*Увлекательный C# - особенности языка программирования*

*(коллекции языка)*

*Стич Назар ИВТ-22 Объектно Ориентированное Программирование*

[*https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/set*](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/set) *- set*

[*https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector?view=net-8.0*](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector?view=net-8.0) *– vector*

[*https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/arrays*](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/arrays) *- array*

[*https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples*](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples) *- tuple*

[*https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/arrays-and-collections*](https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/arrays-and-collections) *list*

# Set (Справочник по C#)

Ключевое слово set определяет метод доступа в свойстве или индексаторе, который присваивает значение свойству элемента индексатора. Дополнительные сведения и примеры см. в разделах [Свойства](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/properties), [Автоматически реализуемые свойства](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/auto-implemented-properties) и [Индексаторы](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/indexers/).

В приведенном ниже примере определен как метод доступа get, так и метод доступа set для свойства с именем Seconds. Для возвращения значения свойства в нем используется закрытое поле с именем \_seconds.

C#Копировать

class TimePeriod

{

private double \_seconds;

public double Seconds

{

get { return \_seconds; }

set { \_seconds = value; }

}

}

Метод доступа set часто состоит из одного оператора, который присваивает значение, как в предыдущем примере. Метод доступа set можно реализовывать как элемент, воплощающий выражение. В приведенном ниже примере методы доступа get и set реализуются как члены, воплощающие выражение.

C#Копировать

class TimePeriod

{

private double \_seconds;

public double Seconds

{

get => \_seconds;

set => \_seconds = value;

}

}

В простых случаях, когда методы доступа get и set свойства не выполняют никаких иных операций, кроме задания или извлечения значения в закрытом поле, можно использовать поддержку автоматически реализуемых свойств в компиляторе C#. В приведенном ниже примере Hours реализуется как автоматически реализуемое свойство.

C#Копировать

class TimePeriod2

{

public double Hours { get; set; }

}

# Vector

Предоставляет коллекцию статических методов для создания универсальных векторов, управления ими и работы с другими способами.

C#Копировать

public static class Vector

Наследование

[Object](https://learn.microsoft.com/ru-ru/dotnet/api/system.object?view=net-8.0)

Vector

Свойства

Развернуть таблицу

|  |  |
| --- | --- |
| [IsHardwareAccelerated](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.ishardwareaccelerated?view=net-8.0#system-numerics-vector-ishardwareaccelerated) | Возвращает значение, указывающее, можно ли применить к операциям с векторами аппаратное ускорение посредством встроенной поддержки JIT. |

