Турсунов Баходурхон

#10. .NET Object-Oriented Programming (OOP). OOPs concept. What is the Namespace. Records. Struct.

1. **Дайте определение термину "пространство имен" в контексте языка C#. Объясните его назначение и то, как оно помогает организовать элементы кода.**   
   Пространство имен (namespace) в C# представляет собой механизм для организации кода и предотвращения конфликтов имен между различными частями программы. Пространства имен позволяют группировать связанные классы, интерфейсы, делегаты и другие типы данных внутри определенного контекста.
2. **Опишите концепцию объявлений пространств имен с копированием файлов в C# 10. Чем они отличаются от обычных пространств имен?**

Раньше при объявлении пространства имен все содержимое этого пространства имен должно было помещаться в фигурные скобки. В C#10 мы можем объявить пространство имен перед всеми другими объявлениями и не ставить после него фигурные скобки. В результате пространство имен будет автоматически включать все определения, которые встречаются в файле

1. **Объясните значение неизменяемости в типах записей. Приведите примеры сценариев, в которых неизменяемость является критически важной.**

Неизменяемость (immutability) в типах записей означает, что после создания экземпляра типа записи его состояние не может быть изменено. В C# 9.0 и более поздних версиях, типы записей (record types) поддерживают неизменяемость по умолчанию, что означает, что их свойства (поля) не могут быть изменены после создания экземпляра.

Пример объявления типа записи с неизменяемыми свойствами:

public record Person

{

public string FirstName { get; }

public string LastName { get; }

public Person(string firstName, string lastName)

{

FirstName = firstName;

LastName = lastName;

}

}

Преимущества неизменяемости в типах записей включают:

Безопасность потоков: Неизменяемые объекты более предсказуемы и безопасны в многопоточных сценариях, так как они не могут быть изменены после создания.

Простота отладки: когда объект неизменяем, можно более легко отслеживать его состояние, так как оно не изменяется в течение времени жизни объекта.

Устойчивость к ошибкам: коскольку неизменяемые объекты нельзя изменить, это снижает вероятность ошибок, связанных с неожиданным изменением состояния объекта.

1. **Дайте определение и различие между понятиями "класс записи" и "структура записи" в C# 10. Приведите примеры кода для обоих типов.**

Класс записи (Record Class):

Класс записи в C# 10 — это ссылочный тип, который автоматически предоставляет реализацию структуры, сравнения значений и других методов для работы с данными. Он поддерживает неизменяемость и упрощает создание неизменяемых объектов данных.

Пример кода для класса записи:

public record class Person

{

public string FirstName { get; init; }

public string LastName { get; init; }

}

class Program

{

static void Main()

{

var person1 = new Person { FirstName = "John", LastName = "Doe" };

var person2 = new Person { FirstName = "John", LastName = "Doe" };

// Автоматически сгенерированный метод Equals для сравнения значений

Console.WriteLine(person1.Equals(person2)); // Выведет True

}

}

Структура записи (Record Struct):

Структура записи также представляет из себя автоматически генерируемый код для работы с данными, но в отличие от класса записи, она является значимым типом (структурой). Структуры записи полезны, когда требуется неизменяемость и эффективность работы с памятью.

Пример кода для структуры записи:

public record struct Point

{

public int X { get; init; }

public int Y { get; init; }

}

class Program

{

static void Main()

{

var point1 = new Point { X = 1, Y = 2 };

var point2 = new Point { X = 1, Y = 2 };

// Автоматически сгенерированный метод Equals для сравнения значений

Console.WriteLine(point1.Equals(point2)); // Выведет True

}

}

Оба примера демонстрируют, как использовать классы и структуры записи для определения неизменяемых объектов данных. Они автоматически генерируют методы для сравнения значений, что делает их удобными для использования в сценариях, где важна устойчивость к изменениям данных.

1. **Какие типы структур существуют в C#? Объясните фундаментальные характеристики, которые отличают их от типов классов.**

В C# существует два основных типа данных: классы (class) и структуры (struct). Вот основные характеристики, которые отличают структуры от классов:

Тип хранения данных:

Классы: Классы являются ссылочными типами, что означает, что переменные, содержащие объекты класса, хранят ссылки на данные в куче (heap). Для сравнения объектов класса используется сравнение ссылок.

