

# OBJECTIVE-C FOUNDATION CLASSES

## REFERENCE CARD

### Part 1: Datatypes

---

#### DBigDouble

##### Methods

- **init** ..... Init to number 0.0 in default precision
- **init** : (ulong) **point** ..... Init to number 0.0 with precision
- **init** : (double) **value** : (long) **point**
  - | Init to (small) double with precision
- **init** : (char \*) **cstring** : (int) **base** : (ulong) **point**
  - | Init with string and precision
- **deepen** ..... Deepen the copied big double
- **free** ..... Free the big double
- (BOOL) **isNegative** ..... Check for negative double
- (BOOL) **isZero** ..... Check for number 0.0
- (ulong) **point** ..... Return the current precision
- **point** : (ulong) **point** ..... Change the precision
- **clear** ..... Clear the big double to 0.0
- **set** : (double) **value** ..... Set with a (small) double
- (BOOL) **set** : (cchar \*) **cstring** : (int) **base**
  - | Set the big double with a cstring and a base
- **move** : (DBigDouble \*) **other** .... Set with another big double
- (double) **get** ..... Return as a (small) double
- (DText \*) **get** : (int) **base** : (unsigned) **digits**
  - | Return as a text string
- **add** : (DBigDouble \*) **other** .... Add with another big double
- **add** : (DBigDouble \*) **src1** : (DBigDouble \*) **src2**
  - | Add two big doubles and store the result in the object
- **sub** : (DBigDouble \*) **other** Subtract with another big double
- **sub** : (DBigDouble \*) **src1** : (DBigDouble \*) **src2**
  - | Subtract two big doubles and store the result in the object
- **mul** : (DBigDouble \*) **other** Multiply with another big double
- **mul** : (DBigDouble \*) **src1** : (DBigDouble \*) **src2**
  - | Multiply two big doubles and store the result in the object
- **div** : (DBigDouble \*) **other** .... Divide by another big double
- **div** : (DBigDouble \*) **src1** : (DBigDouble \*) **src2**
  - | Divide two big doubles and store the result in the object
- **abs** ..... Absolute the big double
- **negate** ..... Negate the big double
- **sqrt** ..... Do a square root on the big double
- **power** : (ulong) **factor** ..... Power the big double
- **ceil** ..... Ceil the big double
- **floor** ..... Floor the big double

- **trunc** ..... Truncate the big double
- (int) **compare** : (DBigDouble \*) **other**
  - | Compare with another big double
- (DText \*) **toText** ..... Convert to a (decimal) text string
- (int) **fromString** : (char \*\*) **cstr**
  - | Parse a cstring for a big double

#### DBigInt

##### Methods

- **init** ..... Init to number 0
- **init** : (long) **number** ..... Init to small number
- **init** : (char \*) **cstring** : (int) **base** ..... Init with string
- **deepen** ..... Deepen the copied object
- **free** ..... Free the big integer
- (BOOL) **isNegative** ..... Check for a negative number
- (BOOL) **isZero** ..... Check for number 0
- **clear** ..... Set big integer to 0
- **set** : (long) **number** ..... Set to small number
- **set** : (uchar \*) **number** : (ulong) **length** : (BOOL) **negative**
  - | Set with array of bytes
- (BOOL) **set** : (const char \*) **cstring** : (int) **base**
  - | Set with a cstring
- **move** : (DBigInt \*) **other** ..... Set with other big integer
- (long) **get** ..... Get as small number
- (DText \*) **get** : (int) **base** ..... Get as text string
- (DData \*) **toData** ..... Get as data string
- **add** : (DBigInt \*) **other** ..... Add with another big integer
- **add** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Add two big integers
- **sub** : (DBigInt \*) **other** ... Subtract with another big integer
- **sub** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Subtract two big integers
- **mul** : (DBigInt \*) **other** ... Multiply with another big integer
- **mul** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Multiply two big integers
- **div** : (DBigInt \*) **other** .... Divide with another big integer
- **div** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Divide two big integers
- **mod** : (DBigInt \*) **other** ... Modulo with another big integer
- **mod** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Modulo two big integers
- **abs** ..... Absolute the big integer
- **com** ..... One complements the big integer
- **negate** ..... Negate the big integer

