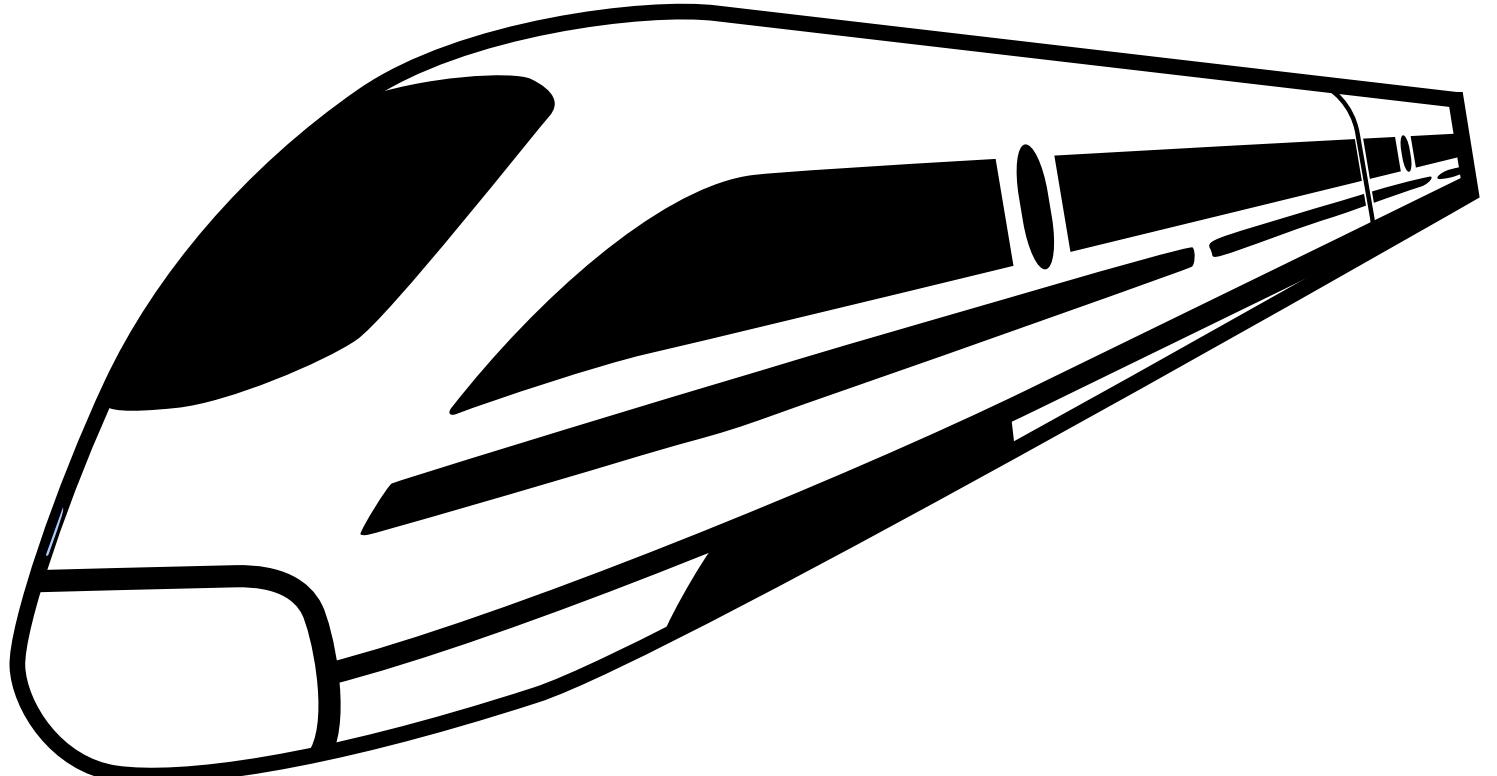
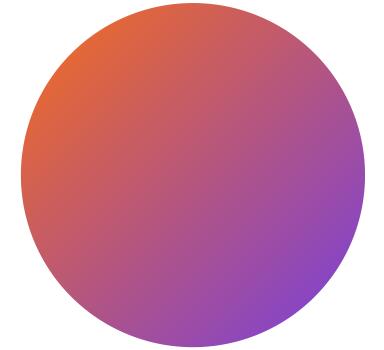




