**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

«Изучение основных конструкций языка C#»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Хабленко Инна |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

# Введение

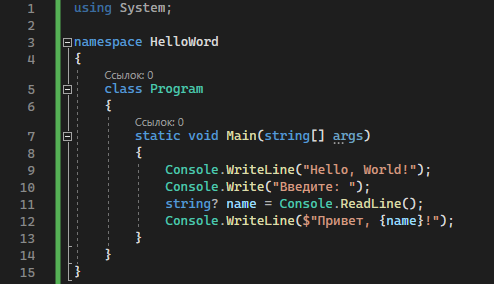
C# – объектно-ориентированный язык программирования, разработанный компанией Microsoft. Создан в 2000 году по инициативе Anders Hejlsberg. Язык стал ключевым элементом платформы .NET Framework и был ориентирован на облегчение разработки приложений для Windows. В настоящее время C# широко используется в различных областях, таких как разработка веб-приложений, десктоп-приложений, игр, мобильных приложений и многих других. Его популярность связана с широкими возможностями, безопасностью и отличной производительностью.

# Основы языка

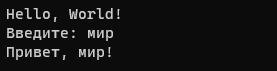
Ввод и вывод в C# осуществляется через стандартные потоки ввода и вывода (System.Console). Для этого используются следующие методы:

– Console.ReadLine() - считывает строку текста с клавиатуры.

– Console.WriteLine() - выводит строку текста на экран.

Пример использования этих методов для ввода и вывода данных:

Результат:



Типы данных:

1. Целочисленные типы данных:

- int: 32-битное целое число

- long: 64-битное целое число

- short: 16-битное целое число

- byte: 8-битное целое число

1. Вещественные типы данных:

- float: 32-битное вещественное число с плавающей запятой

- double: 64-битное вещественное число с плавающей запятой

- decimal: 128-битное десятичное число

1. Логический тип данных:

- bool: Логический тип, который представляет собой значение true или false.

1. Символьный тип данных:

- char: 16-битный символ Unicode.

1. Строковый тип данных:

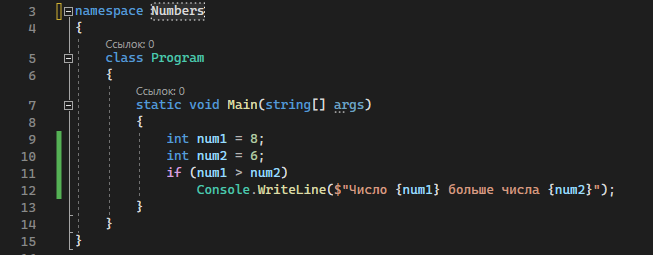
- string: последовательность символов Unicode.

1. Другие типы данных:

- объекты, массивы, перечисления, структуры и т. д.

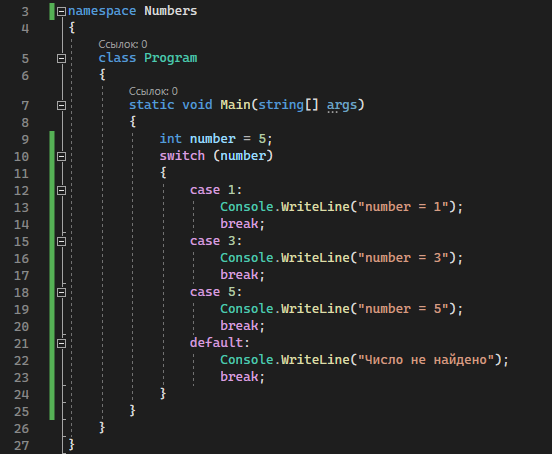
# Управляющие структуры

Оператор if…else используется для выполнения разных блоков кода в зависимости от условия. Пример использования if…else:



Результат:  


Оператор switch используется для выбора одного из нескольких блоков кода на основе значения некоторой переменной или выражения.



Результат:



В C# существуют различные виды циклов для обработки данных, вот некоторые из них:

**Цикл for:** Используется для выполнения блока кода определенное количество раз.

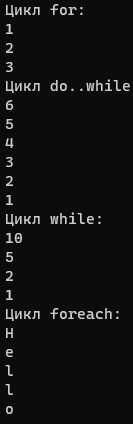
**Цикл do-while:** Выполняет блок кода хотя бы один раз, даже если условие изначально ложно.

**Цикл while:** Выполняет блок кода пока условие истинно.

**Цикл foreach:** Используется для перебора элементов в коллекции.



Результат:



# Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) является одним из основных принципов разработки на C#. В этом стиле программирования программа представляется в виде объектов, каждый из которых имеет свои свойства и методы. Объекты могут взаимодействовать друг с другом, вызывая методы и получая значения свойств.

