Structure

Decimal

A structure representing a base-10 number.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.0+ | macOS 10.0+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
struct Decimal
```

Topics

Creating an empty decimal

`init()`

Creates a decimal initialized to `0`.

Creating a decimal from components

`init(sign: FloatingPointSign, exponent: Int, significand: Decimal)`

Creates a decimal initialized with the given sign, exponent, and significand.

Creating a decimal from a floating point number

`init(Double)`

Creates and initializes a decimal with the provided floating point value.

Creating a decimal from an integer

`init(Int)`

Creates and initializes a decimal with the provided integer value.

`init(Int8)`

Creates and initializes a decimal with the provided integer value.

`init(Int16)`

Creates and initializes a decimal with the provided integer value.

`init(Int32)`

Creates and initializes a decimal with the provided integer value.

`init(Int64)`

Creates and initializes a decimal with the provided integer value.

Creating a decimal from an unsigned integer

`init(UInt)`

Creates and initializes a decimal with the provided unsigned integer value.

`init(UInt8)`

Creates and initializes a decimal with the provided unsigned integer value.

`init(UInt16)`

Creates and initializes a decimal with the provided unsigned integer value.

`init(UInt32)`

Creates and initializes a decimal with the provided unsigned integer value.

`init(UInt64)`

Creates and initializes a decimal with the provided unsigned integer value.

Creating a decimal from another decimal

`init(signOf: Decimal, magnitudeOf: Decimal)`

Creates and initializes a decimal with the sign and magnitude of the given decimals.

`func NSDecimalCopy(UnsafeMutablePointer<Decimal>, UnsafePointer<Decimal>)`

Copies the value of a decimal number.

Creating a decimal by parsing a string

`init(String, format: Decimal.FormatStyle, lenient: Bool) throws`

Creates and initializes a decimal by parsing a string according to the provided format style.

`init(String, format: Decimal.FormatStyle.Currency, lenient: Bool) throws`

Creates and initializes a decimal by parsing a string according to the provided currency format style.

`init(String, format: Decimal.FormatStyle.Percent, lenient: Bool) throws`

Creates and initializes a percentage decimal by parsing a string according to the provided format style.

`init?(string: String, locale: Locale?)`

Creates and initializes a decimal by parsing a string according to the provided locale's conventions.

`init<S>(S.ParseInput, strategy: S) throws`

Creates and initializes a decimal by parsing an arbitrary type according to the provided parse strategy.

`struct ParseStrategy`

A parse strategy for creating decimal values from formatted strings.

Performing arithmetic

`func pow(Decimal, Int) -> Decimal`

Returns a decimal number raised to a given power.

Performing arithmetic using references

`func NSDecimalCompact(UnsafeMutablePointer<Decimal>)`

Compacts the decimal structure for efficiency.

`func NSDecimalAdd(UnsafeMutablePointer<Decimal>, UnsafePointer<Decimal>, UnsafePointer<Decimal>, NSDecimalNumber.RoundingMode) -> NSDecimalNumber.CalculationError`

Adds two decimal values.

`func NSDecimalSubtract(UnsafeMutablePointer<Decimal>, UnsafePointer<Decimal>, UnsafePointer<Decimal>, NSDecimalNumber.RoundingMode) -> NSDecimalNumber.CalculationError`

Subtracts one decimal value from another.

```
func NSDecimalDivide(UnsafeMutablePointer<Decimal>, UnsafePointer<
Decimal>, UnsafePointer<Decimal>, NSDecimalNumber.RoundingMode) ->
NSDecimalNumber.CalculationError
```

Divides one decimal value by another.

```
func NSDecimalMultiply(UnsafeMutablePointer<Decimal>, UnsafePointer<
Decimal>, UnsafePointer<Decimal>, NSDecimalNumber.RoundingMode) ->
NSDecimalNumber.CalculationError
```

Multiplies two decimal numbers together.

```
func NSDecimalMultiplyByPowerOf10(UnsafeMutablePointer<Decimal>, Unsafe
Pointer<Decimal>, Int16, NSDecimalNumber.RoundingMode) -> NSDecimal
Number.CalculationError
```

Multiplies a decimal by the specified power of 10.

```
func NSDecimalRound(UnsafeMutablePointer<Decimal>, UnsafePointer<
Decimal>, Int, NSDecimalNumber.RoundingMode)
```

Rounds off the decimal value.

```
func NSDecimalPower(UnsafeMutablePointer<Decimal>, UnsafePointer<
Decimal>, Int, NSDecimalNumber.RoundingMode) -> NSDecimalNumber.
CalculationError
```

Raises the decimal value to the specified power.

```
func NSDecimalNormalize(UnsafeMutablePointer<Decimal>, UnsafeMutable
Pointer<Decimal>, NSDecimalNumber.RoundingMode) -> NSDecimalNumber.
CalculationError
```

Normalizes the internal format of two decimal numbers to simplify later operations.

```
typealias RoundingMode
```

An alias for an enumeration that specifies possible rounding modes.