Структуры: Структуры являются значимыми типами, и их значения хранятся непосредственно в стеке или внедряются в контейнер, содержащийся в другом объекте. Сравнение значений структур происходит напрямую, без учета ссылок.

Производительность:

Классы: Использование классов может быть менее эффективным с точки зрения производительности из-за выделения и освобождения памяти в куче, а также из-за управления сборкой мусора.

Структуры: Структуры, будучи значимыми типами, могут быть более эффективными в использовании памяти и обеспечивать лучшую производительность для небольших объектов.

Наследование:

Классы: Поддерживают наследование, что означает, что один класс может наследовать функциональность другого класса.

Структуры: Не поддерживают наследование. Они не могут наследовать от других структур или классов, и также не могут быть базовыми классами.

Использование в качестве аргументов методов:

Классы: Передаются по значению, но это значение — ссылка на объект. Изменения объекта, переданного в метод, будут отражены в вызывающем коде.

Структуры: Передаются по значению. Если вы передаете структуру в метод, то метод работает с копией этой структуры, и изменения не затрагивают оригинал.

Инициализация:

Классы: Требуют явного использования ключевого слова new для создания экземпляра.

Структуры: Могут быть инициализированы без использования new, а также поддерживают синтаксис инициализации полей.

Примечание: Существует также категория значимых типов, называемая перечислениями (enum), которые представляют набор именованных константных значений и также являются значимыми типами, подобными структурам.

1. **Обсудите концепцию передачи по значению и передачи по ссылке при работе с типами struct. Как можно избежать копирования значений для больших структурных типов?**

Передача по значению и передача по ссылке для типов struct в C#:

В языке C#, значения переменных могут передаваться в методы либо по значению, либо по ссылке. Это имеет важное значение, особенно когда речь идет о типах значений, таких как struct. Рассмотрим более подробно оба подхода.

Передача по значению (pass by value):

В случае передачи аргумента по значению, метод получает копию значения аргумента. Для структур, таких как struct, это означает, что создается копия структуры, и изменения внутри метода не затрагивают оригинальную структуру. Это поведение по умолчанию для структур в C#.

Пример:

public struct Point

{

public int X;

public int Y;

}

public void ModifyPoint(Point point)

{

point.X = 10; // Изменение копии внутри метода

}

// Использование

Point myPoint = new Point();

ModifyPoint(myPoint);

// myPoint.X все еще равен 0, так как структура передавалась по значению

Передача по ссылке (pass by reference):

Для структур также можно использовать ключевое слово ref для передачи аргумента по ссылке. Это означает, что метод получает не копию структуры, а ссылку на оригинал. Изменения внутри метода влияют на оригинальную структуру.

Пример:

public void ModifyPoint(ref Point point)

{

point.X = 10; // Изменение оригинала, так как передача по ссылке

}

// Использование

Point myPoint = new Point();

ModifyPoint(ref myPoint);

// Теперь myPoint.X равен 10, так как структура передавалась по ссылке

Practice:

1. Создайте основной namespace

2. Внутри данного namespace создайте следующие классы:

3. Класс, представляющий сотрудника. Содержит основную информацию о сотруднике, такую как имя, фамилия, должность и заработная плата. Используйте record для представления сотрудника.

4. Класс, представляющий отдел в компании. Включает в себя список сотрудников, работающих в этом отделе.

5. Класс, представляющий саму компанию. Содержит список отделов и базовую информацию о компании.

6. Создайте несколько сотрудников, отделов и компаний, используя созданные классы.