Основные принципы ООП в C#:

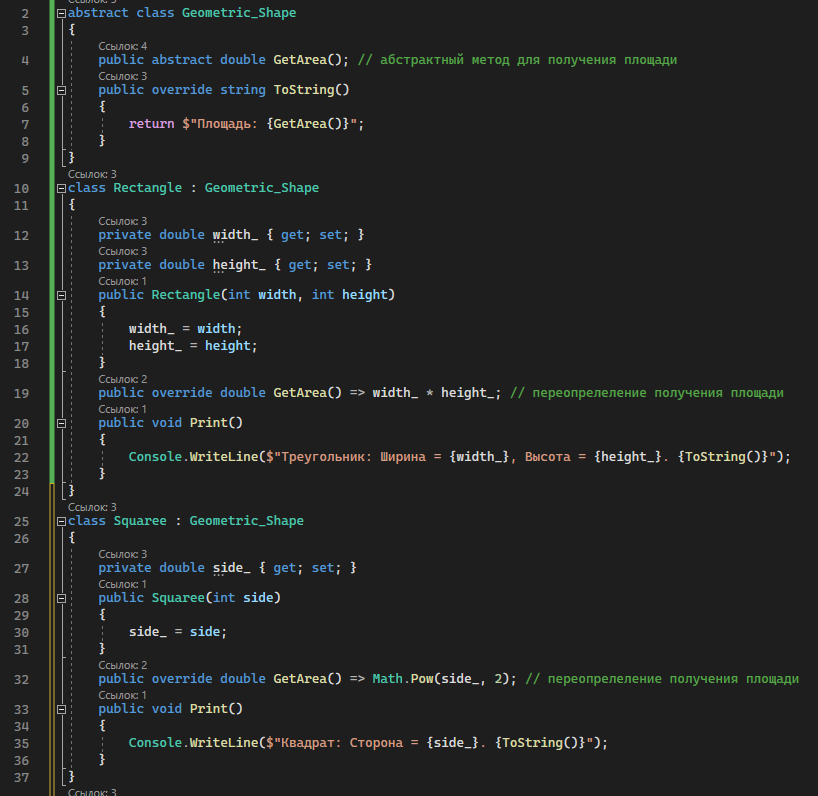
* **Инкапсуляция**: сокрытие деталей реализации объекта и предоставление только интерфейса для работы с ним.
* **Наследование**: возможность создания новых классов на основе существующих с добавлением или изменением функциональности.
* **Полиморфизм:** возможность использовать один интерфейс для выполнения различных действий в зависимости от типа объекта.

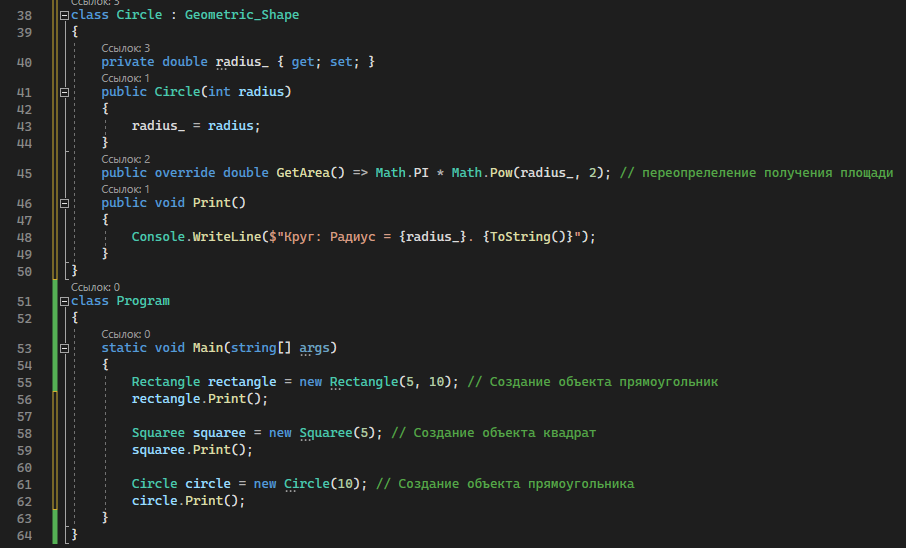
Для создания объектов в C# используются классы, которые определяют свойства и методы объекта. Каждый объект является экземпляром класса и имеет свои уникальные свойства и состояние.