Методы

Развернуть таблицу

|  |  |
| --- | --- |
| [Abs<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.abs?view=net-8.0#system-numerics-vector-abs-1(system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого являются абсолютными значениями элементов данного вектора. |
| [Add<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.add?view=net-8.0#system-numerics-vector-add-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, значения которого являются суммой каждой пары элементов из двух заданных векторов. |
| [AndNot<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.andnot?view=net-8.0#system-numerics-vector-andnot-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, выполняя битовую операцию И НЕ для каждой пары соответствующих элементов в двух векторах. |
| [As<TFrom,TTo>(Vector<TFrom>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.as?view=net-8.0#system-numerics-vector-as-2(system-numerics-vector((-0)))) | Переинтерпретирует [Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0) как новый [Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [AsVectorByte<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorbyte?view=net-8.0#system-numerics-vector-asvectorbyte-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора байтов без знака. |
| [AsVectorDouble<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectordouble?view=net-8.0#system-numerics-vector-asvectordouble-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора двойной точности с плавающей запятой. |
| [AsVectorInt16<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorint16?view=net-8.0#system-numerics-vector-asvectorint16-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора 16-разрядных целых чисел. |
| [AsVectorInt32<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorint32?view=net-8.0#system-numerics-vector-asvectorint32-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора целых чисел. |
| [AsVectorInt64<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorint64?view=net-8.0#system-numerics-vector-asvectorint64-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора длинных целых чисел. |
| [AsVectorNInt<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectornint?view=net-8.0#system-numerics-vector-asvectornint-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора собственного размера целых чисел. |
| [AsVectorNUInt<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectornuint?view=net-8.0#system-numerics-vector-asvectornuint-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты указанного вектора в биты вектора собственного размера без знака целых чисел. |
| [AsVectorSByte<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorsbyte?view=net-8.0#system-numerics-vector-asvectorsbyte-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора байтов со знаком. |
| [AsVectorSingle<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectorsingle?view=net-8.0#system-numerics-vector-asvectorsingle-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора одинарной точности с плавающей запятой. |
| [AsVectorUInt16<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectoruint16?view=net-8.0#system-numerics-vector-asvectoruint16-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора 16-разрядных целых чисел без знака. |
| [AsVectorUInt32<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectoruint32?view=net-8.0#system-numerics-vector-asvectoruint32-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора целых чисел без знака. |
| [AsVectorUInt64<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.asvectoruint64?view=net-8.0#system-numerics-vector-asvectoruint64-1(system-numerics-vector((-0)))) | Повторно интерпретирует биты заданного вектора в биты вектора длинных целых чисел без знака. |
| [BitwiseAnd<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.bitwiseand?view=net-8.0#system-numerics-vector-bitwiseand-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, выполняя битовую операцию And для всех пар элементов в двух векторах. |
| [BitwiseOr<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.bitwiseor?view=net-8.0#system-numerics-vector-bitwiseor-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, выполняя битовую операцию Or для всех пар элементов в двух векторах. |
| [Ceiling(Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.ceiling?view=net-8.0#system-numerics-vector-ceiling(system-numerics-vector((system-double)))) | Возвращает новый вектор, элементы которого являются наименьшими целочисленными значениями, каждое из которых больше значения соответствующего элемента указанного вектора или равно этому значению. |
| [Ceiling(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.ceiling?view=net-8.0#system-numerics-vector-ceiling(system-numerics-vector((system-single)))) | Возвращает новый вектор, элементы которого являются наименьшими целочисленными значениями, каждое из которых больше значения соответствующего элемента указанного вектора или равно этому значению. |
| [ConditionalSelect(Vector<Int32>, Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.conditionalselect?view=net-8.0#system-numerics-vector-conditionalselect(system-numerics-vector((system-int32))-system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Создает новый вектор одинарной точности с элементами, выбранными между двумя заданными исходными векторами одинарной точности на основе целого вектора маски. |
| [ConditionalSelect(Vector<Int64>, Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.conditionalselect?view=net-8.0#system-numerics-vector-conditionalselect(system-numerics-vector((system-int64))-system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Создает новый вектор двойной точности с элементами, выбранными между двумя заданными исходными векторами двойной точности на основе целого вектора маски. |
| [ConditionalSelect<T>(Vector<T>, Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.conditionalselect?view=net-8.0#system-numerics-vector-conditionalselect-1(system-numerics-vector((-0))-system-numerics-vector((-0))-system-numerics-vector((-0)))) | Создает новый вектор заданного типа с элементами, выбранными между двумя заданными исходными векторами того же типа на основе целого вектора маски. |
| [ConvertToDouble(Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttodouble?view=net-8.0#system-numerics-vector-converttodouble(system-numerics-vector((system-int64)))) | Преобразует Vector<Int64> в Vector<Double>. |
| [ConvertToDouble(Vector<UInt64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttodouble?view=net-8.0#system-numerics-vector-converttodouble(system-numerics-vector((system-uint64)))) | Преобразует Vector<UInt64> в Vector<Double>. |
| [ConvertToInt32(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttoint32?view=net-8.0#system-numerics-vector-converttoint32(system-numerics-vector((system-single)))) | Преобразует Vector<Single> в Vector<Int32>. |
| [ConvertToInt64(Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttoint64?view=net-8.0#system-numerics-vector-converttoint64(system-numerics-vector((system-double)))) | Преобразует Vector<Double> в Vector<Int64>. |
| [ConvertToSingle(Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttosingle?view=net-8.0#system-numerics-vector-converttosingle(system-numerics-vector((system-int32)))) | Преобразует Vector<Int32> в Vector<Single>. |
| [ConvertToSingle(Vector<UInt32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttosingle?view=net-8.0#system-numerics-vector-converttosingle(system-numerics-vector((system-uint32)))) | Преобразует Vector<UInt32> в Vector<Single>. |
| [ConvertToUInt32(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttouint32?view=net-8.0#system-numerics-vector-converttouint32(system-numerics-vector((system-single)))) | Преобразует Vector<Single> в Vector<UInt32>. |
| [ConvertToUInt64(Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.converttouint64?view=net-8.0#system-numerics-vector-converttouint64(system-numerics-vector((system-double)))) | Преобразует Vector<Double> в Vector<UInt64>. |
| [Divide<T>(Vector<T>, T)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.divide?view=net-8.0#system-numerics-vector-divide-1(system-numerics-vector((-0))-0)) | Делит вектор на скаляр для вычисления частного элемента. |
| [Divide<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.divide?view=net-8.0#system-numerics-vector-divide-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, значения которого являются результатом деления элементов первого вектора на соответствующие элементы второго вектора. |
| [Dot<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.dot?view=net-8.0#system-numerics-vector-dot-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает скалярное произведение двух векторов. |
| [Equals(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equals?view=net-8.0#system-numerics-vector-equals(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Возвращает новый целый вектор, элементы которого указывают, равны ли элементы в двух заданных векторах двойной точности. |
| [Equals(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equals?view=net-8.0#system-numerics-vector-equals(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Возвращает новый целый вектор, элементы которого указывают, равны ли элементы в двух заданных целых векторах. |
| [Equals(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equals?view=net-8.0#system-numerics-vector-equals(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Возвращает новый вектор, элементы которого указывают, равны ли элементы в двух заданных векторах длинных целых чисел. |
| [Equals(Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equals?view=net-8.0#system-numerics-vector-equals(system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Возвращает новый целый вектор, элементы которого указывают, равны ли элементы в двух заданных векторах одинарной точности. |
| [Equals<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equals?view=net-8.0#system-numerics-vector-equals-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор заданного типа, элементы которого указывают, равны ли элементы в двух заданных векторах того же типа. |
| [EqualsAll<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equalsall?view=net-8.0#system-numerics-vector-equalsall-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, равны ли пары элементов в данных векторах. |
| [EqualsAny<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.equalsany?view=net-8.0#system-numerics-vector-equalsany-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, равна ли каждая отдельная пара элементов в данных векторах. |
| [Floor(Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.floor?view=net-8.0#system-numerics-vector-floor(system-numerics-vector((system-double)))) | Возвращает новый вектор, элементы которого являются наибольшими целочисленными значениями, каждое из которых меньше значения соответствующего элемента указанного вектора или равно этому значению. |
| [Floor(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.floor?view=net-8.0#system-numerics-vector-floor(system-numerics-vector((system-single)))) | Возвращает новый вектор, элементы которого являются наибольшими целочисленными значениями, каждое из которых меньше значения соответствующего элемента указанного вектора или равно этому значению. |
| [GetElement<T>(Vector<T>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.getelement?view=net-8.0#system-numerics-vector-getelement-1(system-numerics-vector((-0))-system-int32)) | Получает элемент с указанным индексом. |
| [GreaterThan(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthan?view=net-8.0#system-numerics-vector-greaterthan(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном векторе двойной точности с плавающей запятой, чем соответствующие элементы во втором векторе двойной точности с плавающей запятой. |
| [GreaterThan(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthan?view=net-8.0#system-numerics-vector-greaterthan(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном целом векторе, чем соответствующие элементы во втором целом векторе. |
| [GreaterThan(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthan?view=net-8.0#system-numerics-vector-greaterthan(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Возвращает новый вектор длинных целых чисел, элементы которого указывают, больше ли элементы в одном векторе длинных целых чисел, чем соответствующие элементы во втором векторе длинных целых чисел. |
| [GreaterThan(Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthan?view=net-8.0#system-numerics-vector-greaterthan(system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном векторе одинарной точности с плавающей запятой, чем соответствующие элементы во втором векторе одинарной точности с плавающей запятой. |
| [GreaterThan<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthan?view=net-8.0#system-numerics-vector-greaterthan-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого указывают, больше ли элементы в одном векторе заданного типа, чем соответствующие элементы во втором векторе того же типа. |
| [GreaterThanAll<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanall?view=net-8.0#system-numerics-vector-greaterthanall-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, больше ли все элементы в первом векторе, чем соответствующие элементы во втором векторе. |
| [GreaterThanAny<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanany?view=net-8.0#system-numerics-vector-greaterthanany-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, больше ли любой элемент в первом векторе, чем соответствующий элемент во втором векторе. |
| [GreaterThanOrEqual(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequal?view=net-8.0#system-numerics-vector-greaterthanorequal(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном векторе, чем соответствующие элементы во втором векторе двойной точности с плавающей запятой, или равны им. |
| [GreaterThanOrEqual(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequal?view=net-8.0#system-numerics-vector-greaterthanorequal(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном целом векторе, чем соответствующие элементы во втором целом векторе, или равны им. |
| [GreaterThanOrEqual(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequal?view=net-8.0#system-numerics-vector-greaterthanorequal(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Возвращает новый вектор длинных целых чисел, элементы которого указывают, больше ли элементы в одном векторе длинных целых чисел, чем соответствующие элементы во втором векторе длинных целых чисел, или равны им. |
| [GreaterThanOrEqual(Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequal?view=net-8.0#system-numerics-vector-greaterthanorequal(system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Возвращает новый целый вектор, элементы которого указывают, больше ли элементы в одном векторе, чем соответствующие элементы во втором векторе одинарной точности с плавающей запятой, или равны им. |
| [GreaterThanOrEqual<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequal?view=net-8.0#system-numerics-vector-greaterthanorequal-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого указывают, больше ли элементы в одном векторе заданного типа, чем соответствующие элементы во втором векторе заданного типа, или равны им. |
| [GreaterThanOrEqualAll<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequalall?view=net-8.0#system-numerics-vector-greaterthanorequalall-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, больше ли все элементы в первом векторе, чем все соответствующие элементы во втором векторе, или равны им. |
| [GreaterThanOrEqualAny<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.greaterthanorequalany?view=net-8.0#system-numerics-vector-greaterthanorequalany-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, больше ли любой элемент в первом векторе, чем соответствующий элемент во втором векторе, или равен ему. |
| [LessThan(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthan?view=net-8.0#system-numerics-vector-lessthan(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном векторе двойной точности с плавающей запятой, чем соответствующие элементы во втором векторе двойной точности с плавающей запятой. |
| [LessThan(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthan?view=net-8.0#system-numerics-vector-lessthan(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном целом векторе, чем соответствующие элементы во втором целом векторе. |
| [LessThan(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthan?view=net-8.0#system-numerics-vector-lessthan(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Возвращает новый вектор длинных целых чисел, элементы которого указывают, меньше ли элементы в одном векторе длинных целых чисел, чем соответствующие элементы во втором векторе длинных целых чисел. |
| [LessThan(Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthan?view=net-8.0#system-numerics-vector-lessthan(system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном векторе одинарной точности, чем соответствующие элементы во втором векторе одинарной точности. |
| [LessThan<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthan?view=net-8.0#system-numerics-vector-lessthan-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор заданного типа, элементы которого указывают, меньше ли элементы в одном векторе, чем соответствующие элементы во втором векторе. |
| [LessThanAll<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanall?view=net-8.0#system-numerics-vector-lessthanall-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, меньше ли все элементы в первом векторе, чем соответствующие элементы во втором векторе. |
