Семинар 1. Повторение

Чуйкин Николай Константинович

15 января 2020 г.

Формула оценивания

$$О_{СР3} = (\frac{СР3_1}{4} + \frac{СР3_2}{4} + \frac{СР3_3}{4} + \frac{СР3_4}{4})$$

$$О_{СР4} = (\frac{СР4_1}{3} + \frac{СР4_2}{3} + \frac{СР4_3}{3})$$

$$О_{н4} = \text{ОКРУГЛ}(0.1 \cdot О_{СР3} + 0.3 \cdot О_{КДЗ3} + 0.2 \cdot О_{КР3} + 0.15 \cdot О_{СР4} + 0.25 \cdot О_{КР4})$$

$$О_{и4} = \text{ОКРУГЛ}(0.3 \cdot О_{н} + 0.1 \cdot О_{и2} + 0.6 \cdot О_{э4})$$

Нововведения

- Контесты

Нововведения

- Контесты
- Лабараторные

Нововведения

- Контесты
- Лабараторные
- Не будет СР на 10 минут

Контесты

- 8 практических задач
- Используются шаблоны
- Предполагается, что их необходимо выполнять на практических занятиях и на лабараторных
- Используется Я.Контест
- Используются **новые** пароли
- Пока используется компилятор Mono C# 4.5

Лабараторные

- Посещение обязательно
- Проходят в четверг на 2 и 3 паре в D210 (можно приходить на любую)
- На лабораторных всегда присутствуют Я и/или ассистенты
- В начале рассказывается небольшой практический материал
- Остальное время отведено на решение задач

Оценивание

$$\text{Ki} = 0,8 \cdot \frac{\text{кол-во реш. задач}}{\text{кол-во задач}} + 1 * \text{посещение семинара} + 1 * \text{посещение лабараторной}$$

В случае двух контестов:

$O_{\text{CP}} = (0.4 \cdot \min(\text{K1},\text{K2}) + 0.6 \cdot \max(\text{K1},\text{K2}))$

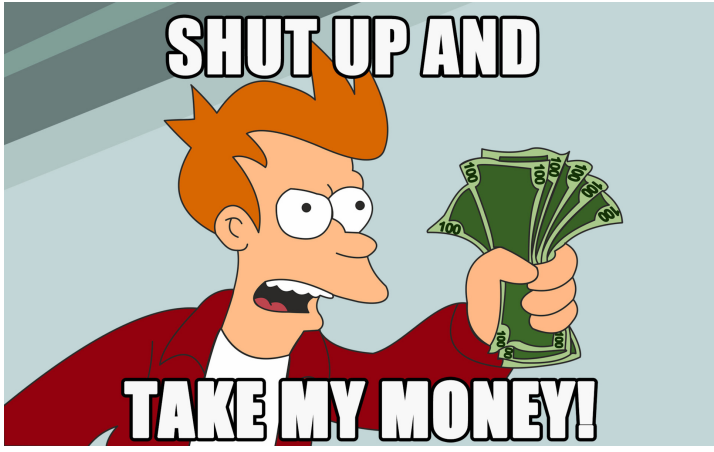
В случае трёх контестов:

$O_{\text{CP}} = (0.4 \cdot \max(\text{K1},\text{K2},\text{K3}) + 0.4 \cdot (\text{K1}+\text{K2}+\text{K3}-\max(\text{K1},\text{K2},\text{K3})-\min(\text{K1},\text{K2},\text{K3}))$
$+0.2 \cdot \min(\text{K1},\text{K2},\text{K3}))$