ASICODE

Курс Основи - C#

ЗАНЯТТЯ № 9 - КОЛЕКЦІЇ ТА STRUCT



Колекція VS масив

Найпримітивнішим способом зберігання об'єктів C# є використання масивів. Однією з основних проблем, з якою зіткнеться розробник за таким підходом, є те, що масиви не надають інструментів для динамічної зміни розміру. У мові C# є два простори імен для роботи зі структурами даних:

- System.Collections;
- System.Collections.Generic.

Перше з них – System.Collections надає структури даних для зберігання об'єктів типу Object . Це рішення має дві основні проблеми – це продуктивність і безпека типів.

Для вирішення зазначених вище проблем Microsoft були розроблені колекції з узагальненими типами (іх ще називають дженерики), вони розташовані у просторі імен System.Collections.Generic . Суть їх полягає в тому, що ви не просто створюєте об'єкт класу List , але і вказуєте, об'єкти якого типу будуть в ньому зберігатися, робиться це так: List<T> , де T може бути int, string, double або якийсь ваш власний клас.



Види колекцій

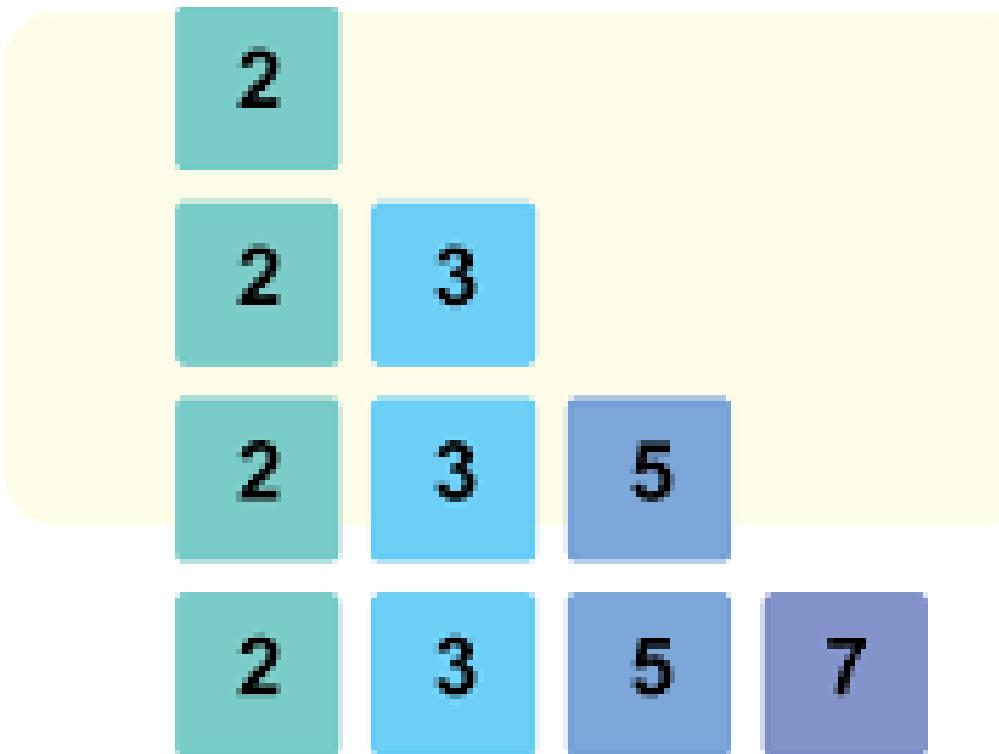
Простір `System.Collections.Generic` містить великий набір колекцій, які дозволяють зручно та ефективно вирішувати широке коло завдань. Нижче в таблиці перераховані деякі з узагальнених класів із зазначенням інтерфейсів, які вони реалізують.

