# **Практическая работа № 3. Свойства и индексаторы**

Объём учебного времени – 2ч

Методические рекомендации

1. Цель работы: Научиться применять свойства и индексаторы.
2. Перечень необходимых средств обучения:
   1. Технические средства обучения:

Компьютер Core i3 3.0, 2 Gb оперативной памяти, винчестер 250 Gb, DVD

* 1. Программное обеспечение:
* Программа ОС Windows XP/7;
* Антивирусные программы: Kaspersky AntiVirus;
* Браузер Internet Explorer;
* Интегрированная среда программирования MS Visual Studio 2017.

1. Основные теоретические положения:

Свойство - это член класса, как поле или метод. Свойство в языке C# - это нечто среднее между полем и методом. Если исходить из того, как свойство используется, то оно, очень напоминает поле. Но принципы реализации свойства больше напоминают описание методов.

  При описании свойства определяются два специальных метода, которые непосредственно связаны со свойством. Эти методы называются аксессорами. Один метод (называется get-аксессором) вызывается при считывании значения свойства. Другой метод (называется set-аксессором) вызывается при присваивании значения свойству. Что будет происходить при вызове методов-аксессоров, определяет программист, когда описывает свойство в классе. Это может быть считывание/присваивание значения некоторому полю, определенный вычислительный процесс, обращение к массиву или другому объекту.

Шаблон, в соответствии с которым описывается свойство:

**тип имя** {

**get** {

// Код get-аксессора

}

**set** {

// Код set-аксессора

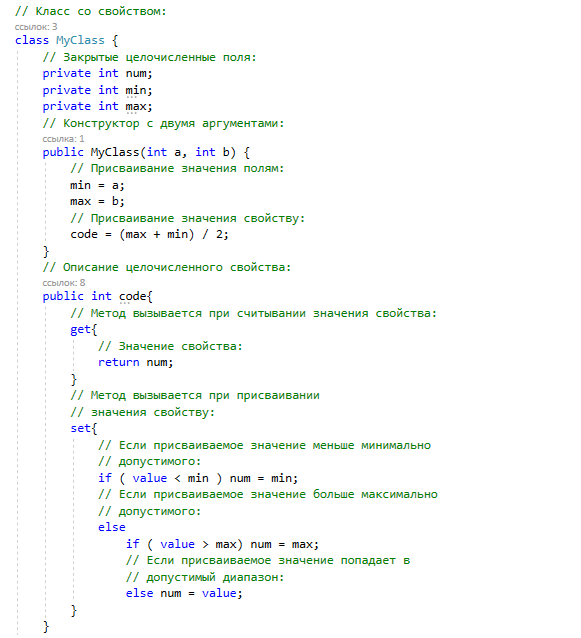
}

}

Еще одна важная особенность свойства состоит в том, что под свойство место в памяти не выделяется. То есть из того факта, что мы описали свойство, не следует, что при создании объекта под это свойство выделяется память. В этом смысле свойство принципиально отличается от поля.

Поскольку свойство не определяет область памяти и не связано с областью памяти, то свойство нельзя использовать с идентификаторами ref и out.

Пример:



В примере описывается класс с названием MyClass. В классе описаны три целочисленных закрытых поля num, min и max. В классе есть конструктор с двумя аргументами, а также имеется описание свойства с названием code. Свойство описано с идентификаторами public и int, что означает, что свойство открытое и целочисленное. Свойство описано с двумя методами-аксессорами. В теле get-акссссора всего одна команда return num, которой значением свойства возвращается значение поля num.

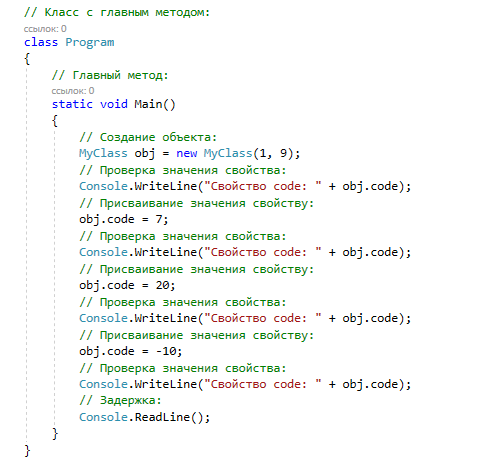
При присваивании значения свойству code будет вызываться set-аксессор, описанный для свойства. В соответствии с кодом этого аксессора выполняются вложенные условные операторы. В этих условных операторах проверяется присваиваемое свойству значение (это значение отождествляется с ключевым словом value). Если это значение меньше значения поля min (условие value < min истинно), то поле num получает значение min при выполнении команды num = min; .