| [LessThanAny<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanany?view=net-8.0#system-numerics-vector-lessthanany-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, меньше ли любой элемент в первом векторе, чем соответствующий элемент во втором векторе. |
| [LessThanOrEqual(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequal?view=net-8.0#system-numerics-vector-lessthanorequal(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном векторе двойной точности с плавающей запятой, чем соответствующие элементы во втором векторе двойной точности с плавающей запятой, или равны им. |
| [LessThanOrEqual(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequal?view=net-8.0#system-numerics-vector-lessthanorequal(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном целом векторе, чем соответствующие элементы во втором целом векторе, или равны им. |
| [LessThanOrEqual(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequal?view=net-8.0#system-numerics-vector-lessthanorequal(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Возвращает новый вектор длинных целых чисел, элементы которого указывают, меньше ли элементы в одном векторе длинных целых чисел, чем соответствующие элементы во втором векторе длинных целых чисел, или равны им. |
| [LessThanOrEqual(Vector<Single>, Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequal?view=net-8.0#system-numerics-vector-lessthanorequal(system-numerics-vector((system-single))-system-numerics-vector((system-single)))) | Возвращает новый целый вектор, элементы которого указывают, меньше ли элементы в одном векторе одинарной точности с плавающей запятой, чем соответствующие элементы во втором векторе одинарной точности с плавающей запятой, или равны им. |
| [LessThanOrEqual<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequal?view=net-8.0#system-numerics-vector-lessthanorequal-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого указывают, меньше ли элементы в одном векторе, чем соответствующие элементы во втором векторе, или равны им. |
| [LessThanOrEqualAll<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequalall?view=net-8.0#system-numerics-vector-lessthanorequalall-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, меньше ли все элементы в первом векторе, чем соответствующие элементы во втором векторе, или равны им. |
| [LessThanOrEqualAny<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.lessthanorequalany?view=net-8.0#system-numerics-vector-lessthanorequalany-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает значение, указывающее, меньше ли любой элемент в первом векторе, чем соответствующий элемент во втором векторе, или равен ему. |
| [Load<T>(T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.load?view=net-8.0#system-numerics-vector-load-1(-0*)) | Загружает вектор из заданного источника. |
| [LoadAligned<T>(T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.loadaligned?view=net-8.0#system-numerics-vector-loadaligned-1(-0*)) | Загружает вектор из заданного выровненного источника. |
| [LoadAlignedNonTemporal<T>(T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.loadalignednontemporal?view=net-8.0#system-numerics-vector-loadalignednontemporal-1(-0*)) | Загружает вектор из заданного выровненного источника. |
| [LoadUnsafe<T>(T)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.loadunsafe?view=net-8.0#system-numerics-vector-loadunsafe-1(-0@)) | Загружает вектор из заданного источника. |
| [LoadUnsafe<T>(T, UIntPtr)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.loadunsafe?view=net-8.0#system-numerics-vector-loadunsafe-1(-0@-system-uintptr)) | Загружает вектор из заданного источника и смещения элемента. |
| [Max<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.max?view=net-8.0#system-numerics-vector-max-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого являются максимальным значением каждой пары элементов в двух данных векторах. |
| [Min<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.min?view=net-8.0#system-numerics-vector-min-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого являются минимальным значением каждой пары элементов в двух данных векторах. |
| [Multiply<T>(T, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.multiply?view=net-8.0#system-numerics-vector-multiply-1(-0-system-numerics-vector((-0)))) | Возвращает новый вектор, значения которого являются скалярным значением, умноженным на каждое из значений заданного вектора. |
| [Multiply<T>(Vector<T>, T)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.multiply?view=net-8.0#system-numerics-vector-multiply-1(system-numerics-vector((-0))-0)) | Возвращает новый вектор, значения которого являются значениями заданного вектора, каждое из которых умножено на скалярное значение. |
| [Multiply<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.multiply?view=net-8.0#system-numerics-vector-multiply-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, значения которого являются произведением каждой пары элементов в двух заданных векторах. |
| [Narrow(Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-double))-system-numerics-vector((system-double)))) | Сводит два экземпляра Vector<Double> в один Vector<Single>. |
| [Narrow(Vector<Int16>, Vector<Int16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-int16))-system-numerics-vector((system-int16)))) | Сводит два экземпляра Vector<Int16> в один Vector<SByte>. |
| [Narrow(Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-int32))-system-numerics-vector((system-int32)))) | Сводит два экземпляра Vector<Int32> в один Vector<Int16>. |
| [Narrow(Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-int64))-system-numerics-vector((system-int64)))) | Сводит два экземпляра Vector<Int64> в один Vector<Int32>. |
| [Narrow(Vector<UInt16>, Vector<UInt16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-uint16))-system-numerics-vector((system-uint16)))) | Сводит два экземпляра Vector<UInt16> в один Vector<Byte>. |
| [Narrow(Vector<UInt32>, Vector<UInt32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-uint32))-system-numerics-vector((system-uint32)))) | Сводит два экземпляра Vector<UInt32> в один Vector<UInt16>. |
| [Narrow(Vector<UInt64>, Vector<UInt64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.narrow?view=net-8.0#system-numerics-vector-narrow(system-numerics-vector((system-uint64))-system-numerics-vector((system-uint64)))) | Сводит два экземпляра Vector<UInt64> в один Vector<UInt32>. |
| [Negate<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.negate?view=net-8.0#system-numerics-vector-negate-1(system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого являются отрицательными значениями соответствующих элементов заданного вектора. |
| [OnesComplement<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.onescomplement?view=net-8.0#system-numerics-vector-onescomplement-1(system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого получаются путем принятия дополнения до единицы элементов заданного вектора. |
| [ShiftLeft(Vector<Byte>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-byte))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<Int16>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-int16))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<Int32>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-int32))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<Int64>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-int64))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<IntPtr>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-intptr))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<SByte>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-sbyte))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<UInt16>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-uint16))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<UInt32>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-uint32))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<UInt64>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-uint64))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftLeft(Vector<UIntPtr>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftleft?view=net-8.0#system-numerics-vector-shiftleft(system-numerics-vector((system-uintptr))-system-int32)) | Сдвигает каждый элемент вектора влево на указанную величину. |
| [ShiftRightArithmetic(Vector<Int16>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightarithmetic?view=net-8.0#system-numerics-vector-shiftrightarithmetic(system-numerics-vector((system-int16))-system-int32)) | Сдвигает (со знаком) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightArithmetic(Vector<Int32>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightarithmetic?view=net-8.0#system-numerics-vector-shiftrightarithmetic(system-numerics-vector((system-int32))-system-int32)) | Сдвигает (со знаком) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightArithmetic(Vector<Int64>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightarithmetic?view=net-8.0#system-numerics-vector-shiftrightarithmetic(system-numerics-vector((system-int64))-system-int32)) | Сдвигает (со знаком) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightArithmetic(Vector<IntPtr>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightarithmetic?view=net-8.0#system-numerics-vector-shiftrightarithmetic(system-numerics-vector((system-intptr))-system-int32)) | Сдвигает (со знаком) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightArithmetic(Vector<SByte>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightarithmetic?view=net-8.0#system-numerics-vector-shiftrightarithmetic(system-numerics-vector((system-sbyte))-system-int32)) | Сдвигает (со знаком) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<Byte>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-byte))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<Int16>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-int16))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<Int32>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-int32))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<Int64>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-int64))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<IntPtr>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-intptr))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<SByte>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-sbyte))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<UInt16>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-uint16))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<UInt32>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-uint32))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<UInt64>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-uint64))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [ShiftRightLogical(Vector<UIntPtr>, Int32)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.shiftrightlogical?view=net-8.0#system-numerics-vector-shiftrightlogical(system-numerics-vector((system-uintptr))-system-int32)) | Сдвигает (без знака) каждый элемент вектора вправо на указанную величину. |
| [SquareRoot<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.squareroot?view=net-8.0#system-numerics-vector-squareroot-1(system-numerics-vector((-0)))) | Возвращает новый вектор, элементы которого являются квадратным корнем элементов заданного вектора. |
| [Store<T>(Vector<T>, T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.store?view=net-8.0#system-numerics-vector-store-1(system-numerics-vector((-0))-0*)) | Сохраняет вектор в заданном месте назначения. |
| [StoreAligned<T>(Vector<T>, T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.storealigned?view=net-8.0#system-numerics-vector-storealigned-1(system-numerics-vector((-0))-0*)) | Сохраняет вектор в заданном выровненном месте назначения. |
| [StoreAlignedNonTemporal<T>(Vector<T>, T\*)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.storealignednontemporal?view=net-8.0#system-numerics-vector-storealignednontemporal-1(system-numerics-vector((-0))-0*)) | Сохраняет вектор в заданном выровненном месте назначения. |
| [StoreUnsafe<T>(Vector<T>, T)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.storeunsafe?view=net-8.0#system-numerics-vector-storeunsafe-1(system-numerics-vector((-0))-0@)) | Сохраняет вектор в заданном месте назначения. |
| [StoreUnsafe<T>(Vector<T>, T, UIntPtr)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.storeunsafe?view=net-8.0#system-numerics-vector-storeunsafe-1(system-numerics-vector((-0))-0@-system-uintptr)) | Сохраняет вектор в заданном месте назначения. |
| [Subtract<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.subtract?view=net-8.0#system-numerics-vector-subtract-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, значения которого являются разностью элементов второго вектора и соответствующих элементов первого вектора. |
| [Sum<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.sum?view=net-8.0#system-numerics-vector-sum-1(system-numerics-vector((-0)))) | Возвращает сумму всех элементов в указанном векторе. |
| [ToScalar<T>(Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.toscalar?view=net-8.0#system-numerics-vector-toscalar-1(system-numerics-vector((-0)))) | Преобразует заданный вектор в скаляр, содержащий значение первого элемента. |
| [Widen(Vector<Byte>, Vector<UInt16>, Vector<UInt16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-byte))-system-numerics-vector((system-uint16))@-system-numerics-vector((system-uint16))@)) | Расширяет Vector<Byte> в два экземпляра Vector<UInt16>. |
| [Widen(Vector<Int16>, Vector<Int32>, Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-int16))-system-numerics-vector((system-int32))@-system-numerics-vector((system-int32))@)) | Расширяет Vector<Int16> в два экземпляра Vector<Int32>. |
| [Widen(Vector<Int32>, Vector<Int64>, Vector<Int64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-int32))-system-numerics-vector((system-int64))@-system-numerics-vector((system-int64))@)) | Расширяет Vector<Int32> в два экземпляра Vector<Int64>. |
| [Widen(Vector<SByte>, Vector<Int16>, Vector<Int16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-sbyte))-system-numerics-vector((system-int16))@-system-numerics-vector((system-int16))@)) | Расширяет Vector<SByte> в два экземпляра Vector<Int16>. |
| [Widen(Vector<Single>, Vector<Double>, Vector<Double>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-single))-system-numerics-vector((system-double))@-system-numerics-vector((system-double))@)) | Расширяет Vector<Single> в два экземпляра Vector<Double>. |
| [Widen(Vector<UInt16>, Vector<UInt32>, Vector<UInt32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-uint16))-system-numerics-vector((system-uint32))@-system-numerics-vector((system-uint32))@)) | Расширяет Vector<UInt16> в два экземпляра Vector<UInt32>. |
| [Widen(Vector<UInt32>, Vector<UInt64>, Vector<UInt64>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widen?view=net-8.0#system-numerics-vector-widen(system-numerics-vector((system-uint32))-system-numerics-vector((system-uint64))@-system-numerics-vector((system-uint64))@)) | Расширяет Vector<UInt32> в два экземпляра Vector<UInt64>. |
| [WidenLower(Vector<Byte>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-byte)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<Int16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-int16)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-int32)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<SByte>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-sbyte)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-single)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<UInt16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-uint16)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenLower(Vector<UInt32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenlower?view=net-8.0#system-numerics-vector-widenlower(system-numerics-vector((system-uint32)))) | Расширяет нижнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<Byte>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-byte)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<Int16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-int16)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<Int32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-int32)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<SByte>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-sbyte)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<Single>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-single)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<UInt16>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-uint16)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WidenUpper(Vector<UInt32>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.widenupper?view=net-8.0#system-numerics-vector-widenupper(system-numerics-vector((system-uint32)))) | Расширяет верхнюю половину до [Vector<T>Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0). |
| [WithElement<T>(Vector<T>, Int32, T)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.withelement?view=net-8.0#system-numerics-vector-withelement-1(system-numerics-vector((-0))-system-int32-0)) | Создает новый [Vector<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector-1?view=net-8.0) с элементом по указанному индексу, для которого задано указанное значение, а остальные элементы получают то же значение, что и в данном векторе. |
| [Xor<T>(Vector<T>, Vector<T>)](https://learn.microsoft.com/ru-ru/dotnet/api/system.numerics.vector.xor?view=net-8.0#system-numerics-vector-xor-1(system-numerics-vector((-0))-system-numerics-vector((-0)))) | Возвращает новый вектор, выполняя битовую операцию исключающего ИЛИ (XOr) для всех пар элементов в двух векторах. |

