Тема 4. Операторы и операции языка. Оператор присваивания. Составные операторы присваивания. Ввод-вывод данных. Класс Math

Учебные вопросы:

1. Операторы языка C#.
2. Оператор присваивания.
3. Арифметические операторы.
4. Ввод-вывод данных. Класс Console.
5. Класс Math.
6. Класс Random.
7. Класс Convert.

1. Операторы и операции языка C#.

Операторы в C# - это специальные символы или ключевые слова, которые используются для выполнения различных операций над операндами (значениями или переменными). Они обеспечивают механизм для манипулирования данными и создания выражений.

Классификация операторов

Операторы в C# можно классифицировать по нескольким критериям:

По количеству операндов

Унарные: Оперируют одним операндом (например, -, !, ++, --).

Бинарные: Оперируют двумя операндами (например, +, -, *, /).

Тернарные: Оперируют тремя операндами (единственный тернарный оператор в C# - условный оператор ?:).

По типу операции

Арифметические: Выполняют математические операции (+, -, *, /, %, ++, --).

Сравнения: Сравнивают значения операндов (<, >, <=, >=, ==, !=).

Логические: Выполняют логические операции (&&, ||, !).

Битовые: Выполняют побитовые операции (|, &, ^, ~, <<, >>).

Присваивания: Присваивают значение переменной (=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=).

Другие: Операторы инкремента/декремента, условный оператор, операторы доступа к элементам массива и т.д.

2. Оператор присваивания.

Оператор присваивания (=) используется для присваивания значения переменной. Он принимает значение справа от оператора и сохраняет его в переменной, находящейся слева.

Пример:

```
int x = 10;

string name = "Alice";
```

В этом примере значение 10 присваивается переменной **x**, а строка "Alice" — переменной **name**.

Составные операторы присваивания объединяют операцию и присваивание в одном шаге. Это позволяет выполнить операцию над значением переменной и сразу же присвоить результат обратно этой переменной.

Основные составные операторы присваивания:

- +=: прибавление и присваивание.
- -=: вычитание и присваивание.
- *=: умножение и присваивание.
- /=: деление и присваивание.
- %=: взятие остатка и присваивание.

Примеры:

```
int x = 5;

// Сложение и присваивание
x += 3; // эквивалентно x = x + 3;  значение x теперь равно 8

// Вычитание и присваивание
x -= 2; // эквивалентно x = x - 2;  значение x теперь равно 6

// Умножение и присваивание
x *= 4; // эквивалентно x = x * 4;  значение x теперь равно 24

// Деление и присваивание
x /= 6; // эквивалентно x = x / 6;  значение x теперь равно 4

// Взятие остатка и присваивание
x %= 3; // эквивалентно x = x % 3;  значение x теперь равно 1
```

3. Арифметические операторы.

Арифметические операторы используются в C# для выполнения различных математических операций над числами.

Они позволяют складывать, вычитать, умножать, делить и выполнять другие числовые вычисления.

3. Арифметические операторы.

Арифметические операторы используются в C# для выполнения различных математических операций над числами.

Оператор	Описание	Пример
+	Сложение	int x = 5 + 3;
-	Вычитание	int y = 10 - 4;
*	Умножение	int z = 2 * 6;
/	Деление	double a = 15.0 / 3.0;
%	Остаток от деления	int b = 10 % 3; (результат будет 1)

Примеры:

```
int a = 5;
int b = 10;
int result = a + b; // result будет равно 15
```

```
int a = 10;
int b = 3;
int result = a / b; // result будет равно 3
```

```
int a = 10;
int b = 3;
int result = a % b; // result будет равно 1
```

Унарные операторы

- **++** (инкремент): Увеличивает значение переменной на 1.
- **--** (декремент): Уменьшает значение переменной на 1.
- **-** (унарный минус): Меняет знак числа.

```
int x = 5;
x++; // x теперь равно 6
int y = -x; // y теперь равно -6
```