Узагальнений клас	Основні інтерфейси	Опис
<code>List<T></code>	<code>ICollection<T>, IEnumerable<T>, IList<T></code>	Список елементів з розміром, що динамічно змінюється
<code>Dictionary<TKey, TValue></code>	<code>ICollection<T>, IDictionary<TKey, TValue>, IEnumerable<T></code>	Колекція елементів, пов'язаних через унікальний ключ
<code>Queue<T></code>	<code>ICollection, IEnumerable<T></code>	Черга – список, що працює за алгоритмом <i>FIFO</i>
<code>Stack<T></code>	<code>ICollection, IEnumerable<T></code>	Стек – список, що працює за алгоритмом <i>LIFO</i>
<code>SortedList<TKey, TValue></code>	<code>IComparer<T>, ICollection<KeyValuePair<TKey, TValue>>, IDictionary<TKey, TValue></code>	Колекція пар "ключ-значення", впорядкованих за ключем



`new List<int>()`

Add



Add

Add

Add

Клас List<T>

Почнемо наше знайомство з колекціями з класу List<T>. Ця колекція є аналогом типізованого масиву, який динамічно розширюватиметься. Як тип можна вказати будь-який вбудований або користувальницький тип.

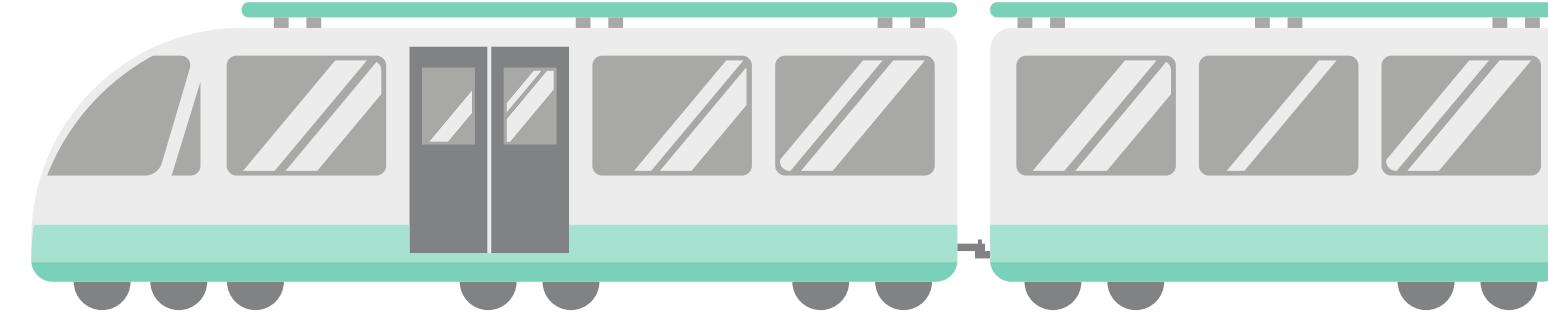
Створення об'єкта класу List<T>

Можна створити порожній список і додати елементи пізніше, за допомогою методу Add() :

```
List < int > numsList = new List < int >();  
numsList. Add ( 1 );
```

Або скористатися синтаксисом, що дозволяє вказати набір об'єктів, який зберігатиметься у списку:

```
List < int > nums = new List < int > { 1 , 2 , 3 , 4 , 5 } ;  
var words = new List < string > { "one" , "two" , "three" } ;
```



Властивості класу List<T>

Нижче наведено деякі корисні властивості та методи класу List<T> .