Применяется к

|  |  |
| --- | --- |
| Продукт | **Версии** |
| **.NET** | Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9 |
| **.NET Standard** | 2.1 |
| **UWP** | 10.0 |

# Массивы

В структуре данных массива можно хранить несколько переменных одного типа. Чтобы объявить массив, следует указать тип его элементов. Если требуется, чтобы массив мог хранить элементы любого типа, можно указать object в качестве его типа. В унифицированной системе типов C# все типы, стандартные и определяемые пользователем, ссылочные типы и типы значений напрямую или косвенно наследуются из [Object](https://learn.microsoft.com/ru-ru/dotnet/api/system.object).

C#Копировать

type[] arrayName;

Массив имеет следующие свойства:

Массив может быть одномерным[,](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/arrays#single-dimensional-arrays)[многомерным или](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/arrays#multidimensional-arrays)[многомерным](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/arrays#jagged-arrays).

Число измерений задается при объявлении переменной массива. Длина каждого измерения устанавливается при создании экземпляра массива. Эти значения нельзя изменить во время существования экземпляра.

Массив массивов является массивом массивов, и каждый массив элементов имеет значение nullпо умолчанию.

Массивы индексируются от нуля: массив с n элементами индексируется от 0 до n-1.

Элементы массива могут иметь любой тип, в том числе тип массива.

Типы массивов — это [ссылочные типы](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/reference-types), производные от абстрактного базового типа [Array](https://learn.microsoft.com/ru-ru/dotnet/api/system.array). Все массивы реализуют [IList](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.ilist) и [IEnumerable](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.ienumerable). Для итерации по массиву можно использовать оператор [foreach](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/iteration-statements" \l "the-foreach-statement). Одномерные массивы также реализуют [IList<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.ilist-1) и [IEnumerable<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.ienumerable-1).

Элементы массива можно инициализировать в известные значения при создании массива. Начиная с C# 12, все типы коллекций можно инициализировать с помощью [выражения](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/collection-expressions) Collection. Элементы, которые не инициализированы, имеют [значение](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/default-values) по умолчанию. Значение по умолчанию — это 0-разрядный шаблон. Все ссылочные типы (включая [типы, не допускающие](https://learn.microsoft.com/ru-ru/dotnet/csharp/nullable-references#known-pitfalls) значение NULL), имеют значения null. Все типы значений имеют 0-разрядные шаблоны. Это означает, [Nullable<T>.HasValue](https://learn.microsoft.com/ru-ru/dotnet/api/system.nullable-1.hasvalue" \l "system-nullable-1-hasvalue) что свойство имеет false значение, и [Nullable<T>.Value](https://learn.microsoft.com/ru-ru/dotnet/api/system.nullable-1.value" \l "system-nullable-1-value) свойство не определено. В реализации Value .NET свойство создает исключение.

В следующих примерах создаются одномерные массивы, многомерные массивы и массивы массивов:

C#Копировать

// Declare a single-dimensional array of 5 integers.

int[] array1 = new int[5];

// Declare and set array element values.

int[] array2 = [1, 2, 3, 4, 5, 6];

// Declare a two dimensional array.

int[,] multiDimensionalArray1 = new int[2, 3];

// Declare and set array element values.

int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

// Declare a jagged array.

int[][] jaggedArray = new int[6][];

// Set the values of the first array in the jagged array structure.

jaggedArray[0] = [1, 2, 3, 4];

Одномерные массивы

Одномерный массив — это последовательность таких элементов. Доступ к элементу осуществляется через его индекс. Индекс — это порядковое положение в последовательности. Первый элемент в массиве находится в индексе 0. Для создания одномерного массива используется оператор [new](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/new-operator) и указывается тип элементов массива и число элементов. В следующем примере объявляются и инициализируется одномерные массивы:

C#Копировать

int[] array = new int[5];

string[] weekDays = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

Console.WriteLine(weekDays[0]);

Console.WriteLine(weekDays[1]);

Console.WriteLine(weekDays[2]);

Console.WriteLine(weekDays[3]);

Console.WriteLine(weekDays[4]);

Console.WriteLine(weekDays[5]);

Console.WriteLine(weekDays[6]);

/\*Output:

Sun

Mon

Tue

Wed

Thu

Fri

Sat

\*/

Первое объявление объявляет неинициализированный массив из пяти целых чисел от array[0] до array[4]. Элементы массива инициализируются до [значения по умолчанию](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/default-values) для типа элемента. Для целых чисел это 0. Второе объявление объявляет массив строк и инициализирует все семь значений этого массива. Оператор [foreach](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/iteration-statements" \l "the-foreach-statement) выполняет итерацию элементов массива weekday и выводит все значения. Для одномерных массивов оператор обрабатывает элементы в увеличении порядка индексов foreach , начиная с индекса 0 и заканчивая индексом Length - 1.

Передача одномерных массивов в качестве аргументов

Инициализированный одномерный массив можно передать в метод. В следующем примере массив строк инициализируется и передается в качестве аргумента в метод DisplayArray для строк. Этот метод отображает элементы массива. Затем метод ChangeArray размещает элементы массива в обратном порядке, а метод ChangeArrayElements изменяет первые три элемента массива. После возврата каждого метода метод DisplayArray показывает, что передача массива по значению не препятствует изменению элементов массива.

C#Копировать

class ArrayExample

{

static void DisplayArray(string[] arr) => Console.WriteLine(string.Join(" ", arr));

// Change the array by reversing its elements.

static void ChangeArray(string[] arr) => Array.Reverse(arr);

static void ChangeArrayElements(string[] arr)

{

// Change the value of the first three array elements.

arr[0] = "Mon";

arr[1] = "Wed";

arr[2] = "Fri";

}

static void Main()

{

// Declare and initialize an array.

string[] weekDays = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

// Display the array elements.