Если присваиваемое свойству значение больше значения ноля max (условие value > max истинно), то полю num присваивается значение max, команда num = max;. Иначе (если присваиваемое свойству значение попадает в диапазон значений от min до max) полю num присваивается то значение, которое собственно указано справа от оператора присваивания: команда num = value;.

Value ключевое слово, обозначающее присваиваемое свойству значение. Идентификатор value отождествляется с некоторой переменной. Причем у переменной есть тип - это тип свойства, которому присваивается значение. В нашем примере value отождествляется с целочисленным значением (тип int).

   Получается, что при запросе значения свойства code в действительности возвращается значение поля num. Само поле num закрытое, поэтому вне кода класса значение этому полю напрямую присвоить не удастся. Значение ноля num изменяется при присваивании значения свойству code. Алгоритм присваивания не сводится к копированию присваиваемого значения в поле num. Мы хотим, чтобы значение поля num не могло быть меньше значения поля min, а также чтобы оно не могло быть больше значения поля max. Поэтому если присваиваемое свойству code значение попадает в диапазон значений от min до max, то поле num получает то значение, которое реально присваивается. Если же присваиваемое значение выходит за пределы допустимого диапазона значений, то поле num получает минимально/максимально возможное значение (в зависимости от того, за какую границу "выскакивает" присваиваемое значение).

  Что касается конструктора класса MyClass, то два его аргумента определяют минимальное и максимальное значение для свойства code (значения полей min и max), а командой code = (max + min) / 2; свойству code (а значит, и полю num) присваивается начальное значение.



В главном методе программы командой MyClass obj = new MyClass(1, 9); создается объект obj класса MyClass. Таким образом, значение свойства code объекта obj не может выходить за пределы значений от 1 до 9 включительно. Начальное значение свойства code равно 5 (поскольку (1+9)/2=5).

При выполнении команды obj.code = 7; свойство code (поле num) получит значение 7.

Выполнение команды obj.code = 20; приводит к тому, что свойство code (поле num) получит значение 9.

Наконец, выполнение команды obj.code = -10; приведет к тому, что свойство code (поле num) получит значение 1.

1. Практические указания:
   1. Напишите программу, в которой есть класс со свойством, для которого описан только get-аксессор, а set-аксессор не определен.
      1. Создайте консольное приложение.
      2. Объявите класс с именем MyClass с двумя открытыми целочисленными полями first и last.
   * добавьте конструктор класса с двумя целочисленными аргументами, в теле конструктора полям first и last присвойте значения соответствующих аргументов.
   * добавьте в класс открытое целочисленное свойство number (доступно только для чтения):
   * добавьте метод, вызывающийся при считывании значения свойства и возвращающий сумму полей first и last.
     1. В главном модуле:
   * создайте объект-экземпляр класса MyClass;
   * реализуйте проверку значения свойства объекта;
   * присвойте полю first целочисленное значение и проверьте измененное значение свойства;
   * присвойте полю last целочисленное значение и проверьте измененное значение свойства.