Властивості класу List<T>

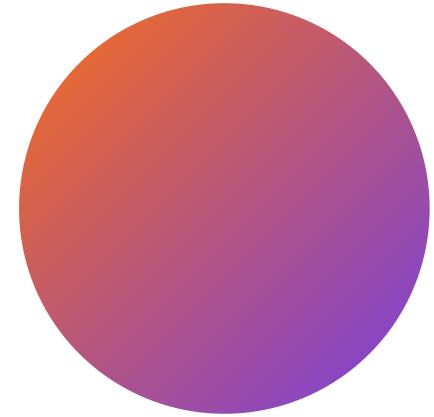
Count - Кількість елементів у списку

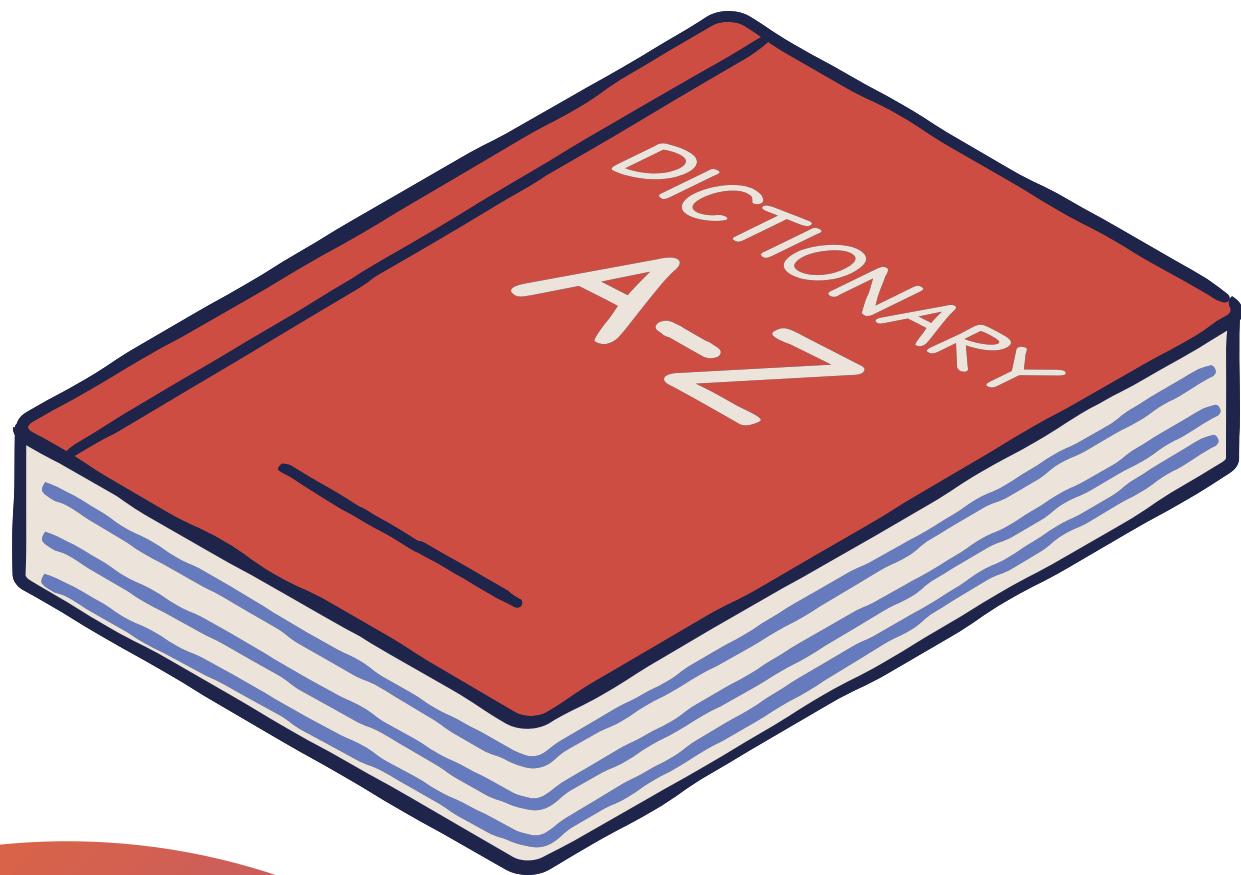
Capacity - Місткість списку – кількість елементів, яка може вмістити список без зміни розміру