DisplayArray(weekDays);

Console.WriteLine();

// Reverse the array.

ChangeArray(weekDays);

// Display the array again to verify that it stays reversed.

Console.WriteLine("Array weekDays after the call to ChangeArray:");

DisplayArray(weekDays);

Console.WriteLine();

// Assign new values to individual array elements.

ChangeArrayElements(weekDays);

// Display the array again to verify that it has changed.

Console.WriteLine("Array weekDays after the call to ChangeArrayElements:");

DisplayArray(weekDays);

}

}

// The example displays the following output:

// Sun Mon Tue Wed Thu Fri Sat

//

// Array weekDays after the call to ChangeArray:

// Sat Fri Thu Wed Tue Mon Sun

//

// Array weekDays after the call to ChangeArrayElements:

// Mon Wed Fri Wed Tue Mon Sun

Многомерные массивы

Массивы могут иметь несколько измерений. Например, следующие объявления создают четыре массива: два имеют два измерения, два имеют три измерения. Первые два объявления объявляют длину каждого измерения, но не инициализировать значения массива. Во втором двух объявлениях используется инициализатор для задания значений каждого элемента в многомерном массиве.

C#Копировать

int[,] array2DDeclaration = new int[4, 2];

int[,,] array3DDeclaration = new int[4, 2, 3];

// Two-dimensional array.

int[,] array2DInitialization = { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

// Three-dimensional array.

int[,,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

// Accessing array elements.

System.Console.WriteLine(array2DInitialization[0, 0]);

System.Console.WriteLine(array2DInitialization[0, 1]);

System.Console.WriteLine(array2DInitialization[1, 0]);

System.Console.WriteLine(array2DInitialization[1, 1]);

System.Console.WriteLine(array2DInitialization[3, 0]);

System.Console.WriteLine(array2DInitialization[3, 1]);

// Output:

// 1

// 2

// 3

// 4

// 7

// 8

System.Console.WriteLine(array3D[1, 0, 1]);

System.Console.WriteLine(array3D[1, 1, 2]);

// Output:

// 8

// 12

// Getting the total count of elements or the length of a given dimension.

var allLength = array3D.Length;

var total = 1;

for (int i = 0; i < array3D.Rank; i++)

{

total \*= array3D.GetLength(i);

}

System.Console.WriteLine($"{allLength} equals {total}");

// Output:

// 12 equals 12

Для многомерных массивов элементы пересекаются таким образом, что индексы самого правого измерения увеличиваются сначала, а затем следующее левое измерение и т. д. до самого левого индекса. В следующем примере перечисляется как 2D, так и трехмерный массив:

C#Копировать

int[,] numbers2D = { { 9, 99 }, { 3, 33 }, { 5, 55 } };

foreach (int i in numbers2D)

{

System.Console.Write($"{i} ");

}

// Output: 9 99 3 33 5 55

int[,,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

foreach (int i in array3D)

{

System.Console.Write($"{i} ");

}

// Output: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

В массиве 2D можно считать левый индекс строкой и правым индексом в качестве столбца.

Однако с многомерными массивами, используя вложенный [цикл,](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/iteration-statements#the-for-statement) вы можете более контролировать порядок обработки элементов массива:

C#Копировать

int[,,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

for (int i = 0; i < array3D.GetLength(0); i++)

{

for (int j = 0; j < array3D.GetLength(1); j++)

{

for (int k = 0; k < array3D.GetLength(2); k++)

{

System.Console.Write($"{array3D[i, j, k]} ");

}

System.Console.WriteLine();

}

System.Console.WriteLine();

}

// Output (including blank lines):

// 1 2 3

// 4 5 6

//

// 7 8 9

// 10 11 12

//

Передача многомерных массивов в качестве аргументов

Инициализированный многомерный массив можно передать в метод так же, как и одномерный массив. В следующем коде показано разделяемое объявление метода печати, который принимает в качестве аргумента двухмерный массив. Новый массив можно инициализировать и передать за один шаг, как показано в следующем примере. В следующем примере инициализируется двухмерный массив целых чисел, который передается в метод Print2DArray. Этот метод отображает элементы массива.

C#Копировать

static void Print2DArray(int[,] arr)

{

// Display the array elements.

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int j = 0; j < arr.GetLength(1); j++)

{

System.Console.WriteLine("Element({0},{1})={2}", i, j, arr[i, j]);

}

}

}

static void ExampleUsage()

{

// Pass the array as an argument.

Print2DArray(new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } });

}

/\* Output:

Element(0,0)=1

Element(0,1)=2

Element(1,0)=3

Element(1,1)=4

Element(2,0)=5

Element(2,1)=6

Element(3,0)=7

Element(3,1)=8

\*/

Массивы массивов

Массив массивов — это массив, элементы которого являются массивами и могут быть различных размеров. Многообразный массив иногда называется массивом массивов. Его элементы являются ссылочными типами и инициализированы в null. В следующих примерах показано, как объявлять и инициализировать массивы массивов, а также получать доступ к ним. Первый пример объявляется jaggedArrayв одной инструкции. Каждый содержащийся массив создается в последующих инструкциях. Второй пример jaggedArray2 объявлен и инициализирован в одной инструкции. Массивы массивов и многомерные массивы можно смешивать. Последним примером jaggedArray3является объявление и инициализация одномерного массива, содержащего три двухмерных элемента массива разных размеров.

C#Копировать

int[][] jaggedArray = new int[3][];

jaggedArray[0] = [1, 3, 5, 7, 9];

jaggedArray[1] = [0, 2, 4, 6];

jaggedArray[2] = [11, 22];

int[][] jaggedArray2 =

[

[1, 3, 5, 7, 9],

[0, 2, 4, 6],

[11, 22]

];

// Assign 77 to the second element ([1]) of the first array ([0]):

jaggedArray2[0][1] = 77;

// Assign 88 to the second element ([1]) of the third array ([2]):

jaggedArray2[2][1] = 88;

int[][,] jaggedArray3 =

[

new int[,] { {1,3}, {5,7} },

new int[,] { {0,2}, {4,6}, {8,10} },

new int[,] { {11,22}, {99,88}, {0,9} }

];

Console.Write("{0}", jaggedArray3[0][1, 0]);

Console.WriteLine(jaggedArray3.Length);

Элементы массива должны быть инициализированы перед их использованием. Каждый из элементов является массивом. Кроме того, можно использовать инициализаторы для заполнения элементов массива значениями. При использовании инициализаторов не требуется размер массива.

В этом примере создается массив, элементы которого являются массивами. Все элементы массива имеют разный размер.

C#Копировать

// Declare the array of two elements.

int[][] arr = new int[2][];

// Initialize the elements.

arr[0] = [1, 3, 5, 7, 9];

arr[1] = [2, 4, 6, 8];

// Display the array elements.

for (int i = 0; i < arr.Length; i++)

{

System.Console.Write("Element({0}): ", i);

for (int j = 0; j < arr[i].Length; j++)

{

System.Console.Write("{0}{1}", arr[i][j], j == (arr[i].Length - 1) ? "" : " ");

}

System.Console.WriteLine();

}

/\* Output:

Element(0): 1 3 5 7 9

Element(1): 2 4 6 8

\*/

Неявно типизированные массивы

Можно создать неявно типизированный массив, в котором тип экземпляра массива выводится из элементов, указанных в инициализаторе массива. Правила для любой неявно типизированной переменной также применяются к неявно типизированным массивам. Дополнительные сведения см. в разделе [Неявно типизированные локальные переменные](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/implicitly-typed-local-variables).

