Реализация принципов ООП в .NET

- Константы
- Статические методы, статические переменные
- Пользовательский индексатор
- Перегрузка методов
- Перегрузка операторов

Константы

Константы – это постоянные значения, которые известны во время компиляции и не изменяются во время выполнения программы.

```
abstract  Class  MyClass {
public const  int  myConst1 = 10;
public const  string  myConst2 = “”; }

Console.WriteLine(MyClass.myConst1);

MyClass.myConst1 = 15; //ошибка!
```

Статические методы

Статический метод – это метод, который не имеет доступа к полям объекта.
Для вызова такого метода не нужно создавать экземпляр класса (объект), в котором он объявлен.

Статический метод может быть вызван только напрямую через уровень класса, т.е. без создания экземпляра объекта данного класса.

Статические методы

```
Class  Hospital()
{
      public static string Complain()
      { … }
}

static void Main(string[] args)
{
     Console.WriteLine(Hospital.Complain());
}
```

Статические переменные

Статические переменные используются всеми объектами того класса, в котором эти данные были определены.

Статические данные предназначены для хранения информации на уровне всего класса, а не отдельных объектов.

Статические переменные

```
Class Hospital()
{    private  static  int  vizitCount ;

     public Hospital()  {  vizitCount++; }
     public  int  VizitCount  { get; }
}

Class Class1()
{    Hospital h1 = new Hospital();
     Hospital h2 = new Hospital();

     Console.WriteLine("Посетителей " + h2.VizitCount );
}
```

Пользовательский индексатор

Индексаторы позволяют индексировать объекты и обращаться к данным по индексу.

Индексатор представляет собой слегка измененное свойство C#.

С помощью индексаторов с объектами можно работать как с массивами.

Пользовательский индексатор

```
class Student
{
        int nom;    string name;
        private Student[]  stArray;

        public Student(int ch)
        {    stArray = new Student[ch];    }

        public  Student(int  _nom, string  _name)
        {
            nom = _nom;  name = _name;
        }
```

12 слайд

```
        public Student this[int pos] {…}   //индексатор
}
```

Пользовательский индексатор

```
//индексатор класса Student:
public  Student  this[int  pos]
{
      get
      {
      return (stArray[pos]);
       }
       set
       {
            stArray[pos] = value;
       }
}
```

Пользовательский индексатор

```
static void Main(string[] args)
{
   Student  st  = new  Student(2);
   st[0] = new  Student(20, "Иванов");
   st[1] = new  Student(25, "Сидоров");


   for (int i = 0; i < 2; i++)
   {
      Console.WriteLine(st[i].Surname);
   }

   Console.ReadLine();
}
```

Пользовательский индексатор

Можно воспользоваться **foreach,** для этого необходимо явно определить методы интерфейсов:

```
public IEnumerator GetEnumerator()
{  return this; }
public object Current
{
   get { return arrStudent[index]; }
}
int index = -1;
public bool MoveNext()
{
   if (index == arrStudent.Length - 1)
   {
      Reset();
      return false;
   }
   index++;
   return true;
}
public void Reset()
{  index = -1; }
```

Пользовательский индексатор

```
static void Main(string[] args)
{
   Student  st = new Student(3);
   st[0] = new Student(10, "Иванов");
   st[1] = new Student(12, "Сидоров");
   st[2] = new Student(15, "Петров");

   foreach (Student  s  in  st)
      Console.WriteLine(s.GetName());
}
```

Перегрузка методов

Перегрузка методов – это объявление методов в одном классе с одинаковыми именами, но с различными параметрами.

Перегрузка методов относится к одному из способов реализации полиморфизма в C#.

Перегрузка методов

```
public void  Mt()
{
    Console.WriteLine("Метод Mt без параметров");
}
public void  Mt(string name, int ch)
{
    Console.WriteLine("Метод Mt с параметрами: {0} {1}", name,
ch);
}


Class1  cl = new Class1();
// Разные реализации вызова перегружаемого метода
cl.Mt();
cl.Mt("Иван", 10);
```

Перегрузка операторов

Перегрузка операторов позволяет переопределить операторы C# для применения их к типам, определенным пользователем.

Перегрузка операторов в программировании – один из способов реализации полиморфизма.

Перегрузка операторов допускает возможность существования нескольких вариантов применения оператора, имеющих одно и то же имя. Главное различие – это типы параметров, к которым они применяются.

Перегрузка операторов

Оператор «+»:

```
int a = 5;
int b = 10;

int c = a + b;
```

Оператор «+» перегружен в классе String:

```
string s1 = «Привет»;
string s2 = « всем!»;

string s3 = s1 + s2;
```

Перегрузка операторов

Пример пользовательской перегрузки оператора «+»:

```
class Point
{
   private  int  x,  y;
   public  Point(int _x,  int _y)
   {
      x = _x;
      y = _y;
   }
   public  static  Point  operator +  (Point  pl1, Point  pl2)
   {
      Point   newPoint  =  new  Point(pl1.x + pl2.x, pl1.y + pl2.y);
      return   newPoint;
   }
   public  override  string  ToString()
   {
      return  "X=" + x + " Y=" + y;
   } }
```

Перегрузка операторов

```
static void Main(string[] args)
{
    Point pt1 = new Point(100, 200);
    Point pt2 = new Point(50, 60);

    Point pt3 = pt1 + pt2;
    Console.WriteLine(pt3.ToString());
}
```

Перегрузка операторов

Общая форма перегрузки унарного оператора:

```
public static возвращаемый_тип operator op(тип_параметра операнд)
{…}
```

Общая форма перегрузки бинарного оператора:

```
public static возвращаемый_тип operator op(тип_параметра1 операнд1, тип_параметра2 операнд2)
{ … }
```

Перегрузка операторов

Возможность перегрузки операторов C#

Оператор C#	Можно ли произвести перегрузку
+, -, !, ~, ++, --, true, false	Любой оператор из этого набора унарных операторов может быть перегружен
+, -, *, /, %, &, |, ^, <<, >>	Любой оператор из этого набора бинарных операторов может быть перегружен
= =, !=, <, >, <=, >=	Все эти операторы сравнения могут быть перегружены. Однако следующие операторы могут быть перегружены только парами: > и <, <= и >=, = = и !=
[]	Этот оператор не может быть перегружен. Однако можно применять его к конкретному классу, если в нем реализован метод индексатора

Лабораторная работа 5

1. Перегрузить оператор «-» для чисел таким образом, чтобы он выполнял операцию сложения.

2. (Пользовательский индексатор)
   Вывести список планет солнечной системы, имеющих больший радиус, чем радиус земли.