Методи класу List<T>

Метод	Опис
<i>Add(T)</i>	Додає елемент до списку
<i>BinarySearch(T)</i>	Виконує пошук за списком
<i>Clear()</i>	Очищення списку
<i>Contains(T)</i>	Повертає <i>true</i> , якщо список містить вказаний елемент
<i>IndexOf(T)</i>	Повертає індекс переданого елемента
<i>ForEach(Action<T>)</i>	Виконує вказану дію для всіх елементів списку
<i>Insert(Int32, T)</i>	Вставляє елемент у вказану позицію
<i>Find(Predicate<T>)</i>	Здійснює пошук першого елемента, для якого виконується заданий предикат
<i>Remove(T)</i>	Видаляє вказаний елемент зі списку
<i>RemoveAt(Int32)</i>	Видаляє елемент із заданої позиції
<i>Sort()</i>	Сортує список
<i>Reverse()</i>	Змінює порядок розташування елементів на протилежний





Клас Dictionary< TKey, TValue >

Клас Dictionary реалізує структуру даних , яку іноді називають Словник або Асоціативний масив. Ідея досить проста: у звичайному масиві доступ до даних ми отримуємо через цілий індекс, у словнику використовується ключ, який може бути числом, рядком або будь-яким іншим типом даних, який реалізує метод GetHashCode() . При додаванні нового об'єкта в таку колекцію вказується унікальний ключ, який використовується для подальшого доступу до нього.

Створення об'єкту класу Dictionary

Порожній словник:

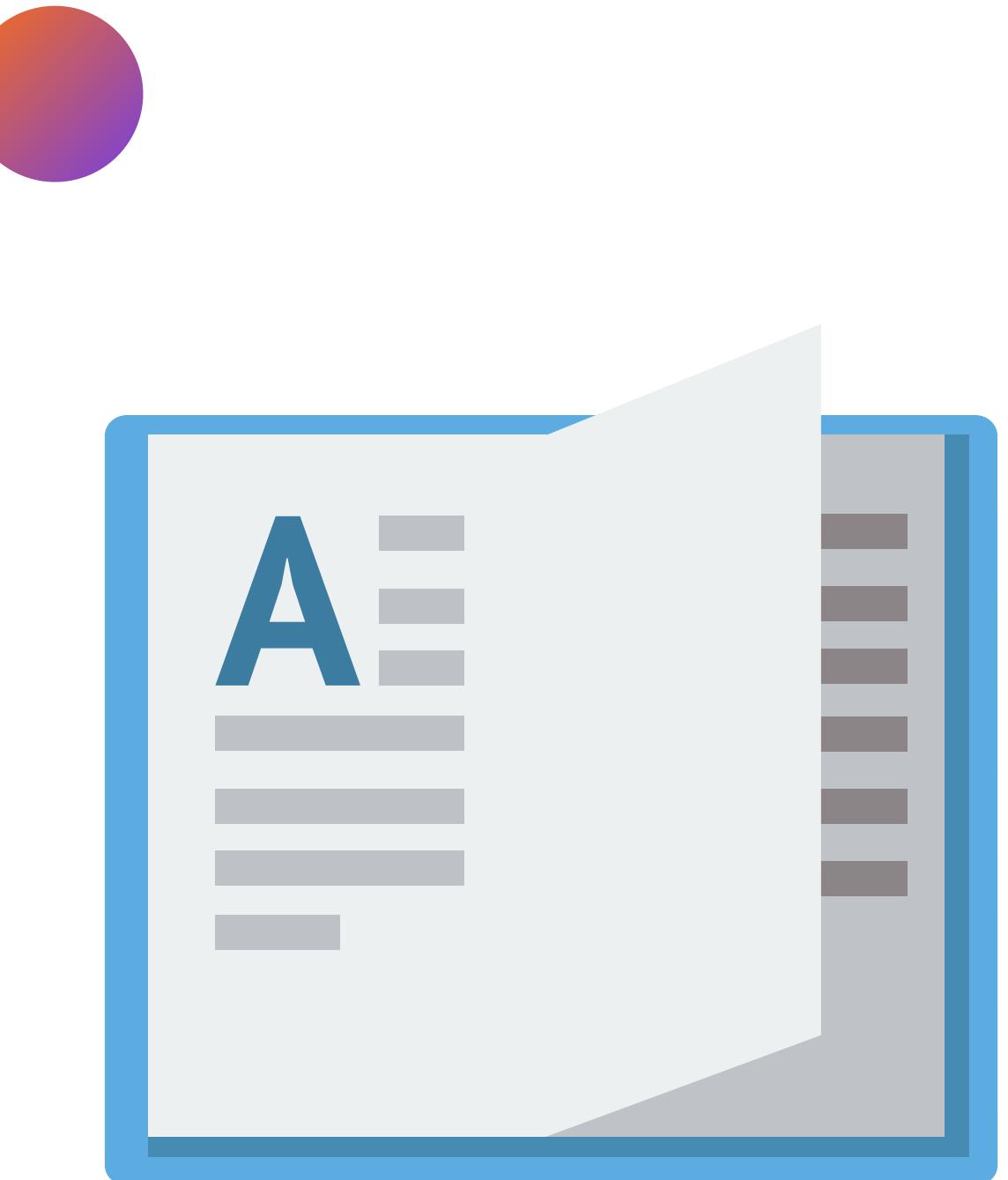
```
var dict = new Dictionary< string, int >();
```