Префиксная форма. Когда оператор стоит перед переменной, он называется префиксным. В этом случае значение переменной изменяется **перед** использованием ее в выражении.

```
int x = 5;
int y = ++x; // x становится 6, y также равно 6
```

Постфиксная форма. Когда оператор стоит после переменной, он называется постфиксным. В этом случае значение переменной изменяется **после** использования ее в выражении.

```
int x = 5;
int y = x++; // y равно 5, а затем x становится 6
```

Порядок выполнения арифметических операций определяется **приоритетом операторов**. Вот порядок приоритета от самого высокого к самому низкому:

- Унарные операторы: +, -, !, ~, ++, --
- Умножение, деление, остаток от деления: *, /, %
- Сложение, вычитание: +, -
- Сравнение: ==, !=, >, <, >=, <=
- Логические операторы: &&, ||
- Присваивание: =, +=, -=, *= и др.

4. Ввод-вывод данных. Класс Console.

Класс Console

Класс Console в C# предоставляет набор статических методов, позволяющих взаимодействовать с консольным окном.

Это ваш основной инструмент для вывода информации на экран и получения пользовательского ввода в консольных приложениях.

Основные методы класса Console:

WriteLine(string value): Выводит указанную строку в консоль и переводит курсор на новую строку.

Write(string value): Выводит указанную строку в консоль без перевода курсора на новую строку.

ReadLine(): Считывает строку с консоли.

ReadKey(): Считывает нажатую клавишу с консоли.

Clear(): Очищает содержимое консоли.

Beep(): Издает звуковой сигнал.

Подробнее:

https://learn.microsoft.com/ru-ru/dotnet/api/system.console?view=net-8.0

Некоторые методы для управления консольным окном и форматированием выводимого текста:

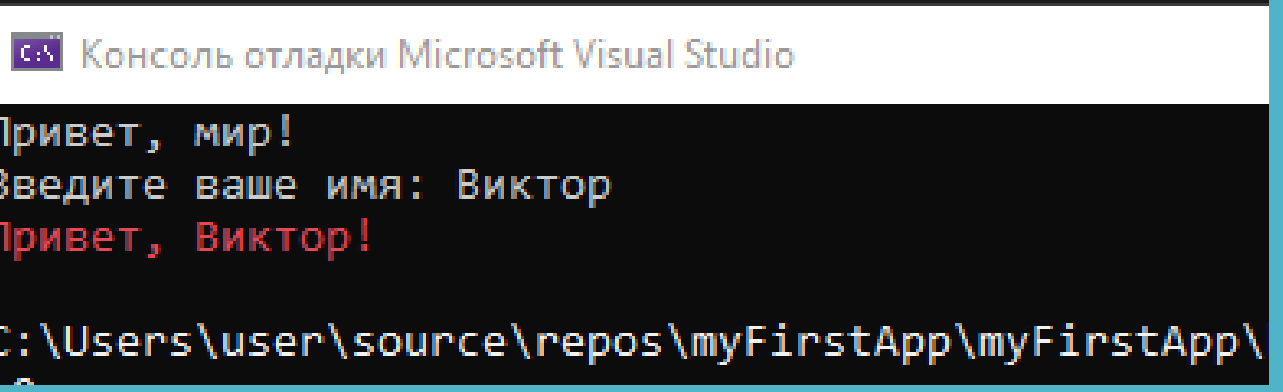
Управление цветом:

- **Console.ForegroundColor:** Устанавливает цвет текста.
- **Console.BackgroundColor:** Устанавливает цвет фона.
- **Console.ResetColor:** Сбрасывает цвета текста и фона на значения по умолчанию.

Управление позицией курсора:

- **Console.SetCursorPosition(int left, int top):** Устанавливает позицию курсора на заданные координаты.
- **Console.CursorLeft:** Получает или устанавливает позицию курсора по горизонтали.
- **Console.CursorTop:** Получает или устанавливает позицию курсора по вертикали.

```
Console.WriteLine("Привет, мир!"); // Вывод сообщения
Console.Write("Введите ваше имя: ");
string name = Console.ReadLine();  // чтение строки из консоли
Console.ForegroundColor = ConsoleColor.Red; // цвет шрифта - красный
Console.WriteLine($"Привет, {name}!");
Console.ForegroundColor = ConsoleColor.White;
```



Console.ReadLine() всегда возвращает строку, даже если пользователь ввел число.