Нововведения

- Контесты
- Лабараторные
- Не будет СР на 10 минут

Нам нравится

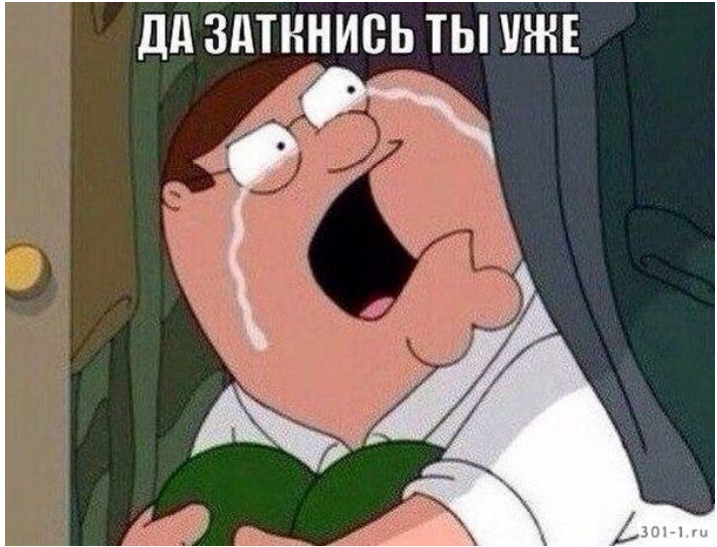


Нам нравится

Форма обратной связи:
`http://bit.ly/progChuykin`
Ссылка на github (с шаблонами и презентациями):
`https://github.com/nkchuykin/Programming`

Чат в Telegram

Нам не нравится



Нам не нравится

Вы можете перейти в другую группу по дисциплине ”Программирование”к другому преподавателю, обратитесь ко мне по почте **nchuykin@hse.ru**.

Повторение

- Методы
- Члены типов
- Ссылка this

Методы

- Передача параметров в метод
- Экземплярные методы
- Статические
- Перегрузка методов

Передача параметров в метод

Параметры в метод могут передаваться:

Передача параметров в метод

Параметры в метод могут передаваться:

- По значению

Передача параметров в метод

Параметры в метод могут передаваться:

- По значению
- По ссылке

Передача параметров в метод

Параметры в метод могут передаваться:

- По значению
- По ссылке

По умолчанию значения в метод передаются по значению.
А что с ссылочными типами?

Передача параметров в метод

Параметры в метод могут передаваться:

- По значению
- По ссылке

По умолчанию значения в метод передаются по значению.
А что с ссылочными типами?
Так как значение переменной ссылочного типа это адрес переменной в памяти, то изменить то, что находится по этому адресу возможно, но не сам адрес.
Для передачи параметров по ссылке используются модификаторы параметров **ref** и **out**

Передача параметров в метод по значению

```
int Sum(int a, int b)
{
    return a + b;
}
```

Передача параметров в метод с модификатором **ref**

```
void Swap(ref int a, ref int b)
{
    int c = a;
    a = b;
    b = c;
}
```

Передача параметров в метод с модификатором **out**

```
bool TryConvert(int val, out uint result)
{
    result = (uint)val;
    return val >= 0;
}
```

Экземплярные методы

- Получают неявно ссылку на текущий объект `this`
- Имеют доступ ко всем членам класса

Статические методы

- Имеют доступ только к статическим членам класса
- Могут явно получать параметром ссылку на объект своего типа

Экземплярные методы

```
class A
{
    int a;
    static int b;

    public void Count(int a)
    {
        this.a = a;
        b += a;
    }
}
```

Статические методы

```
class A
{
    int a;
    static int b;

    public static void Count(int a)
    {
        // this.a = a; Compile Error!
        b = a;
    }
}
```

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

- Название

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

- Название
- Типы параметров

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

- Название
- Типы параметров
- Количество параметров

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

- Название
- Типы параметров
- Количество параметров
- Модификатор ref/out у параметров

Перегрузка методов

Сигнатура метода:

```
public static int Method(ref int x,double d)
```

К сигнатуре методов при перегрузке относятся:

- Название
- Типы параметров
- Количество параметров
- Модификатор ref/out у параметров

К сигнатуре методов при перегрузке не относятся:

- Тип возвращаемого значения
- Модификатор доступа
- Модификатор `static`

Почему?