Словник з набором елементів:

```
var prodPrice = новий Dictionary< string, double >()
{
    [ "bread" ] = 23.3 ,
    [ "apple" ] = 45.2
};
Console.WriteLine ( $ "bread price: {prodPrice[ " bread " ]}" );
```

Властивості класу Dictionary

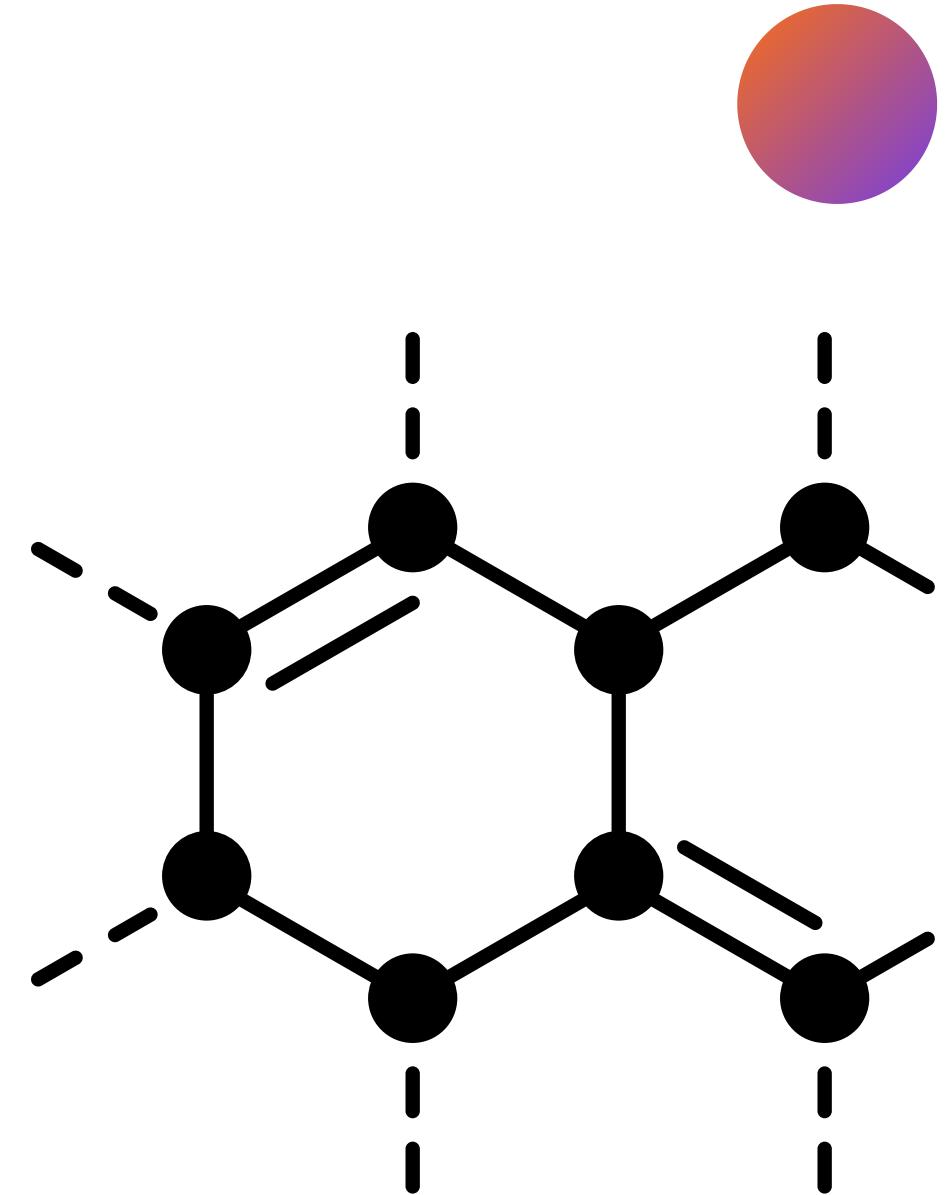
Властивість	Опис
<i>Count</i>	Кількість об'єктів у словнику
<i>Keys</i>	Ключі словника
<i>Values</i>	Значення елементів словника