* 1. Напишите программу, которая по введенному значению отражает бинарный код числа. В программе имеется класс со свойством, содержащим set-аксессор, но не определен get – аксессор.
     1. Создайте консольное приложение.
     2. Объявите класс с именем MyClass. В классе должны быть определены:
  + закрытое текстовое поле code;
  + опишите конструктор класса с одним аргументом неотрицательного целочисленного типа, в теле конструктора выполните присваивание целого неотрицательного числа, которое присваивается свойству number;
  + выполните переопределение метода ToString(), возвращающего значение текстового поля;
  + добавьте в класс открытое беззнаковое целочисленное свойство number с описанием метода, вызывающегося при присваивании значения свойству:
  + опишите локальную беззнаковую целочисленную переменную num и инициализируйте ее значением value, которое присваивается свойству;
  + задайте начальное значение текстового поля – пустой текст;
  + в цикле do-while, условием завершения которого является равенство числа нулю, выполните вычисление битов числа: значение последнего (младшего) бита дописывается слева к тексту. Для вычисления последнего (младшего) бита в числе (значение переменой num) вычисляем остаток от деления этого числа на 2; выполните сдвиг кода числа на одну позицию вправо: num >>= 1;
    1. В главном модуле:
  + создайте объект класса MyClass, в качестве параметра передайте значение числа;
  + выполните проверку бинарного кода целого числа;
  + присвойте свойству целочисленное значение и проверьте новое значение бинарного кода целого числа.
  1. Напишите программу, которая иллюстрирует использование нескольких свойств в одном классе.
     1. Создайте консольное приложение.
     2. Объявите класс с именем MyClass.В классе должны быть определены:
  + закрытое поле nums, представляющее собой ссылку на целочисленный массив (этот массив будет использоваться при описании свойств с именами content, element и data);
  + открытое текстовое свойство content , которое можно прочитать, но нельзя присвоить. Свойство должно возвращать текстовую строку со значениями элементов массива. Для этого, при вызове get-аксессора для этого свойства в условной конструкции проверяется условие nums==null, истинное, если поле nums не ссылается на массив. Если так, то значением свойства возвращается строка "{}". Если же поле nums ссылается на массив, то формируется текстовая прока, содержащая значения элементов массива, для этого:
  + необходимо сформировать текстовую строку 
  + в цикле, по всей длине массива, добавить к строке элементы массива;
  + открытое целочисленное свойство element, которому можно присвоить значение, но нельзя узнать значение свойства. Эффект от его использования получается такой, что в массиве, на который ссылается поле nums, появляется дополнительный последний элемент. При присваивании значения свойству в массив nums добавляется новый элемент с присваиваемым значением. Для этого в теле set-аксессора в условной конструкции проверяется условие nums==null. Истинность условия означает, что поле nums на массив не ссылается.
  + при этом, необходимо создаеть массив из одного элемента ;
  + значение этого элемента есть присваиваемое свойству значение;
  + Если условие nums==null в условной конструкции ложно, то создается новый массив, размер которого на единицу больше размера массива, на который ссылается поле nums 
  + далее выполняется поэлементное копирование значений из массива nums во вновь созданный массив, при этом остается незаполненным один последний элемент;
  + элементу присваивается значение, которое по факту присваивается свойству element:
  + в поле nums записывается ссылка на новый массив;
  + открытое Свойство data является ссылкой на целочисленный массив. Для свойства необходимо описать два аксессора. Это означает, что свойству значением можно присвоить ссылку на массив и результатом свойства является также ссылка на массив.
  + в get-аксессоре необходимо создать массив такого же размера, что и массив, на который ссылается поле nums. Затем выполните копирование массивов. Значением свойства является ссылка на новый созданный массив, являющийся копией массива, на который ссылается поле nums;
  + в set-аксессоре создайте новый массив, и ссылка на него запишите в поле nums. Здесь необходимо учесть, что ключевое слово value, обозначающее присваиваемое свойству значение, относится к типу int[] (тип свойства data) - то есть является ссылкой на целочисленный массив и обрабатывается как массив. Размер такого массива вычисляется выражением value.Length: 
  + с помощью циклической конструкции выполняется копирование значений элементов из присваиваемого массива value в созданный массив nums.
    1. В главном методе:
  + создайте объект obj класса MyClass;
  + проверьте, что его поле nums на массив не ссылается;
  + присвойте несколько целочисленных значений свойству element и выполните проверку содержимого массива из объекта;
  + объявите переменную массива А и запишите в нее ссылку на копию массива, на который ссылается поле nums объекта obj;
  + чтобы проверить, что это именно копия, а не сам массив nums, командой obj.element = 12; в массив nums объекта obj добавляется еще один элемент;
  + проверьте, что содержимое массива А не изменилось;
  + еще раз проверьте содержимое массива из объекта;
  + создайте целочисленный массив В, инициализируйте его значениями, и ссылку на массив присвойте свойству data объекта obj;
  + измените значение первого элемента массива В: B[0] = 0;
  + проверьте, что массив B меняется, а массив, на который ссылается поле nums объекта obj, остается таким, как до выполнения предыдущей команды.
  1. . Напишите программу, которая иллюстрирует использование индексаторов.
     1. Создайте консольное приложение.
     2. Объявите класс с именем MyClass.В классе должны быть определены:
  + закрытое целочисленное поле code;
  + конструктор с одним символьным аргументом, в теле конструктора полю присвойте значение аргумента;
  + символьный индексатор с целочисленным индексом. В индексаторе опишите методы для считывания и присваивания значения выражения с проиндексированным объектом. В get – методе возвратите значение свойства. В set-методе выполните присваивание значения полю code;
    1. В главном модуле:
  + создайте объект obj класса MyClass;
  + в цикле выполните индексирование объекта для считывания значения: 
  + присвойте выражению с индексированным объектом символьное значение и еще раз в цикле прочитайте;
  + примените отрицательный индекс при индексировании объекта.
  1. Разработайте приложение, иллюстрирующее использование двумерного индексатора.
     1. Создайте консольное приложение.
     2. Объявите класс с именем MyClass. В классе должны быть определены:
  + закрытое поле, являющееся ссылкой на двумерный символьный массив;
  + конструктор с двумя аргументами, определяющими размерность массива. В теле конструктора создайте двумерный массив и заполните его символьными значениями ‘0’;
  + метод для отображения содержимого символьного массива;
  + двумерный индексатор типа char. В индексаторе опишите методы для считывания и присваивания значения выражения с проиндексированным объектом. В get – методе возвратите значение элемента массива. В set-методе элементу массива присвойте значение value;
    1. В главном модуле:
  + создайте объект obj класса MyClass;
  + выполните проверку содержимого массива;
  + выполните индексирование объекта символьными значениями и проверьте содержимое массива;
  + еще раз проиндексируйте объект уже имеющимися значениями и отобразите результат.