- **and** : (DBigInt \*) **other** ..... And with another big integer
- **and** : (DBigInt \*) **src1** : (DBigInt \*) **src2** .... And two big integers
- **or** : (DBigInt \*) **other** ..... Or with another big integer
- **or** : (DBigInt \*) **src1** : (DBigInt \*) **src2** Or two big integers
- **xor** : (DBigInt \*) **other** ..... Xor with another big integer
- **xor** : (DBigInt \*) **src1** : (DBigInt \*) **src2**
  - | Xor two bit integers
- **lshift** : (ulong) **shifts** ..... Left shift the big integer
- **lshift** : (DBigInt \*) **src** : (ulong) **shifts**
  - | Left shift the source
- **rshift** : (ulong) **shifts** ..... Right shift the big integer
- **rshift** : (DBigInt \*) **src** : (ulong) **shifts**
  - | Right shift the source
- (int) **compare** : (DBigInt \*) **other**
  - | Compare with another big integer
- (DText \*) **toText** ..... Convert to a decimal text string
- (int) **fromString** : (char \*\*) **cstr**
  - | Parse a cstring for a big integer

#### DBitArray

##### Methods

- **init** ..... Init default bit array [0..255]
- **init** : (int) **min** : (int) **max** ..... Init bit array [min..max]
- **copy** ..... Copy the bit array
- **free** ..... Free the bit array
- (int) **min** ..... Return the minimum value in the array
- (int) **max** ..... Return the maximum value in the array
- **reset** ..... Reset all values in bit array
- **set** : (int) **val** ..... Set a value in array
- **reset** : (int) **val** ..... Reset a value in array
- **set** : (int) **from** : (int) **to** ..... Set from..to in array
- **reset** : (int) **from** : (int) **to** ..... Reset from..to in array
- **set** : (int) **from** : (int) **to** : (unsigned) **step**
  - | Set from..to in steps in array
- **reset** : (int) **from** : (int) **to** : (unsigned) **step**
  - | Reset from..to in steps in array
- (BOOL) **has** : (int) **val** ..... Test if value is set in array
- (int) **count** ..... Count number values set in array

#### DBool

##### Methods

- **init** ..... Init to false

```
- init :(BOOL) state.....Init to state
- (BOOL) get ..... Get the state
- set :(BOOL) state.....Set the state
- (int) compare :(DBool *) other
    | Compare two bool objects
- (int) fromString :(char **) cstr
    | Parse a string for a boolean state
- (DText *) toText ..... Convert to text string
- (DData *) toData ..... Convert to data string
```

**DColor**

*Constants*

DCLR\_BLACK ..... Black text color  
DCLR\_RED ..... Red text color  
DCLR\_GREEN ..... Green text color  
DCLR\_YELLOW ..... Yellow text color  
DCLR\_BLUE ..... Blue text color  
DCLR\_MAGENTA ..... Magenta text color  
DCLR\_CYAN ..... Cyan text color  
DCLR\_WHITE ..... White text color

*Publicmembers*

unsigned char \_red.....the red factor  
unsigned char \_green.....the green factor  
unsigned char \_blue.....the blue factor  
unsigned char \_alpha.....the alpha factor  
int \_text ..... the text color

*Methods*

```
- init.....Init to solid black color
- init :(char *) name.....Init to named color
- init :(uchar) red :(uchar) blue :(uchar) green
    | Init to solid rgb color
- init :(uchar) red :(uchar) blue :(uchar) green
    |:(uchar) alpha ..... Init to transparent rgb color
- (uchar) red .....Return the red factor
- (uchar) green .....Return the green factor
- (uchar) blue .....Return the blue factor
- (uchar) alpha .....Return the alpha factor
- alpha :(uchar) alpha ..... Set the alpha factor
- (int) textColor .....Return the text color
- textColor :(int) color ..... Set the text color
- (BOOL) set :(char *) name .....Set to a named color
- set :(uchar) red :(uchar) green :(uchar) blue
    | Set to a rgb color
- set :(uchar) red :(uchar) green :(uchar) blue
```

```
    |:(uchar) alpha .....Set to a transparent rgb color
- set :(uchar) red :(uchar) green :(uchar) blue
    |:(uchar) alpha :(int) color .....Set all color fields
- lighter :(double) factor .....Make color lighter or darker
- blend :(uchar) red :(uchar) green :(uchar) blue
    |:(uchar) alpha ..... Alpha blend with second color
- toRGB :(double *) red :(double *) green
    |:(double *) blue .....Convert object to RGB color
- fromRGB :(double) red :(double) green :(double) blue
    | Set object with RGB color
- toYIQ :(double *) Y :(double *) I :(double *) Q
    | Convert object ot YIQ color
- fromYIQ :(double) Y :(double) I :(double) Q
    | Set object with YIQ color
- toHLS :(double *) H :(double *) L :(double *) S
    | Convert object to HLS color
- fromHLS :(double) H :(double) L :(double) S
    | Set object with HLS color
- toHSV :(double *) H :(double *) S :(double *) V
    | Convert object to HSV color
- fromHSV :(double) H :(double) S :(double) V
    | Set object with HSV color
- toCMY :(double *) C :(double *) M :(double *) Y
    | Convert object to CMY color
- fromCMY :(double) C :(double) M :(double) Y
    | Set object with CMY color
- (DText *) toText .....Convert object to a text object
- (int) fromString :(char **) cstr .... Parse string for color
```

