Тема 3. Использование переменных значимых типов

Парадигмы программирования. C#

План

- Общая система типов
- Именование переменных
- Использование встроенных типов данных
- Создание собственных типов данных
- Приведение значимых типов данных

Общая система типов

Часть 1

Ссылочные и значимые типы данных

- Значимые типы данных
  - Непосредственно содержат данные
  - У каждой переменной есть своя копия данных
  - Операции с одной переменной не могут повлиять на данные другой переменной
- Ссылочные типы данных
  - Хранят ссылку на данные (известны как объекты)
  - Две переменных ссылочных типов данных могут ссылаться на один объект
  - Операции с одной переменной ссылочного типа данных могут повлиять на другие переменные

Встроенные и пользовательские типы данных

- Встроенные значимые типы данных
  - int
  - float
- Пользовательские значимые типы данных
  - enum
  - struct

Простейшие типы данных

- Тип данных CLR: System.Int32
- Соответствующий тип данных C#: int
- http://msdn.microsoft.com/en-us/library/ya5y69ds.aspx

Зарезервированные слова C# и встроенные типы данных CLR

Зарезервированное слово (псевдоним)	Тип данных
sbyte	System.SByte
byte	System.Byte
short	System.Int16
ushort	System.UInt16
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
char	System.Char
float	System.Single
double	System.Double
bool	System.Boolean
decimal	System.Decimal

Именование переменных

Часть 2

Правила

- Использовать буквы, цифры и символ подчёркивания
- Идентификатор не может начинаться с цифры
- Идентификатор не может совпадать с ключевым словом
- Идентификаторы отличаются по регистру символов
- Не используйте только прописные буквы
- Не начинайте с подчёркивания
- Не используйте непонятных сокращений

Ключевые слова C#

- abstract
- as
- base
- bool
- break
- byte
- case
- catch
- char
- checked
- class
- const
- continue
- decimal
- default
- delegate
- do
- double
- else
- enum
- event
- explicit
- extern
- false
- finally
- fixed
- float
- for
- foreach
- goto
- if
- implicit
- in
- int
- interface
- internal
- is
- lock
- long
- namespace
- new
- null
- object
- operator
- out
- override
- params
- private
- protected
- public
- readonly
- ref
- return
- sbyte
- sealed
- short
- sizeof
- stackalloc
- static
- string
- struct
- switch
- this
- throw
- true
- try
- typeof
- uint
- ulong
- unchecked
- unsafe
- ushort
- using
- virtual
- void
- volatile
- while

Руководство по именованию в CLR

- http://msdn.microsoft.com/en-us/library/xzf533w0(VS.71).aspx

Использование встроенных типов данных

Часть 3

Декларирование переменных

- `int elementCount;`
- `int elementCount, penNumber;`
- ```
  int elementCount,
    penNumber;
  ```
- `char firstLetter = 'A';`
- Перед использованием переменная должна быть инициализирована
  ```

Инициализация переменных / операция присваивания

- ```
  int elementCount;
  elementCount = 45;
  ```
- ```
  int elementCount = 45;
  ```
- ```
  char firstLetter = 'A';
  ```
  ```

Сложное присваивание

- `elementCount = elementCount + 5;`
- `elementCount += 5;`
- `elementCount -= 8;`
- Другие варианты
  - `*=`
  - `/=`
  - `%=`

Общие операции

Операция	Пример
Равенства	`== !=`
Сравнения	`< > <= >= is`
Условные	`&& || ?:`
Битовые	`<< >> & | ^`
Инкрементации	`++`
Декрементации	`--`
Арифметические	`+ - * / %`
Присваивания	`= *= /= %= += -= <<= >>= &= ^= |=`

Инкрементация и декрементация

- Общее определение
  - `elementCount += 1;`
  - `elementCount -= 1;`
- Сокращённое определение
  - `elementCount++;`
  - `elementCount--;`
- Альтернативное сокращённое определение
  - `++elementCount;`
  - `--elementCount;`

Особенности операции присваивания

- Пример 1
  - `int itemCount = 0;`
  - `Console.WriteLine(itemCount = 2); // Prints 2`
  - `Console.WriteLine(itemCount = itemCount + 40); // Prints 42`
- Пример 2
  - `int itemCount = 0;`
  - `Console.WriteLine(itemCount += 2); // Prints 2`
  - `Console.WriteLine(itemCount -= 2); // Prints 0`
- Пример 3
  - `int itemCount = 42;`
  - `int prefixValue = ++itemCount; // prefixValue == 43`
  - `int postfixValue = itemCount++; // postfixValue = 43`

Порядок выполнения операций

- Все бинарные операции являются лево-ассоциативными (выполняются слева направо) за исключением операций присваивания и условных операций
- Операции присваивания и условные операции являются право-ассоциативными (выполняются справа налево)

Создание собственных типов данных

Часть 4

Перечисления

- Декларирование

```
enum FlagColor { White, Blue, Red }
```

- Использование

```
FlagColor color; // Declare the variable
color = FlagColor.Red; // Set value
color = (FlagColor)2; // Type casting int to Color
```

- Отображение значения

```
Console.WriteLine("{0}", color);
```

Истинное лицо перечислений

- На самом деле все перечисления являются реализациями типа System.Enum
- http://msdn.microsoft.com/en-us/library/system.enum.aspx
- Основные методы
  - GetValues()
  - GetNames()
  - GetValue()
  - GetName()
  - Parse()

Флаги

```
[Flags]
enum Contract
{
    Designing = 0x0,
    Coding = 0x1,
    Testing = 0x2,
    Deploying = 0x4,
    Everything = Designing | Coding | Deploying
}
class MyClass
{
    Contract contract1 = Contract.Designing |
        Contract.Coding;
    Contract contract2 = Contract.Everything;
}
```

Структуры

- Декларирование

```
public struct Employee
{
    public string firstName;
    public int age;
}
```

- Использование

```
Employee companyEmployee; // Declare variable
companyEmployee.firstName = "Sam"; // Set value
companyEmployee.age = 43;
```

Приведение значимых типов данных

Часть 5

Неявные преобразования

- Преобразование `int` в `long`

```
using System;
class Test
{
    static void Main( )
    {
        int intValue = 123;
        long longValue = intValue;
        Console.WriteLine("(long) {0} = {1}", intValue,
            longValue);
    }
}
```

- Неявные преобразования не могут привести к исключению
  - Значение не может потеряться

Явные преобразования

- Преобразование `long` в `int` без проверки

```
using System;
class Test
{
    static void Main( )
    {
        long longValue = Int64.MaxValue;
        int intValue = (int) longValue;
        Console.WriteLine("(int) {0} = {1}",
            longValue, intValue);
    }
}
```

Явные преобразования (продолжение)

- Преобразование `long` в `int` с проверкой

```
using System;
class Test
{
    static void Main( )
    {
        checked
        {
            long longValue = Int64.MaxValue;
            int intValue = (int) longValue;
            Console.WriteLine("(int) {0} = {1}",
                longValue, intValue);
        }
    }
}
```

Заключение

- Спасибо за внимание!