заданий**

6.1 Коллекция «Товары»: название, вес, цена».

6.2 Коллекция «Студенты»: фамилия, специальность, группа».

6.3 Коллекция «Геометрические фигуры»: название (прямоугольник, треугольник, эллипс), ширина, высота».

6.4 Коллекция «Банки»: название (ВТБ 24, Альфа-банк, Возрождение), национальный рейтинг (AAA, AAA, AA), международный рейтинг (A, A-, BBB)».

6.5 Коллекция «Граждане»: фамилия, пол, дата рождения».

6.6 Коллекция «Специальности»: наименование (Компьютерные системы и комплексы, Программирование в компьютерных системах, Информационные системы (по отраслям)), код (09.02.01, 09.02.03, 09.02.04), квалификация специалиста (техник по компьютерным системам, техник-программист, техник по информационным системам)».

6.7 Коллекция «Мебель»: название (Стол интегральный, Стол компьютерный, Конференц-стол), артикул (А-33 R, А-10 R, А-13 П), размер Ш×Г×В (1400×900×750, 900×600×750, 1800×900×750)».

6.8 Коллекция «Города»: название (Москва, Санкт-Петербург, Ростов-на-Дону), численность населения (12 000 000, 5 000 000, 1 114 806), площадь (2550 кв.км, 1439 кв.км, 426,5 кв.км)».

6.9 Коллекция «Дисциплины»: наименование (Архитектура компьютерных систем, Системное программирование, Прикладное программирование), форма итоговой аттестации (экзамен, экзамен, экзамен), максимальная нагрузка (96 часов, 120 часов, 150 часов)».

6.10 Коллекция «ВУЗы»: название (Московский государственный технический университет имени Н.Э. Баумана, Санкт-Петербургский государственный университет, Национальный исследовательский ядерный университет МИФИ), год основания (1830, 1724, 1942), рейтинг (4,321, 4,168, 3,924)».

6.11 Коллекция «Государства»: название (Армения, Беларусь, Россия), столица (Ереван, Минск, Москва), плотность населения (107,9 чел./кв.км, 47,0 чел./кв.км, 8,3 чел./кв.км)».

6.12 Коллекция «Животные»: название (переднеазиатский леопард, тур, выдра), отряд (хищные, парнокопытные, хищные), семейство (кошачьи, полорогие, куньи)».

6.13 Коллекция «Автомобили»: марка (Toyota, Kia, Mazda), модель (Prius, Rio, CX-9), модификация (1.8, 1.4 хэтчбек, 3.7)».

6.14 Коллекция «Газеты»: название (Аргументы и факты, Собеседник, Вечерняя Москва), подписной индекс (40860, 99596, Е0079), периодичность (1 раз в неделю, 1 раз в неделю, 5 раз в неделю)».

6.15 Коллекция «Компьютер»: тактовая частота процессора, объем ОЗУ, емкость диска.

6.16 Коллекция «Служащий»: ФИО, табельный номер, оклад.

6.17 Коллекция «Накладная»: номер, дата отпуска, сумма.

**7 Используемая литература**

7.1 Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул Технология разработки программного обеспечения. Форум, Инфра-М, 2009

7.2 Эндрю Троелсен Язык программирования С# 2010 и платформа .NET 4.0, М., ВИЛЬЯМС, 2011

7.3 А. Марченко C#. Введение в программирование, М, Вильямс, 2009

7.4 http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.

"C:\Program Files\Geth\geth" --dev --rpc --rpcaddr "0.0.0.0" --rpccorsdomain "\*" --rpcapi "admin,debug,miner,shh,txpool,personal,eth,net,web3" console

"C:\Users\Маргарита\AppData\Local\Ethereum\keystore\geth" --dev --rpc --rpcaddr "0.0.0.0" --rpccorsdomain "\*" --rpcapi "admin,debug,miner,shh,txpool,personal,eth,net,web3" console

"C:\Program Files\Mist\Mist" --rpc http://localhost:8545 --swarmurl "null"

mklink /j C:\Users\Маргарита\AppData\Roaming\Ethereum C:\Ethereum\_blockchain

"C:\Program Files\Geth\geth" --mine --rpc --networkid 1999 --datadir C:\Users\Маргарита\AppData\Local\Ethereum\geth\chaindata console

"C:\Program Files\Geth\geth" --mine --rpc --networkid 1999 --allow-insecure-unlock console

public string Name { get; set; }

int age;

public int Age

{ get { return age; }

set { if (value < 0) { MessageBox.Show("Возраст не может быть меньше 0!"); age = Math.Abs(value); }

else age = Math.Abs(value);

}

}