В следующих примерах показано, как создать неявно типизированный массив:

C#Копировать

int[] a = new[] { 1, 10, 100, 1000 }; // int[]

// Accessing array

Console.WriteLine("First element: " + a[0]);

Console.WriteLine("Second element: " + a[1]);

Console.WriteLine("Third element: " + a[2]);

Console.WriteLine("Fourth element: " + a[3]);

/\* Outputs

First element: 1

Second element: 10

Third element: 100

Fourth element: 1000

\*/

var b = new[] { "hello", null, "world" }; // string[]

// Accessing elements of an array using 'string.Join' method

Console.WriteLine(string.Join(" ", b));

/\* Output

hello world

\*/

// single-dimension jagged array

int[][] c =

[

[1,2,3,4],

[5,6,7,8]

];

// Looping through the outer array

for (int k = 0; k < c.Length; k++)

{

// Looping through each inner array

for (int j = 0; j < c[k].Length; j++)

{

// Accessing each element and printing it to the console

Console.WriteLine($"Element at c[{k}][{j}] is: {c[k][j]}");

}

}

/\* Outputs

Element at c[0][0] is: 1

Element at c[0][1] is: 2

Element at c[0][2] is: 3

Element at c[0][3] is: 4

Element at c[1][0] is: 5

Element at c[1][1] is: 6

Element at c[1][2] is: 7

Element at c[1][3] is: 8

\*/

// jagged array of strings

string[][] d =

[

["Luca", "Mads", "Luke", "Dinesh"],

["Karen", "Suma", "Frances"]

];

// Looping through the outer array

int i = 0;

foreach (var subArray in d)

{

// Looping through each inner array

int j = 0;

foreach (var element in subArray)

{

// Accessing each element and printing it to the console

Console.WriteLine($"Element at d[{i}][{j}] is: {element}");

j++;

}

i++;

}

/\* Outputs

Element at d[0][0] is: Luca

Element at d[0][1] is: Mads

Element at d[0][2] is: Luke

Element at d[0][3] is: Dinesh

Element at d[1][0] is: Karen

Element at d[1][1] is: Suma

Element at d[1][2] is: Frances

\*/

В предыдущем примере обратите внимание, что с неявно типизированными массивами квадратные скобки не используются слева от инструкции инициализации. Кроме того, многомерные массивы инициализированы так new [] же, как и одномерные массивы.

При создании анонимного типа, содержащего массив, этот массив необходимо неявно типизировать в инициализаторе объекта типа. В следующем примере contacts представляет собой неявный типизированный массив анонимных типов, каждый из которых содержит массив с именем PhoneNumbers. Ключевое var слово не используется внутри инициализаторов объектов.

C#Копировать

var contacts = new[]

{

new

{

Name = "Eugene Zabokritski",

PhoneNumbers = new[] { "206-555-0108", "425-555-0001" }

},

new

{

Name = "Hanying Feng",

PhoneNumbers = new[] { "650-555-0199" }

}

};

# Типы кортежей (справочник по C#) Tuple

Функция кортежей предоставляет краткий синтаксис для группировки нескольких элементов данных в упрощенной структуре данных. В следующем примере показано, как можно объявить переменную кортежа, инициализировать ее и получить доступ к ее элементам данных.

C#Копировать

Выполнить

(double, int) t1 = (4.5, 3);

Console.WriteLine($"Tuple with elements {t1.Item1} and {t1.Item2}.");

// Output:

// Tuple with elements 4.5 and 3.

(double Sum, int Count) t2 = (4.5, 3);

Console.WriteLine($"Sum of {t2.Count} elements is {t2.Sum}.");

// Output:

// Sum of 3 elements is 4.5.

Как показано в предыдущем примере, для определения типа кортежа необходимо указать типы всех его элементов данных и, при необходимости, [имена полей](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples#tuple-field-names). Невозможно определить методы в типе кортежа, но можно использовать методы, предоставляемые .NET, как показано в следующем примере:

C#Копировать

Выполнить

(double, int) t = (4.5, 3);

Console.WriteLine(t.ToString());

Console.WriteLine($"Hash code of {t} is {t.GetHashCode()}.");

// Output:

// (4.5, 3)

// Hash code of (4.5, 3) is 718460086.

Типы кортежей поддерживают [операторы](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/equality-operators)== равенства и !=. Дополнительные сведения см. в разделе [Равенство кортежей](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples#tuple-equality).

Типы кортежей являются [типами значений](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types), а элементы кортежа — общедоступными полями. Поэтому кортежи представляют собой изменяемые типы значений.

Можно определить кортежи со сколь угодно большим числом элементов.

C#Копировать

Выполнить

var t =

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

11, 12, 13, 14, 15, 16, 17, 18,

19, 20, 21, 22, 23, 24, 25, 26);

Console.WriteLine(t.Item26); // output: 26

Варианты использования кортежей

Чаще всего кортежи используются как возвращаемый методом тип. То есть вместо определения [параметров метода out](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/method-parameters#out-parameter-modifier) можно сгруппировать результаты метода в возвращаемый тип кортежа, как показано в следующем примере.

C#Копировать

Выполнить

int[] xs = [4, 7, 9];

var limits = FindMinMax(xs);

Console.WriteLine($"Limits of [{string.Join(" ", xs)}] are {limits.min} and {limits.max}");

// Output:

// Limits of [4 7 9] are 4 and 9

int[] ys = [-9, 0, 67, 100];

var (minimum, maximum) = FindMinMax(ys);

Console.WriteLine($"Limits of [{string.Join(" ", ys)}] are {minimum} and {maximum}");

// Output:

// Limits of [-9 0 67 100] are -9 and 100

(int min, int max) FindMinMax(int[] input)

{

if (input is null || input.Length == 0)

{

throw new ArgumentException("Cannot find minimum and maximum of a null or empty array.");

}

// Initialize min to MaxValue so every value in the input

// is less than this initial value.

var min = int.MaxValue;

// Initialize max to MinValue so every value in the input

// is greater than this initial value.

var max = int.MinValue;

foreach (var i in input)

{

if (i < min)

{

min = i;

}

if (i > max)

{

max = i;

}

}

return (min, max);

}

Как показано в предыдущем примере, с возвращаемым экземпляром кортежа можно работать напрямую или [деконструировать](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples#tuple-assignment-and-deconstruction) его в отдельные переменные.

Типы кортежей можно также использовать вместо [анонимных типов](https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/anonymous-types), например в запросах LINQ. Дополнительные сведения см. в статье [Выбор между анонимными типами и кортежами](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/choosing-between-anonymous-and-tuple).

Как правило, кортежи используются для группирования слабо связанных элементов данных. В общедоступных API рекомендуется определить [класс](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/class) или [тип структуры](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/struct) .

Имена полей кортежей

Вы явно указываете имена полей кортежей в выражении инициализации кортежа или в определении типа кортежа, как показано в следующем примере:

C#Копировать

Выполнить

var t = (Sum: 4.5, Count: 3);

Console.WriteLine($"Sum of {t.Count} elements is {t.Sum}.");

(double Sum, int Count) d = (4.5, 3);

Console.WriteLine($"Sum of {d.Count} elements is {d.Sum}.");

Если имя поля не указано, оно может быть выведено из имени соответствующей переменной в выражении инициализации кортежа, как показано в следующем примере:

C#Копировать

Выполнить

var sum = 4.5;

var count = 3;

var t = (sum, count);

Console.WriteLine($"Sum of {t.count} elements is {t.sum}.");

Это называется инициализаторами проекции кортежей. Имя переменной не проецируется на имя поля кортежа в следующих случаях:

Имя кандидата — это имя элемента типа кортежа, например Item3, ToStringили Rest.

Имя кандидата является дубликатом другого имени поля кортежа, явного или неявного.

В предыдущих случаях можно явно указать имя поля или получить доступ к полю по умолчанию.

По умолчанию поля кортежа имеют имена Item1, Item2, Item3 и т. д. Всегда можно использовать имя поля по умолчанию, даже если имя поля указано явно или является выводимым, как показано в следующем примере.

C#Копировать

Выполнить

var a = 1;

var t = (a, b: 2, 3);

Console.WriteLine($"The 1st element is {t.Item1} (same as {t.a}).");

Console.WriteLine($"The 2nd element is {t.Item2} (same as {t.b}).");

Console.WriteLine($"The 3rd element is {t.Item3}.");

// Output:

// The 1st element is 1 (same as 1).

// The 2nd element is 2 (same as 2).

// The 3rd element is 3.

Имена полей не учитываются при [присваивании кортежа](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples#tuple-assignment-and-deconstruction) и [сравнении кортежей на равенство](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples#tuple-equality).

Во время компиляции компилятор заменяет имена полей не по умолчанию соответствующими именами по умолчанию. В результате явно указанные или выводимые имена полей будут недоступны во время выполнения.

**Совет**

Включите правило [**стиля кода .NET IDE0037**](https://learn.microsoft.com/ru-ru/dotnet/fundamentals/code-analysis/style-rules/ide0037) , чтобы задать предпочтения для выводимых или явных имен полей кортежей.

Начиная с C# 12, можно указать псевдоним для типа кортежа с директивой[using](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/using-directive#using-alias). В следующем примере добавляется global using псевдоним для типа кортежа с двумя целыми значениями допустимого Min и Max значения:

C#Копировать

global using BandPass = (int Min, int Max);

После объявления псевдонима можно использовать BandPass имя в качестве псевдонима для этого типа кортежа:

C#Копировать

BandPass bracket = (40, 100);

Console.WriteLine($"The bandpass filter is {bracket.Min} to {bracket.Max}");

Псевдоним не вводит новый тип, но создает синоним только для существующего типа. Вы можете деконструировать кортеж, объявленный псевдонимом BandPass так же, как и с его базовым типом кортежа:

C#Копировать

(int a , int b) = bracket;

Console.WriteLine($"The bracket is {a} to {b}");

Как и при назначении кортежа или деконструкции, имена элементов кортежа не должны соответствовать; Типы выполняются.

Аналогичным образом можно использовать второй псевдоним с одинаковыми типами arity и членов взаимозаменяемо с исходным псевдонимом. Можно объявить второй псевдоним:

C#Копировать

using Range = (int Minimum, int Maximum);

Кортеж можно назначить RangeBandPass кортеже. Как и во всех назначениях кортежей, имена полей не должны совпадать, только типы и arity.

C#Копировать

Range r = bracket;

Console.WriteLine($"The range is {r.Minimum} to {r.Maximum}");

Псевдоним для типа кортежа предоставляет более семантические сведения при использовании кортежей. Он не вводит новый тип. Чтобы обеспечить безопасность типов, вместо этого следует объявить позиционный [record](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/record) элемент.

Присваивание и деконструкция кортежей

В C# поддерживается присваивание между типами кортежей, которые соответствуют обоим следующим условиям:

оба типа кортежей должны содержать одинаковое количество элементов;

для каждой позиции кортежа тип правого элемента кортежа аналогичен типу соответствующего левого элемента кортежа или может быть неявно преобразован в этот тип.

Значения элементов кортежа присваиваются в порядке расположения элементов кортежа. Имена полей кортежа не учитываются и не присваиваются, как показано в следующем примере.

C#Копировать

Выполнить

(int, double) t1 = (17, 3.14);

(double First, double Second) t2 = (0.0, 1.0);

t2 = t1;

Console.WriteLine($"{nameof(t2)}: {t2.First} and {t2.Second}");

// Output:

// t2: 17 and 3.14

(double A, double B) t3 = (2.0, 3.0);

t3 = t2;

Console.WriteLine($"{nameof(t3)}: {t3.A} and {t3.B}");

// Output:

// t3: 17 and 3.14

Оператор присваивания = можно также использовать для деконструкции экземпляра кортежа в отдельные переменные. Это можно сделать различными способами:

Вы можете использовать ключевое слово var за пределами круглых скобок, чтобы объявить неявно типизированные переменные и позволить компилятору вывести их типы.

C#Копировать

Выполнить

var t = ("post office", 3.6);

var (destination, distance) = t;

Console.WriteLine($"Distance to {destination} is {distance} kilometers.");

// Output:

// Distance to post office is 3.6 kilometers.

Вы можете явно объявить тип каждой переменной в скобках.

C#Копировать

Выполнить

var t = ("post office", 3.6);

(string destination, double distance) = t;

Console.WriteLine($"Distance to {destination} is {distance} kilometers.");

// Output:

// Distance to post office is 3.6 kilometers.

Объявите некоторые типы явно и другие типы неявно (с var) внутри скобки:

C#Копировать

Выполнить

var t = ("post office", 3.6);

(var destination, double distance) = t;

Console.WriteLine($"Distance to {destination} is {distance} kilometers.");

// Output:

// Distance to post office is 3.6 kilometers.

Вы можете использовать существующие переменные.

C#Копировать

Выполнить

var destination = string.Empty;

var distance = 0.0;

var t = ("post office", 3.6);

(destination, distance) = t;

Console.WriteLine($"Distance to {destination} is {distance} kilometers.");

// Output:

// Distance to post office is 3.6 kilometers.

Назначение деконструкционного выражения может включать как существующие переменные, так и переменные, объявленные в объявлении деконструкции.

Вы также можете объединить деконструкцию с сопоставлением[шаблонов,](https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/functional/pattern-matching)чтобы проверить характеристики полей в кортеже. В следующем примере выполняется цикл по нескольким целым числам и выводится число, которое делится на 3. Он деконструирует результат [Int32.DivRem](https://learn.microsoft.com/ru-ru/dotnet/api/system.int32.divrem) кортежа и соответствует Remainder 0:

C#Копировать

Выполнить

for (int i = 4; i < 20; i++)

{

if (Math.DivRem(i, 3) is ( Quotient: var q, Remainder: 0 ))

{

Console.WriteLine($"{i} is divisible by 3, with quotient {q}");

}

}

Подробнее о деконструкции кортежей с помощью и других типов см. в статье [Деконструкция кортежей и других типов](https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/functional/deconstruct).

Равенство кортежей

Типы кортежей == поддерживают операторы и != операторы. Эти операторы сравнивают элементы левого операнда с соответствующими элементами правого операнда в соответствии с порядком расположения элементов кортежа.

C#Копировать

Выполнить

(int a, byte b) left = (5, 10);

(long a, int b) right = (5, 10);

Console.WriteLine(left == right); // output: True

Console.WriteLine(left != right); // output: False

var t1 = (A: 5, B: 10);

var t2 = (B: 5, A: 10);

Console.WriteLine(t1 == t2); // output: True

Console.WriteLine(t1 != t2); // output: False

Как показано в предыдущем примере, в операциях == и != не учитываются имена полей кортежей.

Два кортежа сравнимы, если выполнены оба следующих условия:

оба кортежа содержат одинаковое количество элементов. Например, t1 != t2 не компилируется, если t1 и t2 имеют разное количество элементов.

Для каждой позиции кортежа соответствующие элементы из левого и правого операндов кортежа сравниваются с помощью операторов == и !=. Например, не компилируется, (1, (2, 3)) == ((1, 2), 3) так как 1 не сравнивается с (1, 2).

Операторы == и != сравнивают кортежи с сокращенной обработкой. Это значит, что операция останавливается, как только она соответствует паре неравных элементов или достигает конца кортежей. Однако перед любым сравнением все элементы кортежа вычисляются, как показано в следующем примере.

C#Копировать

Выполнить

Console.WriteLine((Display(1), Display(2)) == (Display(3), Display(4)));

int Display(int s)

{

Console.WriteLine(s);

return s;

}

// Output:

// 1

// 2

// 3

// 4

// False

Кортежи как параметры вывода

Как правило, вы выполняете рефакторинг метода, имеющего [параметры out](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/method-parameters#out-parameter-modifier), в метод, возвращающий кортеж. Однако бывают случаи, когда параметр out может иметь тип кортежа. В следующем примере показано, как работать с кортежами в виде параметров out.

C#Копировать

Выполнить

var limitsLookup = new Dictionary<int, (int Min, int Max)>()

{

[2] = (4, 10),

[4] = (10, 20),

[6] = (0, 23)

};

if (limitsLookup.TryGetValue(4, out (int Min, int Max) limits))

{

Console.WriteLine($"Found limits: min is {limits.Min}, max is {limits.Max}");

}

// Output:

// Found limits: min is 10, max is 20

Кортежи и System.Tuple

Кортежи C# с типами [System.ValueTuple](https://learn.microsoft.com/ru-ru/dotnet/api/system.valuetuple), отличаются от кортежей, представленных типами [System.Tuple](https://learn.microsoft.com/ru-ru/dotnet/api/system.tuple). Основные различия заключаются в следующем.

Типы System.ValueTuple являются [типами значений](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types). Типы System.Tuple являются [ссылочными типами](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/reference-types).

Типы System.ValueTuple являются изменяемыми. Типы System.Tuple являются неизменяемыми.

Элементами данных типов System.ValueTuple являются поля. Элементами данных типов System.Tuple являются свойства.

# Узнайте, как управлять коллекциями данных с помощью List<T> в C#

Это вводное руководство содержит общие сведения о языке C# и классе [List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1).

Предварительные требования

Для работы с руководством вам потребуется компьютер, настроенный для разработки в локальной среде. Инструкции по установке и общие сведения о разработке приложений в .NET см. в статье [Настройка локальной среды](https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/local-environment) .

Если вы предпочитаете выполнять код без настройки локальной среды, ознакомьтесь с [интерактивной версией этого руководства в браузере](https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/list-collection).

Пример простого списка

Создайте каталог с именем list-tutorial. Откройте этот каталог и выполните команду dotnet new console.

**Важно!**

В шаблонах C# для .NET 6 используются операторы верхнего уровня. Приложение может не соответствовать коду, приведенному в этой статье, если вы уже выполнили обновление до .NET 6. Дополнительные сведения см. в статье [**Новые шаблоны C# для создания инструкций верхнего уровня**](https://learn.microsoft.com/ru-ru/dotnet/core/tutorials/top-level-templates).

