В.В. Подбельский

Иллюстрации к курсу лекций по дисциплине «Программирование на C#»

01

Программа на C#. Элементы синтаксиса

Благодарность Д.Д. Сагалову за помощь в переработке презентаций

Структура Простой Программы

```
// using-директивы
using System;

class Program // Класс приложения
{
static void Main() // Метод Main - точка входа
    {
        // Код решения задачи
    } // Main()
} // class Program
```

Обратите внимание: C# - объектно-ориентированный язык! Любая программа явно (или неявно, начиная с C# 9.0) содержит хотя бы 1 тип данных – в данном случае, class Program.

Упрощение Написания Программ – C# 9.0

```
using System;

// Работает с C# 9.0! Класс Program
// генерируется неявно.
Console.WriteLine("Hello world!");
```

Начиная с C# 9.0, написание коротких программ было значительно упрощено за счёт top-level statements.

Помните: в данном случае мы не отходим от объектно-ориентированной парадигмы! Класс программы, метод Main и (опционально) параметр args генерируются неявно.

Роль Декларации using

```
System.Console.WriteLine("Для нажмите выхода ENTER.");
System.Console.ReadLine();
```

```
using System;

Console.WriteLine("Для нажмите выхода ENTER.");
Console.ReadLine();
```

Комментарии в Коде на C#

- Однострочные:

```
// Это однострочный комментарий
```

- С двумя ограничителями:

```
int b, /* начало */ e = 1; /* конец */
```

- Документирующие:

```
/// <summary>
/// Этот текст будет содержимым
/// элемента XML-документа
/// </summary>
```

Консольный Ввод-Вывод Данных

Средства вывода:

```
System.Console.Write("text1");
System.Console.WriteLine("text2");
```

Чтение строки:

```
string line = System.Console.ReadLine();
```

Получение целого значения с проверкой корректности:

```
int.TryParse(строковое_представление, out переменная);
```

Получение вещественного значения с проверкой корректности:

```
double.TryParse(строковое_представление, out переменная);
```

Программа на Обработку Ввода Чисел

```
// Задание: Ввести вещественное число, вывести
// целую и дробную части числа.


class Program // Класс приложения.
{
    static void Main() // Точка входа.
    {
        double real;  // Вещественное число.
        string input; // Вводимая строка.
           // Ввести число, обработать, вывести результат
        System.Console.WriteLine("Выход - ENTER");
        System.Console.ReadLine();
    } // Main()
} // class Program
```

Ввести Число, Обработать, Вывести Результат

```
using System;

Console.Write("Введите число:");
string input = Console.ReadLine();
if (!double.TryParse(input, out double real))
{
    Console.WriteLine("Ошибка ввода");
    return; // Избавляемся от последующего else-блока.
}
int integer = (int)real;    // Целая часть.
double fraction = real - integer;   // Дробная часть.
Console.WriteLine($"integer = {integer};  fraction = {fraction}");
Console.WriteLine("целая часть: " + ((int)real));
Console.WriteLine("дробная часть: " + (real - (int)real));
```

Результат выполнения:

Введите вещественное число: 4,9

integer = 4; fraction = 0,9

Для выхода нажмите ENTER.

Идентификаторы C#

Правильные:

@if

_007

якорь

s_2_4_6

день_недели

Ошибочные:

if

666

номер дома

name@

mail.ru

- Идентификаторы не могут начинаться с цифр или содержать пробелов;
- Идентификаторы могут содержать буквенные символы Юникода, десятичные числа, символы соединения Юникода, несамостоятельные знаки Юникода или символы форматирования Юникода;
- Идентификатор не может содержать символов @ и $ (@ допустима как первый символ для получения буквального идентификатора).

Ключевые (Служебные) 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	

Контекстно-Ключевые Слова C#

add	**and**	**alias**	**ascending**	**async**
await	**by**	**descending**	**dynamic**	**equals**
from	**get**	**global**	**group**	**init**
into	**join**	**let**	**managed** *(*)*	**not**
nint	**notnull**	**nuint**	**on**	**or**
orderby	**partial** *(тип)*	**partial** *(метод)*	**record**	**remove**
select	**set**	**unmanaged** *(*)*	**unmanaged** *(**)*	**value**
var	**when** *(условие фильтра)*	**where** *(**)*	**where** *(предложение запроса)*	**with**
yield				

(*) – соглашение о вызове по указателю на функцию;
(**) – ограничение типизирующих параметров.

Метод Main()

```
static int Main()
{
    Операторы
}
```

```
static int Main(string[] args)
{
    Операторы
}
```

```
static void Main()
{
    Операторы
}
```

- Main обязательно должен быть помечен **static**.
- Результатом работы Main( ) может быть только: **void** , **int**, **Task** или **Task<int>** (последние два варианта допустимы при работе с асинхронными сценариями);
- Можно указать **string[ ]** – единственный допустимый параметр для Main( ), т. е. набор аргументов, получаемых из командной строки.

Блок, Оператор и Операция в C#

```
{ // Блок – набор операторов, выполняемых последовательно.
    int group = 5; // оператор – выражение из нескольких инструкций.
    System.Console.WriteLine("Номер группы: " + group);
}

int num1 = 10;
int num2 = 20;
int num3 = 30;
// Несколько операций «+» в рамках одного оператора.
System.Console.WriteLine(Сумма: " + (num1 + num2 + num3));
```

Попробуйте убрать круглые скобки