Класс **Convert** предоставляет множество методов для преобразования между различными типами данных.

Convert.ToInt16 - в 16-битовое целое число со знаком.

Convert.ToInt32 - в 32-битовое целое число со знаком.

Convert.ToInt64 - в 64-битовое целое число со знаком.

Convert.ToSingle – в число с плавающей запятой (float)

Convert.ToDouble - в число с плавающей запятой двойной точности.

Convert.ToDecimal - в десятичное число.

https://learn.microsoft.com/ru-ru/dotnet/api/system.convert?view=net-8.0

```
Console.WriteLine("Ведите число a= ");
double a = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("Ведите число b= ");
double b = Convert.ToDouble(Console.ReadLine());

double result = a + b;
Console.WriteLine($"Результат: {a} + {b} = {result}");
```

Консоль отладки Microsoft Visual Studio

```
Ведите число a=
55,375
Ведите число b=
-12,456
Результат: 55,375 + -12,456 = 42,919
```

5. Класс Math.

Класс **Math** в C# предоставляет широкий набор статических методов для выполнения различных математических операций. Он является неотъемлемой частью стандартной библиотеки .NET Framework и используется для:

- Тригонометрических вычислений: синус, косинус, тангенс и их обратные функции.
- Экспоненциальных и логарифмических функций: возведение в степень, логарифмирование по различным основаниям.
- Округления чисел: округление вверх, вниз, до ближайшего целого числа.
- Вычисления корней и модулей: квадратный корень, абсолютное значение.
- Константы: числа Пи (Math.PI) и E (Math.E).

Основные методы класса Math:

- Тригонометрические: Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Tanh
- Экспоненциальные и логарифмические: Exp, Log, Log10, Pow
- Округление: Round, Floor, Ceiling, Truncate
- Корни и модули: Sqrt, Abs
- Минимальное и максимальное значение: Min, Max
- Случайные числа: Random

```
double x = 3.14;
double y = -2.5;

// Тригонометрические функции
double sinX = Math.Sin(x);
double cosY = Math.Cos(y);

// Экспоненциальные и логарифмические функции
double expX = Math.Exp(x);
double log10Y = Math.Log10(Math.Abs(y)); // Логарифм по основанию 10 от модуля y

// Округление
int roundedUp = (int)Math.Ceiling(x); // Округление вверх
int roundedDown = (int)Math.Floor(x); // Округление вниз
int rounded = (int)Math.Round(x, 2); // Округление до 2-х знаков

// Корень и модуль
double sqrtX = Math.Sqrt(x);
double absY = Math.Abs(y);
```

5. Класс Random.

Класс Random в C# является одним из основных инструментов для генерации псевдослучайных чисел. Он предоставляет различные методы для создания случайных чисел различных типов: целых, с плавающей запятой и т.д.

Для генерации случайных чисел необходимо создать новый экземпляр класса Random:

Random: Это имя класса, который мы используем.

random: Это имя переменной, в которую мы сохраним созданный объект.

new: Ключевое слово, которое используется для создания нового объекта.

Random(): Вызов конструктора класса Random.

Для генерации случайных чисел необходимо создать новый экземпляр класса Random:

```
Random rnd = new Random();
```

Random: Это имя класса, который мы используем.

rnd: Это имя переменной, в которую мы сохраним созданный объект.

new: Ключевое слово, которое используется для создания нового объекта.

Random(): Вызов конструктора класса Random.