Одним из ключевых понятий в ООП является инкапсуляция. Она позволяет скрыть детали реализации объекта и предоставить только интерфейс для работы с ним. Это делает код более понятным и облегчает его поддержку.

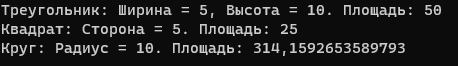
Наследование позволяет создавать новые классы на основе существующих, добавляя или изменяя функциональность. Это позволяет сократить количество кода и упростить его понимание.

Полиморфизм позволяет использовать один и тот же интерфейс для выполнения разных действий в зависимости от типа объекта.



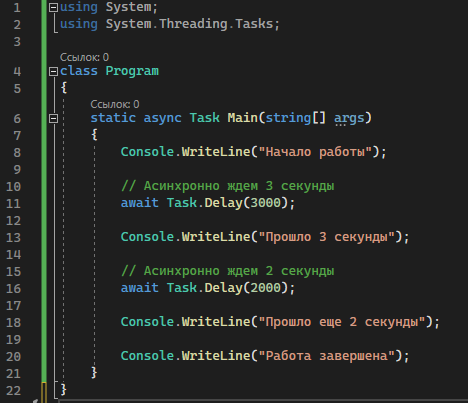


Результат:

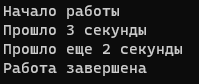


# Асинхронное программирование

В этом примере метод `Main` объявлен как асинхронный с помощью ключевого слова `async`, и он использует `await` для вызова асинхронных операций, таких как `Task.Delay`. Когда программа достигает `await`, она приостанавливает выполнение `Main` и продолжает работу других задач. Когда асинхронная операция завершается, выполнение `Main` возобновляется сразу после операции `await`.



Результат:



# Пример программы

using System;

class TemperatureConverter

{

public static double CelsiusToFahrenheit(double celsius)

{

return (celsius \* 9 / 5) + 32;

}

public static double FahrenheitToCelsius(double fahrenheit)

{

return (fahrenheit - 32) \* 5 / 9;

}

public static double CelsiusToKelvin(double celsius)

{

return celsius + 273.15;

}

public static double KelvinToCelsius(double kelvin)

{

return kelvin - 273.15;

}

public static void Print\_Fahrenheit(double fahrenheit)

{

Console.WriteLine($"Температура в Фаренгейтах: {fahrenheit} °F");

}

public static void Print\_Celsius(double celsius)

{

Console.WriteLine($"Температура в Цельсиях: {celsius} °C");

}

public static void Print\_Kelvin(double kelvin)

{

Console.WriteLine($"Температура в Кельвинах: {kelvin} К");

}

}

class Program

{

static void Main()

{

while (true)

{

Console.WriteLine("Выберите опцию:");

Console.WriteLine("1. Конвертировать из Цельсия в Фаренгейт");

Console.WriteLine("2. Конвертировать из Фаренгейта в Цельсия");

Console.WriteLine("3. Конвертировать из Цельсия в Кельвины");

Console.WriteLine("4. Конвертировать из Кельвинов в Цельсия");

Console.WriteLine("5. Выход");

int option = Convert.ToInt32(Console.ReadLine());

if (option == 1)

{

Console.WriteLine("Введите температуру в Цельсиях:");

double celsius = Convert.ToDouble(Console.ReadLine());

double fahrenheit = TemperatureConverter.CelsiusToFahrenheit(celsius);

TemperatureConverter.Print\_Fahrenheit(fahrenheit);

}

else if (option == 2)

{

Console.WriteLine("Введите температуру в Фаренгейтах:");

double fahrenheit = Convert.ToDouble(Console.ReadLine());

double celsius = TemperatureConverter.FahrenheitToCelsius(fahrenheit);

TemperatureConverter.Print\_Celsius(celsius);

}

else if (option == 3)

{

Console.WriteLine("Введите температуру в Цельсиях:");

double celsius = Convert.ToDouble(Console.ReadLine());

double kelvin = TemperatureConverter.CelsiusToKelvin(celsius);

TemperatureConverter.Print\_Kelvin(kelvin);

}

else if (option == 4)

{

Console.WriteLine("Введите температуру в Кельвинах:");

double kelvin = Convert.ToDouble(Console.ReadLine());

double celsius = TemperatureConverter.KelvinToCelsius(kelvin);

TemperatureConverter.Print\_Celsius(celsius);

}

else if (option == 5)

{

break;

}

else

{

Console.WriteLine("Ошибка: Неправильная опция");

}

}

}

}

Результат:

