Type Method

ramp(withInitialValue:increment:count:)

Returns a double-precision vector that contains monotonically incrementing or decrementing values using an initial value and increment.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst | macOS 10.15+ | tvOS 13.0+ | visionOS | watchOS 6.0+

```
static func ramp(
    withInitialValue initialValue: Double,
    increment: Double,
    count: Int
) -> [Double]
```

Parameters

`initialValue`
The initial value of the ramp.

`increment`
The increment, or decrement if negative, between each generated element.

`count`
The number of elements in the ramp.

Discussion

Use this function to generate and return a vector populated with ramped values.

The following code generates a ramped vector with values in the range `0 ... 7`:

```
let ramp = vDSP.ramp(withInitialValue: Double(0),
                     increment: 1,
                     count: 8)

// Prints "[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0]".
print(ramp)
```

See Also

Type Methods

`static func absolute<U>(U) -> [Double]`

Returns the absolute value of each element in the supplied double-precision vector.

`static func absolute<U>(U) -> [Float]`

Returns the absolute value of each element in the supplied single-precision vector.

`static func absolute<V>(DSPSplitComplex, result: inout V)`

Calculates the absolute value of each element in the supplied single-precision complex vector.

`static func absolute<V>(DSPDoubleSplitComplex, result: inout V)`

Calculates the absolute value of each element in the supplied double-precision complex vector.

`static func absolute<U, V>(U, result: inout V)`

Calculates the absolute value of each element in the supplied double-precision vector.

`static func absolute<U, V>(U, result: inout V)`

Calculates the absolute value of each element in the supplied single-precision vector.

`static func add<U>(Double, U) -> [Double]`

Returns the double-precision element-wise sum of a vector and a scalar value.

`static func add<T, U>(T, U) -> [Double]`

Returns the double-precision element-wise sum of two vectors.

`static func add<U>(Float, U) -> [Float]`

Returns the single-precision element-wise sum of a vector and a scalar value.

```
static func add<T, U>(T, U) -> [Float]
```

Returns the single-precision element-wise sum of two vectors.

```
static func add<U, V>(Double, U, result: inout V)
```

Calculates the single-precision element-wise sum of a vector and a scalar value.

```
static func add<U, V>(Float, U, result: inout V)
```

Calculates the single-precision element-wise sum of a vector and a scalar value.

```
static func add<T, U, V>(T, U, result: inout V)
```

Calculates the double-precision element-wise sum of two vectors.

```
static func add<T, U, V>(T, U, result: inout V)
```

Calculates the single-precision element-wise sum of two vectors.

```
static func add(DSPSplitComplex, to: DSPSplitComplex, count: Int,
result: inout DSPSplitComplex)
```

Calculates the single-precision elementwise sum of the supplied complex vectors.