Генерация случайных чисел:

Next(): Генерирует случайное целое число в диапазоне от 0 (включительно) до int.MaxValue (исключительно).

```
// Вывод: случайное целое число
int randomNumber = rnd.Next();
```

Next(int maxValue): Генерирует случайное целое число в диапазоне от 0 (включительно) до maxValue (исключительно).

```
// Случайное число от 0 до 9
int randomNumber2 = rnd.Next(10);
```

Next(int minValue, int maxValue): Генерирует случайное целое число в диапазоне от minValue (включительно) до maxValue (исключительно).

```
// Случайное число от 5 до 14
int randomNumber3 = rnd.Next(5, 15);
```

NextDouble(): Генерирует случайное число с плавающей точкой в диапазоне от 0.0 (включительно) до 1.0 (исключительно).

```
// Вывод: случайное число с плавающей точкой от 0.0 до 1.0
double randomDouble = rnd.NextDouble();
```

7. Класс Convert.

В C# класс Convert используется для преобразования типов данных из одного типа в другой.

Он содержит множество статических методов, позволяющих преобразовывать базовые типы данных, такие как числа, строки, логические значения и т.д.

Основные методы для преобразования в целочисленные типы

ToByte(): в тип byte (8-битное целое число без знака).

ToUInt16(): в тип ushort (16-битное целое число без знака).

ToUInt32(): в тип uint (32-битное целое число без знака).

ToUInt64(): в тип ulong (64-битное целое число без знака).

ToSByte(): в тип sbyte (8-битное целое число со знаком).

ToInt16(): в тип short (16-битное целое число со знаком).

ToInt32(): в тип int (32-битное целое число со знаком).

ToInt64(): в тип long (64-битное целое число со знаком).

Основные методы для преобразования в числа с плавающей точкой

ToSingle(): Преобразует значение в тип float (32-битное число с плавающей точкой одинарной точности).

ToDouble(): Преобразует значение в тип double (64-битное число с плавающей точкой двойной точности).

ToDecimal(): Преобразует значение в тип decimal (128-битное число с высокой точностью, обычно используется для финансовых расчетов).

Пример: Преобразование строки в целое число (int)

```
Console.Write("Введите число: ");
string input = Console.ReadLine();  // Чтение строки с консоли
int number = Convert.ToInt32(input); // Преобразование строки в целое число


Console.WriteLine($"Вы ввели число: {number}");
```

Пример: Преобразование строки в число с плавающей точкой (double)

```
Console.Write("Введите дробное число: ");
string input = Console.ReadLine();
double number = Convert.ToDouble(input); // Преобразование строки в double


Console.WriteLine($"Вы ввели число: {number}");
```

Некоторые другие методы.

ToBoolean(): Преобразует значение в логическое значение (bool).

```
string boolString = "true";
bool flag = Convert.ToBoolean(boolString);
```

ToString(): Преобразует значение любого типа в строку (string).

```
int number = 123;
string numberString = Convert.ToString(number);
```

ToChar(): Преобразует значение в символ (char).

```
int charCode = 65;
char letter = Convert.ToChar(charCode);  // Результат: 'A'
```

ToDateTime(): Преобразует значение в объект DateTime.

```
string dateString = "2024-09-01";
DateTime date = Convert.ToDateTime(dateString);
```

Контрольные вопросы:

- Какие типы операторов существуют в языке C#?
- Что делает оператор присваивания в C#? Приведите пример.
- Какие арифметические операторы поддерживаются в C#?
- Объясните разницу между префиксной и постфиксной формами инкремента.
- Как работает класс Console для ввода-вывода данных? Назовите основные методы.
- Какие функции предоставляет класс Math в C#? Приведите примеры.
- Как создать экземпляр класса Random и для чего он используется?
- Что делает метод Console.ReadLine() и как преобразовать его результат в число?
- Какие методы класса Random используются для генерации случайных чисел?
- Назовите основные методы управления консольным окном с помощью класса Console.

Домашнее задание:

1. Повторить материал лекции.
2. Учебное пособие, с. 18..24
3. Решить задачи, с 25

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com

Список литературы:

1. Жаксыбаева Н.Н. Основы объектно-ориентированного программирования: язык C#. Часть 1./ Учебное пособие предназначено для учащихся технического и профессионального образования, Алматы, 2010,

2. https://learn.microsoft.com/ru-ru/dotnet/csharp/