Члены типов

В CLS существуют только следующие члены типов:

Члены типов

В CLS существуют только следующие члены типов:

- Поля

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

- Поля

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

- Поля
- Делегат

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

- Поля
- Делегат
- Методы
- Свойства
- Индексаторы
- Конструкторы

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

- Поля
- Делегат
- Методы
- Свойства
- Индексаторы
- Конструкторы
- Деструкторы

Кто может вызывать деструктор класса?

Члены типов

В CLS существуют только следующие члены типов:

- Поля
- Методы

В языке C# тип может содержать в себе:

- Поля
- Делегат
- Методы
- Свойства
- Индексаторы
- Конструкторы
- Деструкторы

Кто может вызывать деструктор класса?
Garbage Collector

Ссылка `this`

Ссылка `this` используется:

Ссылка `this`

Ссылка `this` используется:

- Для разрешения конфликта между названием поля и названием параметра в методе

Ссылка `this`

Ссылка `this` используется:

- Для разрешения конфликта между названием поля и названием параметра в методе
- Вызова конструктора этого же класса с другим набором параметров

Ссылка `this`

Ссылка `this` используется:

- Для разрешения конфликта между названием поля и названием параметра в методе
- Вызова конструктора этого же класса с другим набором параметров
- Как модификатор параметра в методе расширения

Методы расширения

Методы расширения позволяют вызывать Ваш метод через ссылку на объект другого типа.

Метод расширения описывается в статическом классе и должен быть статическим. Первый параметр, объект расширяемого типа, помечается модификатором `this`

Методы расширения используются только если класс изолированный и нет доступа к исходному коду.

Пример такого класса:

Методы расширения

Методы расширения позволяют вызывать Ваш метод через ссылку на объект другого типа.

Метод расширения описывается в статическом классе и должен быть статическим. Первый параметр, объект расширяемого типа, помечается модификатором `this`

Методы расширения используются только если класс изолированный и нет доступа к исходному коду.

Пример такого класса:
`string`

Методы расширения

```
public static class StringExstension {
    public static int DigitsCount(this string str) {
        int k = 0;
        foreach (var item in str)
            if (item >= '0' && item <= '9')
                k++;
        return k;
    }
}
class Program {
    static void Main(string[] args) {
        Console.WriteLine("123".DigitsCount()) ;
    }
}
```

Бонус

- List<T>
- Dictionary<T,V>

List<T>

Представляет из себя динамически расширяемый массив
Основные члены:

- `List<int> l = new List<int>();`
- `l.Add(3);`
- `l.Count`
- `l[i]`

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*
- `Console.WriteLine(numbers[2])`

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*
- `Console.WriteLine(numbers[2])`
- *EXCEPTION!!!*

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*
- `Console.WriteLine(numbers[2])`
- *EXCEPTION!!!*
- `Console.WriteLine(numbers.ContainsKey(2)`

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*
- `Console.WriteLine(numbers[2])`
- *EXCEPTION!!!*
- `Console.WriteLine(numbers.ContainsKey(2)`
- *False*

Dictionary<TKey,TValue>

Представляет из себя словарь (коллекцию сопоставляющую пару значений)
Основные члены:

- `Dictionary<int, string> numbers = new Dictionary<int, string>();`
- `numbers[3] = "three"`
- `Console.WriteLinr(numbers[3])`
- *three*
- `Console.WriteLine(numbers[2])`
- *EXCEPTION!!!*
- `Console.WriteLine(numbers.ContainsKey(2)`
- *False*
- Как посчитать количество различных цифр в числе используя словарь?

Dictionary<TKey,TValue>

```
Dictionary<int, int> digits = new Dictionary<int, int>();
int n = 120321278;
while (n>0) {
    int d = n % 10;
    if (!digits.ContainsKey(d))
        digits[d] = 1;
    else
        digits[d] += 1;
    n /= 10;
}
for (int i = 0; i < 10; i++) {
    if (digits.ContainsKey(i))
    Console.WriteLine(i+": "+digits[i]);
}
```

Dictionary<TKey,TValue>

```
0: 1
1: 2
2: 3
3: 1
7: 1
8: 1
```

Спасибо за внимание