**DComplex**

*Methods*

```
- init ..... Init to complex number zero
- init :(double) re :(double) im ..... Init complex number
- (double) re .....Return real part
- re :(double) re .....Set read part
- (double) im ..... Return imaginary part
- im :(double) im .....Set imaginary part
- set :(double) re :(double) im .Set real and imaginary part
- move :(DComplex *) otherAssign complex number from other
- add :(DComplex *) other .....Add with complex number
- add :(DComplex *) s1 :(DComplex *) s2
    | Add two complex numbers
- sub :(DComplex *) other .....Subtract with complex number
- sub :(DComplex *) s1 :(DComplex *) s2
```

```
    | Subtract two complex number
- mul :(DComplex *) other .... Multiply with complex number
- mul :(DComplex *) s1 :(DComplex *) s2
    | Multiply two complex numbers
- rmul :(double) re .....Multiply complex with real number
- imul :(double) im .Multiply complex with imaginary number
- div :(DComplex *) other .....Divide with complex number
- div :(DComplex *) s1 :(DComplex *) s2
    | Divide two complex numbers
- cng .....Conjugate
- (double) abs ..... Modulus
- (double) nrm ..... Square modulus
- sqrt ..... Square root
- exp ..... Exponent
- log ..... Natural logarithm
- sin ..... Trigonometric sine
- cos ..... Trigonometric consine
- tan ..... Trigonometric trangent
- asin ..... Inverse trigonometric sine
- acos ..... Inverse trigonometric cosine
- atan ..... Inverse trigonometric trangent
- sinh ..... Hyperbolic sine
- cosh ..... Hyperbolic cosine
- tanh ..... Hyperbolic tangent
- asinh ..... Inverse hyperbolic sine
- acosh ..... Inverse hyperbolic cosine
- atanh ..... Inverse hyperbolic tangent
- (DText *) toText .....Convert complex number to (new)
    | text string
```

**DData**

*Methods*

```
- init .....Init empty data string
- init :(uchar *) data :(ulong) len ... Init string with data
- copy ..... Copy data string
- free .....Free the data string
- size :(ulong) size .....Insure the size of data string
- extra :(unsigned) extra .....Set extra size during resizing
- (DText *) tohexString ..... Convert to new hex text string
- (ulong) hash ..... Calculate hash from data string
- (DText *) toText ..... Convert to text object
- (DText *) toBase64 .....Convert to base64
- (DText *) toPrintable ..... Convert to printable text object
- clear .....Clear the data string
```

- set : (uchar \*) data : (ulong) len ..... Set with data
- set : (uchar \*) data : (long) from : (long) to
  - | Set with substring of data
- fromBase64 : (char \*) cstring ..... Convert from base64
- put : (long) index : (uchar) byte ... Set byte in data string
- (uchar) get : (long) index ..... Get byte from data string
- delete : (long) index ..... Remove byte from data string
- insert : (long) fr : (long) to : (uchar \*) data : (ulong) len
  - | Insert data in part of data string
- (DData \*) get : (long) from : (long) to
  - | Return new sub data string
- delete : (long) from : (long) to
  - | Delete part of data string
- (BOOL) isEmpty ..... Test for empty data string
- (ulong) length ..... Return length of data string
- (uchar \*) data ..... Return pointer to data string
- (ulong) size ..... Return the size of the data string
- (int) error ..... Return the current error (or 0)
- (DData \*) toData ..... Convert data to new DData object
- (DText \*) readText : (ulong) len
  - | Read length text from data string
- (DData \*) readData : (ulong) len
  - | Read length data from data string
- (char) readChar ..... Read character from data string
- (uchar) readByte ..... Read byte from data string
- (short) readShort ..... Read short from data string
- (long) readLong ..... Read long from data string
- (double) readDouble ..... Read double from data string
- (BOOL) isEof ..... Test for end position in string
- (DText \*) scanText : (char) sep .. Scan text until separator
- (DText \*) scanText : (char \*) seps : (char \*) sep
  - | Scan text until one of seps
- (BOOL) cmatch : (char \*) cstr . Match with string, case sens.
- (BOOL) imatch : (char \*) cstr Match with string, case insens.
- (int) scanInt : (int) wrong ..... Scan text for int
- (int) skipChar : (char) ch ..... Skip character
- (int) skipWhiteSpace ..... Skip whitespace
- (BOOL) writeText : (char \*) text .. Write text in data string
- (BOOL) writeData : (uchar \*) text : (ulong) len
  - | Write data in data string
- (BOOL) writeChar : (char) ch . Write character in data string
- (BOOL) writeByte : (uchar) byte .. Write byte in data string
- (BOOL) writeShort : (short) sh ... Write short in data string
- (BOOL) writeLong : (long) sh ..... Write long in data string
- (BOOL) writeDouble : (double) sh .... Write double in string

