Задание к лабораторной работе

В класс B добавить поле-массив. Разработать конструктор для инициализации массива, который при своем вызове передает управление собственному конструктору класса B. Размер массива – поле a, инициализация элементов массива: свойство c2 (см. лабораторную работу №2), умноженное на индекс элемента массива. В программе вывести на экран элементы массива. Для вывода использовать цикл foreach.

Контрольные вопросы

1. Что такое массив?
2. Как массив представляется в C#?
3. Какие виды массивов определяются в C#?
4. Какие назначение и логика работы цикла foreach?
5. Какое значение индекса первого элемента в массиве?

ЛАБОРАТОРНАЯ РАБОТА №4. ИНДЕКСАТОРЫ. СТАТИЧЕСКИЕ ПОЛЯ. ПАРАМЕТРИЗОВАННЫЕ КЛАССЫ

Цель лабораторной работы: Научиться работать с индексаторами, статическими полями и параметризованными классами.

Теоретические основы

Индексаторы являются синтаксическим удобством, позволяющим создавать класс, структуру или интерфейс, доступ к которому клиентские приложения получают, как к массиву. Чаще всего индексаторы реализуются для доступа к закрытой внутренней коллекции или закрытому массиву. Вместе с модификаторами доступа индексаторы реализуют механизм инкапсуляции для полей-массивов и являются аналогами свойств, определяемых для обычных полей.

Пример объявления индексатора

```
public class AClass1
   {
      private int[] imyArray = new int[20];
      public int  this[int ind1] //индексатор
      {
         get
         { return imyArray[ind1]; }
         set
         { imyArray[ind1] = value; }
      }
}
```

Статические поля – поля, принадлежащие классу. Они объявляются с ключевым словом static. Основное отличие от обычных полей – для обращения к статическим полям не требуется создание объекта. Доступ осуществляется напрямую через имя класса. Более того, через объекты к статическим полям обратиться нельзя.

Пример объявления статического поля:

public static int I;

Параметризованные классы – классы, позволяющие определить тип своих аргументов при непосредственном создании объектов.

Пример параметризованного класса:

```
public class AClass1<T>
{
   private T[] imyArray = new T[20];
 }
public class M
{
   static void Main(string[] args)
   {
      AClass1<string> K = new AClass1<string>();
      AClass1<int> K2 = new AClass1<int>();
   }
}
```

Основное ограничение, налагаемое на параметризованные классы при их создании: необходимо следить, чтобы операции, используемые для типа-параметра, были определены для всех типов или же использовать механизмы преобразования типов.

Задание к лабораторной работе

В классе B определить индексатор для исходного массива. Вывести в программе на экран элементы массива через индексатор. Добавить в B еще один массив и определить индексатор и для него. Вывести на экран значения элементов второго массива через индексатор. Второй массив инициализировать при описании (то есть НЕ в конструкторе) . Создать параметризованный класс C со статическим полем. В программе продемонстрировать умение работы со статическим полем и параметризацией класса. В качестве параметров взять строковый тип и числовой тип (то есть создать 2 объекта с разными параметрами). Статическое поле – тип строка.

Контрольные вопросы

1. Что такое индексатор?
2. Сколько индексаторов может быть у класса?
3. В чем отличие статических полей от обычных?
4. Что такое параметризованные классы?

5. Что необходимо учитывать при проектировании параметризованных классов?

ЛАБОРАТОРНАЯ РАБОТА №5. ПЕРЕОПРЕДЕЛЕНИЕ ОПЕРАЦИЙ

Цель лабораторной работы: научиться переопределять операции для классов. В частности разобраться с определением критериев истинности для объектов классов и перегрузкой логических операций.

Теоретические основы

Перегрузка операций в C# позволяет определять смысл стандартных операций C# (+, - и т. д.) для классов, определяемых пользователем. Например, что значит, сложить два объекта класса A. Перегрузка операций строится на основе открытых статических функций-членов, объявленных с использованием ключевого слова operator.

Не все операции могут быть перегружены. Существуют определенные правила и ограничения на перегрузку операций:

+, –, !, ~, ++, —, true, false	Унарные символы операций, допускающие перегрузку. true и false также являются операциями
+, –, *, /, %, &, |, ^, <<, >>	Бинарные символы операций, допускающие перегрузку
==, !=, <, >, <=, >=	Операции сравнения перегружаются
&&, ||	Условные логические операции моделируются с использованием ранее переопределенных операций & и |
[]	Операции доступа к элементам массивов моделируются за счет индексаторов
+=, –=, *=, /=, %=, &=, =, ^=, <<=, >>=	Операции не перегружаются по причине невозможности перегрузки операции присвоения
=, ., ?:, –>, new, is, sizeof, typeof	Операции, не подлежащие перегрузке

Правила:

1. Префиксные операции ++ и — перегружаются парами;
2. Операции сравнения перегружаются парами: если перегружается операция ==, также должна перегружаться операция !=, < и >, <= и >=.
3. Операции true и false также перегружаются парами. В этом случае для объекта класса определяются критерии истинности. Необходимо следить, чтобы критерии истинности, определенные в операции true и в операции false, не противоречили друг другу.

Синтаксис:

public static <тип возвращаемого значения> operator <операция>(<параметры>)

Пример

```
class Program
  {
    public static Program operator ++(Program par1)
    {
      par1.x++;
      return par1;
    }
  }
```

Задание к лабораторной работе

Для класса B переопределить операции согласно варианту (см. вариант в таблице 3). В основной программе продемонстрировать использование переопределенных операций.

Таблица 3. Варианты заданий для лабораторной работы №5

Вариант	Операции	Вариант	Операции	Вариант	Операции
1	true, false, &	10	true, false, &	19	true, false, &
2	true, false, |	11	true, false, |	20	true, false, |
3	true, false, !	12	true, false, !	21	true, false, !
4	true, false, &	13	true, false, &	22	true, false, &
5	true, false, |	14	true, false, |	23	true, false, |
6	true, false, !	15	true, false, !	24	true, false, !
7	true, false, &	16	true, false, &	25	true, false, &
8	true, false, |	17	true, false, |	26	true, false, |
9	true, false, !	18	true, false, !	27	true, false, !

Контрольные вопросы

1. Какой базовый принцип ООП лежит в основе переопределения операций?
2. В чем особенность переопределения логических операторов?
3. Какие принципы следует учитывать при переопределении операций?
4. Какие операции не подлежат переопределению?
5. Какие операции моделируются за счет других?