7. Добавьте функциональность для вывода информации о сотрудниках, отделах и компании в консоль.

```

namespace Company

{

internal class Program

{

public record Employee // класс представляющий сотрудника

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Position { get; set; } // занимаемая должность

public decimal Salary { get; set; } // зарплата

}

public class Department // класс представляющий отдел в компании

{

public string DepartmentName { get; set; } // название отдела

public Employee[] employee { get; set; } // список сотрудников работающих в этой компании

}

public class Company // класс представляющий саму компанию

{

public string CompanyName { get; set; }

public Department[] Departmentlist { get; set; } // списки отделов

public string CompanyDescription { get; set; }

}

public static void DisplayInfo()

{

Employee employee1 = new()

{

FirstName = "Баходур",

LastName = "Турсунов",

Position = "Backend developer",

Salary = 10000

};

Employee employee2 = new()

{

FirstName = "Ислам",

LastName = "Исмаилов",

Position = "Full stack developer",

Salary = 20000

};

Employee[] employees =

{

employee1,

employee2

};

Department department1 = new()

{

DepartmentName = "IT",

employee = employees

};

Department[] departments =

{

department1

};

Company FirstCompany = new()

{

CompanyName = "Alif",

Departmentlist = departments,

CompanyDescription = "Алиф Бонк, англ. Alif Bank; ранее — Алиф Капитал, Алиф Сармоя) — финансово-технологическая компания, основанная в 2014 году и преобразованная в банк в 2020 году. Алиф использует принципы исламского финансирования. Офисы расположены в Душанбе и Худжанде."

};

Company SecondCompany = new()

{

CompanyName = "Eskhata",

Departmentlist = departments,

CompanyDescription = "Банк Эсхата — таджикский коммерческий банк, один из крупнейших коммерческих Банков Таджикистана, который входит в пятерку лидеров в банковской сфере страны. Полное наименование — Открытое акционерное общество «Банк Эсхата». Кроме того, в деловой практике зачастую используется сокращение"

};

Console.WriteLine($"Company: {FirstCompany.CompanyName}");

foreach (var department in FirstCompany.Departmentlist)

{

Console.WriteLine($" Department: {department1.DepartmentName}");

foreach (var employee in department1.employee)

{

Console.WriteLine(

$" Employee: {employee.FirstName} {employee.LastName}, {employee.Position}, Salary: {employee.Salary}$");

}

}

Console.WriteLine($"Company: {SecondCompany.CompanyName}");

foreach (var department in FirstCompany.Departmentlist)

{

Console.WriteLine($" Department: {department1.DepartmentName}");

foreach (var employee in department1.employee)

{

Console.WriteLine(

$" Employee: {employee.FirstName} {employee.LastName}, {employee.Position}, Salary: {employee.Salary}$");

}

}

}

static void Main(string[] args)

{

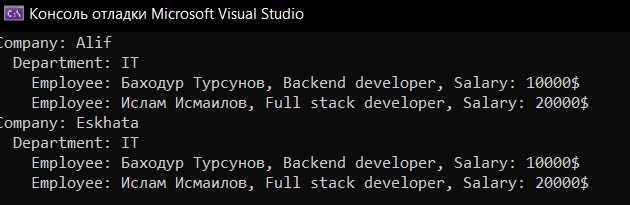
DisplayInfo();

}

}

}

```



Дополнительное задание:

1. Проведите исследование того, как работает проверка на равенство в типах record? Опишите свое исследование и то, что вы узнали. Приведите пример кода.

```

public record Person

{

public string Name { get; set; }

public Person(string name) => Name = name;

}

static void Main(string[] args)

{

var person1 = new Person("Tom");

var person2 = new Person("Tom");

Console.WriteLine(person1.Equals(person2));

}

Здесь вывод будет true, потому что метод Equals сравнивает значение двух record-ов

```

1. Изучите что такое ref struct, какие имеет нюансы в использовании, где проходит его аллокация.,

Экземпляры ref struct типа выделяются в стеке и не могут экранироваться в управляемую кучу. Компилятор ограничивает использование таких типов следующим образом:

* Не может быть типом элемента массива
* Не может быть объявленным типом поля класса
* Не может реализовать интерфейсы
* Не удастся упаковать в System.ValueType или System.Object
* Переменная не может быть захвачена лямбда выражением или локальной функцией

Объявляются они следующим образом:

public ref struct CustomRef

{

public bool IsValid;

public Span<int> Inputs;

public Span<int> Outputs;

}