- (ulong) tell ..... Return current position in data string
- seek : (ulong) off : (int) origin
  - | Move position in data string
- skip : (ulong) off ..... Skip position in data string
- append : (uchar \*) data : (ulong) len Append data to string
- prepend : (uchar \*) data : (ulong) len
  - | Prepend data to string
- push : (uchar) ch ..... Push a byte behind the data string
- (uchar) pop ..... Pull a byte from the data string
- multiply : (unsigned) times ... Repeat the data in the string
- (int) compare : (DData \*) obj Compare data string with obj
- (int) bcompare : (uchar \*) data : (ulong) len
  - | Binairy compare data string with data (-1,0,1)
- (ulong) count : (uchar \*) srch : (ulong) len : (long) from
  - | : (long) to ..... Count 'srch' occurences in data string
- (long) index : (uchar \*) srch : (ulong) len : (long) from
  - | : (long) to ..... Return first index where 'srch' is found
- (long) rindex : (uchar \*) srch : (ulong) len : (long) from
  - | : (long) to ..... Return last index where 'srch' is found
- replace : (uchar \*) old : (ulong) olen : (uchar \*) new
  - | : (ulong) nlen : (long) max
  - | Replace old with new in data string, max times

## DDateTime

### Constants

DDT\_SUNDAY ..... Weekday sunday  
 DDT\_MONDAY ..... Weekday monday  
 DDT\_TUESDAY ..... Weekday tuesday  
 DDT\_WEDNESDAY ..... Weekday wednesday  
 DDT\_THURSDAY ..... Weekday thursday  
 DDT\_FRIDAY ..... Weekday friday  
 DDT\_SATURDAY ..... Weekday saturday  
 DDT\_MIN\_YEAR ..... Minimum value for year

### ClassMethods

- + (BOOL) isLeapYear : (int) year ..... Check for leap year
- + (int) daysInMonth : (int) year : (int) month
  - | Return number of days in month
- + (BOOL) isValidDate : (int) year : (int) month : (int) day
  - | Check if date is valid
- + (BOOL) isValidTime : (int) hours : (int) minutes
  - | : (int) seconds : (int) millis ..... Check if time is valid
- + (BOOL) isValid : (int) year : (int) month : (int) day
  - | : (int) hours : (int) minutes : (int) seconds
  - | : (int) millis ..... Check if date and time is valid

### ObjectMethods

- init ..... Init empty date/time
- init : (int) year : (int) month : (int) day
  - | : (int) hours : (int) minutes : (int) seconds
  - | Init with date/time
- copy ..... Copy the object
- (int) year ..... Return year
- (int) month ..... Return month
- (int) day ..... Return day
- (int) hours ..... Return hours
- (int) minutes ..... Return minutes
- (int) seconds ..... Return seconds
- (int) millis ..... Return milliseconds
- (int) weekday ..... Return the day of the week
- (BOOL) set : (int) year : (int) month : (int) day
  - | : (int) hours : (int) minutes : (int) seconds
  - | Set the date/time
- (BOOL) set : (int) year : (int) month : (int) day
  - | : (int) hours : (int) minutes : (int) seconds
  - | : (int) millis ..... Set date/time with milliseconds
- (BOOL) time : (int) hours : (int) minutes : (int) seconds
  - | Set the time
- (BOOL) time : (int) hours : (int) minutes : (int) seconds
  - | : (int) millis ..... Set the time with milliseconds
- (BOOL) date : (int) year : (int) month : (int) day
  - | Set the date
- (BOOL) localTime ..... Set with current local time
- (BOOL) UTCTime ..... Set with current UTC time
- (BOOL) norm ..... Normalize the date/time
- (DText \*) toISO8601 ..... Format to ISO8601
- (DText \*) toRFC1123 ..... Format to RFC1123
- (DText \*) toRFC850 ..... Format to RFC850
- (DText \*) toRFC822 ..... Format to RFC822
- (DText \*) toASC ..... Format as asctime
- (DText \*) format : (char \*) format .... Format as strftime
- (DText \*) toText ..... Convert to text object
- (int) fromString : (char \*\*) cstr ..... Parse from string
- (int) date : (char \*\*) cstr ..... Parse date from string
- (int) time : (char \*\*) cstr ..... Parse time from string
- (BOOL) parse : (char \*\*) cstr : (char \*) format
  - | Parse accordingly the format
- (BOOL) fromRFC1123 : (char \*\*) cstr
  - | Parse accordingly RFC1123
- (BOOL) fromRFC850 : (char \*\*) cstr
  - | Parse accordingly RFC850