Пакет SDK для .NET 6 также добавляет набор неявныхglobal using директив для проектов, использующих следующие пакеты SDK:

Microsoft.NET.Sdk

Microsoft.NET.Sdk.Web

Microsoft.NET.Sdk.Worker

Эти неявные директивы global using включают наиболее распространенные пространства имен для соответствующего типа проектов.

Дополнительные сведения см. в статье [**о директивах неявного использования**](https://learn.microsoft.com/ru-ru/dotnet/core/project-sdk/overview#implicit-using-directives).

Откройте Program.cs в любом редакторе и замените существующий код следующим:

C#Копировать

List<string> names = ["<name>", "Ana", "Felipe"];

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

Замените <name> собственным именем. Сохраните Program.cs. Введите в окне консоли команду dotnet run для тестирования.

Вы создали список строк, добавили три имени в этот список и вывели имена прописными буквами. Для циклического прохода по списку вы примените концепции, которые изучили в предыдущих руководствах.

В коде для отображения имен используется функция [интерполяции строк](https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/tokens/interpolated). Если перед string добавить символ $, код C# можно внедрять в объявление строки. Фактическая строка заменяет код C# генерируемым значением. В этом примере она заменяет {name.ToUpper()} именами, буквы каждого из которых преобразованы в прописные, так как вызван метод [ToUpper](https://learn.microsoft.com/ru-ru/dotnet/api/system.string.toupper).

Продолжим изучение.

Изменение содержимого списка

В созданной коллекции используется тип [List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1). При применении такого типа сохраняются последовательности элементов. Тип элементов указывается в угловых скобках.

Важный аспект типа [List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1) — возможность увеличения или уменьшения, что позволяет добавлять или удалять элементы. Добавьте этот код в конец программы:

C#Копировать

Console.WriteLine();

names.Add("Maria");

names.Add("Bill");

names.Remove("Ana");

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

В конец списка добавлены еще два имени. При этом одно имя удалено. Сохраните файл и введите dotnet run для тестирования.

[List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1) позволяет добавлять ссылки на отдельные элементы по **индексу**. Поместите индекс между токенами [ и ] после имени списка. Для первого индекса в C# используется значение 0. Добавьте следующий код сразу после добавленного фрагмента и протестируйте его:

C#Копировать

Console.WriteLine($"My name is {names[0]}");

Console.WriteLine($"I've added {names[2]} and {names[3]} to the list");

Доступ к индексу за пределами списка получить невозможно. Помните, что индексы начинаются с 0, поэтому максимальный допустимый индекс меньше, чем число элементов в списке. Вы можете проверить, как долго в списке используется свойство [Count](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1.count). Добавьте следующий код в конец программы:

C#Копировать

Console.WriteLine($"The list has {names.Count} people in it");

Сохраните файл и еще раз введите dotnet run, чтобы просмотреть результаты.

Поиск по спискам и их сортировка

В наших примерах используются сравнительно небольшие списки. Но приложения часто создают списки с гораздо большим количеством элементов, иногда они исчисляются в тысячах. Чтобы найти элементы в таких больших коллекциях, необходимо выполнить поиск различных элементов по списку. Метод [IndexOf](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1.indexof) выполняет поиск элемента и возвращает его индекс. Если элемент отсутствует в списке, IndexOf возвращает -1. Добавьте этот код в конец программы:

C#Копировать

var index = names.IndexOf("Felipe");

if (index == -1)

{

Console.WriteLine($"When an item is not found, IndexOf returns {index}");

}

else

{

Console.WriteLine($"The name {names[index]} is at index {index}");

}

index = names.IndexOf("Not Found");

if (index == -1)

{

Console.WriteLine($"When an item is not found, IndexOf returns {index}");

}

else

{

Console.WriteLine($"The name {names[index]} is at index {index}");

}

Кроме того, можно сортировать элементы в списке. Метод [Sort](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1.sort) сортирует все элементы списка в обычном порядке (строки — в алфавитном). Добавьте этот код в конец программы:

C#Копировать

names.Sort();

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

Сохраните файл и введите dotnet run, чтобы протестировать последнюю версию.

Прежде чем перейти к следующему разделу, переместим текущий код в отдельный метод. Это упростит начало работы с новым примером. Поместите весь написанный код в новый метод с именем WorkWithStrings(). Вызовите этот метод в начале кода программы. В результате ваш код должен выглядеть примерно следующим образом:

C#Копировать

WorkWithStrings();

void WorkWithStrings()

{

List<string> names = ["<name>", "Ana", "Felipe"];

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

Console.WriteLine();

names.Add("Maria");

names.Add("Bill");

names.Remove("Ana");

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

Console.WriteLine($"My name is {names[0]}");

Console.WriteLine($"I've added {names[2]} and {names[3]} to the list");

Console.WriteLine($"The list has {names.Count} people in it");

var index = names.IndexOf("Felipe");

if (index == -1)

{

Console.WriteLine($"When an item is not found, IndexOf returns {index}");

}

else

{

Console.WriteLine($"The name {names[index]} is at index {index}");

}

index = names.IndexOf("Not Found");

if (index == -1)

{

Console.WriteLine($"When an item is not found, IndexOf returns {index}");

}

else

{

Console.WriteLine($"The name {names[index]} is at index {index}");

}

names.Sort();

foreach (var name in names)

{

Console.WriteLine($"Hello {name.ToUpper()}!");

}

}

Списки других типов

Вы уже использовали в списках тип string. Создадим [List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1) с использованием другого типа. Сначала создадим набор чисел.

Добавьте следующий код в программу после вызова WorkWithStrings():

C#Копировать

List<int> fibonacciNumbers = [1, 1];

Будет создан список целых чисел. Для первых двух целых чисел будет задано значение 1. Это два первых значения последовательности Фибоначчи. Каждое следующее число Фибоначчи — это сумма двух предыдущих чисел. Добавьте этот код:

C#Копировать

var previous = fibonacciNumbers[fibonacciNumbers.Count - 1];

var previous2 = fibonacciNumbers[fibonacciNumbers.Count - 2];

fibonacciNumbers.Add(previous + previous2);

foreach (var item in fibonacciNumbers)

{

Console.WriteLine(item);

}

Сохраните файл и введите dotnet run, чтобы просмотреть результаты.

Бонус в виде 2х книг и/или обучающих материалов:

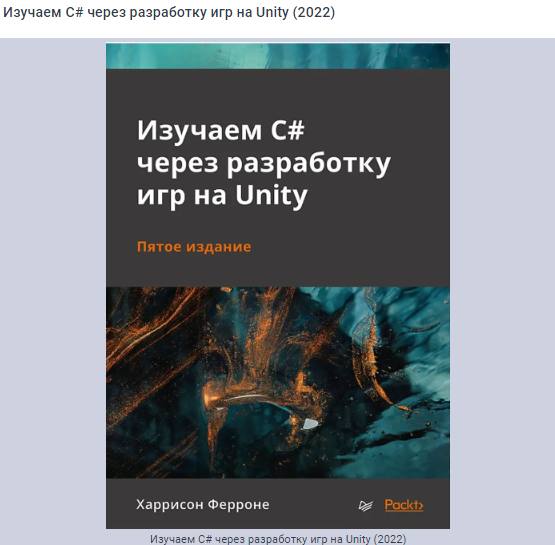
<https://proglib.io/p/top-10-knig-po-c-v-2023-godu-ot-novichka-do-professionala-2023-08-09>



***Автор:****Алексей Васильев*

*Этот учебник, созданный Алексеем Васильевым, популярным автором российских пособий по программированию, познакомит читателей с основами языка программирования C# на базовом уровне. В нем представлена история развития языка, его структура, а также изложены основные типы данных и переменные, операторы, циклы и множество другой важной информации, необходимой для работы с этим языком. Помимо этого, в книге представлены примеры кода с разъяснениями автора, что позволяет лучше усвоить материал.*

*Данный учебник отлично подходит для начинающих программистов, желающих освоить C# самостоятельно.*



***Автор:****Харрисон Фероне*

*Это пособие по изучению программирования на C# с нуля, использующее в качестве отправной точки игровой движок Unity.*

*Книга научит вас основам ООП и базовым структурам рассматриваемого языка при помощи огромного количества примеров кода, упражнений и советов. Вы поймете, как писать сценарии на C# для простой игровой механики, выполнять процедурное программирование и усложнять свои игры, добавляя интеллектуальных врагов и снаряды, наносящие урон. Вы также научитесь делить свой код на подключаемые стандартные блоки, используя интерфейсы, абстрактные классы и их расширения.*

*Книга подойдет любому, кто хочет начать работу с программированием на Unity и C#.*