Методи класу Dictionary

Метод	Опис
<i>Add(TKey, TValue)</i>	Додає до словника елемент із заданим ключем та значенням
<i>Clear()</i>	Видаляє зі словника всі ключі та значення
<i>ContainsValue(TValue)</i>	Перевіряє наявність у словнику вказаного значення
<i>ContainsKey(TKey)</i>	Перевіряє наявність у словнику вказаного ключа
<i>GetEnumerator()</i>	Повертає перечислювач для перебору елементів словника
<i>Remove(TKey)</i>	Видаляє елемент із зазначеним ключем
<i>TryAdd(TKey, TValue)</i>	Метод, що реалізує спробу додати до словника елемент із заданим ключем та значенням
<i>TryGetValue(TKey, TValue)</i>	Метод, що реалізує спробу отримати значення за заданим ключем



Що таке структура

Структура (struct) в C# — це тип даних користувача , який використовується поряд з класами і може містити будь-які дані і методи. Структурами є такі типи даних як `int`, `double` т.д. Основна відмінність структури (struct) від класу (class) у тому, що структура — це тип значень , а клас — це тип посилань.



ASICODE

оголошення структури

Для использования структуры ее необходимо инициализировать. Для инициализации создания объектов структуры, как и в случае с классами, применяется вызов конструктура с оператором new. Даже если в коде структуры не определено ни одного конструктора, тем не менее имеет как минимум один конструктор - конструктор по умолчанию, который генерируется компилятором. Этот конструктор не принимает параметров и создает объект структуры со значениями по умолчанию.

new назва_структурі();





Відмінність структури від класу C#



Структура - тип значень, клас - тип посилання
Якщо не вдаватися далеко в подробиці роботи програм, то основна відмінність полягає в тому, що структура зберігатиметься цілком у стеку, а об'єкт класу зберігатиметься у купі, а посилання на нього - у стеку struct. classB результаті цього, доступ до цих структур буде шлях не набагато, але швидше, ніж до класу.

Структури не підтримують успадкування

Коли використовувати структури (struct), а коли класи (class) у C#
Звичайно, питання про те, що краще використовувати залежить, в першу чергу, від того, в контексті чого задається таке питання, але основна рекомендація від Microsoft може бути сформульована наступним чином: структури (struct) варто використовувати в тому випадку, якщо ваш об'єкт містить мінімальне кількість будь-яких логічно пов'язаних операцій або не містить їх взагалі.



ASICODE

Дякую за увагу