```
enum RoundingMode
```

These constants specify rounding behaviors.

```
typealias CalculationError
```

An alias for a type that specifies possible calculation errors.

```
enum CalculationError
```

Calculation error constants used to describe an error in `exceptionDuringOperation(_:error:leftOperand:rightOperand:)`.

Getting a decimal's characteristics

`var sign: FloatingPointSign`

The sign of the decimal.

`var exponent: Int`

The exponent of the decimal.

`var significand: Decimal`

The significand of the decimal.

`var floatingPointClass: FloatingPointClassification`

The IEEE 754 class of this type.

`var isCanonical: Bool`

A Boolean value indicating whether the representation of this decimal is canonical.

`var isFinite: Bool`

A Boolean value indicating whether this decimal is zero, subnormal, or normal (not infinity or NaN).

`var isInfinite: Bool`

A Boolean value indicating whether this decimal is infinity.

`var isNaN: Bool`

A Boolean value indicating whether this decimal is NaN.

`var isNormal: Bool`

A Boolean value indicating whether this decimal is normal (not zero, subnormal, infinity, or NaN).

`var isSignMinus: Bool`

A Boolean value indicating whether this decimal has a negative sign.

`var isSignaling: Bool`

A Boolean value indicating whether this decimal is a signaling NaN.``

`var isSignalingNaN: Bool`

A Boolean value indicating whether this decimal is a signaling NaN.

`var isSubnormal: Bool`

A Boolean value indicating whether this decimal is subnormal.

`var isZero: Bool`

A Boolean value indicating whether this value is zero.

`var nextDown: Decimal`

The greatest representable value that is less than this decimal.

`var nextUp: Decimal`

The least representable value that is greater than this decimal.

`var ulp: Decimal`

The unit in the last place of the decimal.

Getting particular decimals

`static let greatestFiniteMagnitude: Decimal`

The decimal that contains the largest possible non-infinite magnitude for the underlying representation.

`static let leastFiniteMagnitude: Decimal`

The decimal that contains the smallest possible non-infinite magnitude for the underlying representation.

`static let leastNonzeroMagnitude: Decimal`

The decimal value that represents the smallest possible non-zero value for the underlying representation.

`static let leastNormalMagnitude: Decimal`

The decimal value that represents the smallest possible normal magnitude for the underlying representation.

`static let pi: Decimal`

The mathematical constant pi.

`static var nan: Decimal`

The value that represents “not a number.”

`static var quietNaN: Decimal`

A quiet representation of not-a-number.

`static var radix: Int`

The radix used by decimal numbers.

`var NSDecimalMaxSize: Int32`

The maximum size of `Decimal`.

`var NSDecimalNoScale: Int32`

Specifies that the number of digits allowed after the decimal separator in a decimal number should not be limited.

Formatting decimals

`func formatted() -> String`

Formats the decimal using a default localized format style.

`func formatted<S>(S) -> S.FormatOutput`

Formats the decimal using the provided format style.

`struct FormatStyle`

A structure that converts between decimal values and their textual representations.

Converting between decimals and strings

`func NSDecimalString(UnsafePointer<Decimal>, Any?) -> String`

Returns a string representation of the decimal value appropriate for the specified locale.

Comparing decimals

`func isEqual(to: Decimal) -> Bool`

Indicates whether this decimal is equal to the specified one.

`func isLess(than: Decimal) -> Bool`

Indicates whether this decimal is less than the specified one.

`func isLessThanOrEqualTo(Decimal) -> Bool`

Indicates whether this decimal is less than or equal to the specified one.

`func isTotallyOrdered(belowOrEqualTo: Decimal) -> Bool`

Returns a Boolean value indicating whether this instance should precede the given value in an ascending sort.

`func NSDecimalCompare(UnsafePointer<Decimal>, UnsafePointer<Decimal>) -> ComparisonResult`

Compares two decimal values.

Using reference types

`class NSDecimalNumber`

An object for representing and performing arithmetic on base-10 numbers.

Supporting Types

`struct FormatStyle`

A structure that converts between decimal values and their textual representations.

Operators

`static func / (Decimal, Decimal) -> Decimal`

`static func /= (inout Decimal, Decimal)`

Instance Methods

`func add(Decimal)`

`func divide(by: Decimal)`

`func multiply(by: Decimal)`

`func subtract(Decimal)`

Relationships

Conforms To

AdditiveArithmetic
Comparable
ConvertibleFromGeneratedContent
ConvertibleToGeneratedContent
Copyable

CustomStringConvertible
Decodable
Encodable
Equatable
ExpressibleByFloatLiteral
ExpressibleByIntegerLiteral
Generable
Hashable
InstructionsRepresentable
Numeric
Plottable
PromptRepresentable
Sendable
SendableMetatype
SignedNumeric
Strideable

See Also

Numbers

`@frozen struct Int`

A signed integer value type.

`@frozen struct Double`

A double-precision, floating-point value type.

`class NumberFormatter`

A formatter that converts between numeric values and their textual representations.