1. Контрольные вопросы
2. Свойство это…
3. Специальные методы свойства называются…?
4. Как называется метод, позволяющий прочитать информацию из свойства?
5. Как называется метод, позволяющий записать информацию в свойство?
6. Выделяется ли место в памяти для свойства?
7. Индексатор – это…
8. Какими бывают индексаторы?
9. Перегружаются ли индексаторы?
10. Список рекомендуемой литературы:

**Основная литература:**

1. Немцова Т.И. Програмиирование на языке выского уровня. Программирование на языке Object Pascal: учеб. пособие /Т.И. Немцова, С.Ю. Голова, И.В. Абрамова. – М.: ФОРУМ: ИНФРА – М, 2012. – 496 с.
2. Овечкин Г.В. Компьютерное моделирование: учебник для студ. учреждений сред. проф. образования / Г.В. Овечкин.- М.: Издательский центр «Академия», 2015.-224с.
3. Немцова, Т. И.  Программирование на языке высокого уровня. Программирование на языке С++ : учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев. - М. : ФОРУМ : ИНФРА-М, 2012. - 512 с. : ил. + CD.
4. Гуриков, С. Р. Введение в программирование на языке Visual C# : учеб. пособие / С. Р. Гуриков. - М. : ФОРУМ : ИНФРА-М, 2013. - 448 с.

**Дополнительная литература:**

1. Рао Сиддхартха Освой самостоятельно С++ за 21 день, 7 изд.: Пер с англ.-М.: ООО «И.Д.Вильямс», 2013 – 688 с.: ил. – Парал.тит.англ.
2. Голицына О.Л. Программное обеспечение: учеб. пособие для среднего профессионального образования-М.:ФОРУМ;ИНФРА-М,2006.-432 с.
3. Виллемер А. Программирование на С++/А. Виллемер;[пер. с нем. М.А.Райтман].-М.:Эксмо,2013.-528с.+CD.-(Мировой компьютерный бестселлер).
4. Культин Н.Б. Microsoft Visual C++ в задачах и примерах.- СПб.: БХВ-Петербург,2010.-272 с.:ил.+CD-ROM.
5. Партыка Т.Л. Операционные системы, среды и оболочки: учеб. пособие для студ. учреждений сред. проф. Образования/ Т.Л. Партыка, И.И. Попов. – 3-е изд., перераб. и доп. – М.: ФОРУМ, 2010. – 543 с.
6. Окулов С.М. Основы программирования: учебное пособие.- М.:БИНОМ. Лаборатория знаний, 2010.- 440 с.
7. Голицына О.Л. Основы алгоритмизации и программирования: учеб. пособие для сред. проф. образования.- М: ФОРУМ; ИНФРА-М,2005.-432 с.
8. Рихтер Дж. Программирование приложений для Microsoft Windows /Пер. с англ. – M.: Microsoft Press, 2003. – C.48-313.