- (BOOL) fromRFC822 :(char \*\*) cstr
  - └ Parse accordingly RFC822
- (BOOL) fromASC :(char \*\*) cstr . Parse from asctime format
- (int) compare :(DDateTime \*) other
  - └ Compare two date/times

## DDouble

### Methods

- init.....Init to zero double
- init :(double) number ..... Init to double number
- (double) get ..... Return the double
- set :(double) number.....Set the double number
- (int) compare :(DDouble \*) other Compare to other double
- (DText \*) toText ..... Convert to new text string
- (DData \*) toData ..... Convert to new data string
- (int) fromString :(char \*\*) cstr...Set double from string

## DDoubleArray

### Methods

- init.....Init to an empty array
- init :(const double \*) doubles :(ulong) length.Init with array
- deepen.....Deepen the copied object
- free.....Free the object
- (BOOL) isEmpty.....Check if the array is empty
- (ulong) length.....Return the length of the array
- (const double \*) array.....Return the array
- size :(ulong) size.....Insure the array size
- extra :(unsigned) extra .... Set the extra size during resize
- (DText \*) toText ..... Convert to text string
- (int) fromString :(char \*\*) cstr....Set array from string
- clear ..... Clear the array
- set :(double \*) doubles :(ulong) length...Set with array
- put :(long) index :(double) value ..... Put value in array
- (double) get :(long) index.....Get value from array
- insert :(long) index :(double) value.Insert value in array
- delete :(long) index.....Delete value from array
- insert :(long) from :(long) to :(double \*) doubles
  - └:(ulong) length ..... Insert array in part of array
- (DDoubleArray \*) get :(long) from :(long) to
  - └ Return sub array
- delete :(long) from :(long) to.....Delete range in array
- append :(double \*) doubles :(ulong) length

- └ Append an array
- prepend :(double \*) doubles :(ulong) length
  - └ Prepend an array
- push :(double) value.....Push value at end of array
- (double) pop ..... Pop value from end of array
- (double) tos.....Return value at end of array
- (BOOL) enqueue :(double) value . Put value at start of array
- (double) dequeue ..... Pop value from end of array
- (int) compare :(DDoubleArray \*) other
  - └ Compare with another array object
- (int) bcompare :(const double \*) doubles
  - └:(ulong) length ..... Compare with an array
- (ulong) count :(double) search :(long) from :(long) to
  - └ Count search in array
- (long) index :(double) search :(long) from :(long) to
  - └ Find smallest index for search
- (long) rindex :(double) search :(long) from :(long) to
  - └ Find biggest index for seach
- (double) sum :(long) from :(long) to.....Calculate sum
- (double) max :(long) from :(long) toDetermine max value
- (double) min :(long) from :(long) to Determine min value
- (double) average :(long) from :(long) to
  - └ Calculate average
- (double) variance :(long) from :(long) to
  - └ Calculate variance
- (double) standardDeviation :(long) from :(long) to
  - └ Calculate standard deviation
- sort :(long) from :(long) to..Sort the array (low to high)
- invert :(long) from :(long) to...Invert (mirror) the array

## DFile

### Classmethods

- + (int) error ..... Return the last error (for class methods)
- + (BOOL) move :(char \*) path :(char \*) newPath
  - └ Move/Rename a file
- + (BOOL) remove :(char \*) path.....Remove a file
- + (BOOL) isFile :(char \*) path.....Check for file
- + (BOOL) isDirectory :(char \*) path .... Check for directory
- + (long long) size :(char \*) path.....Return file size
- + (DDateTime \*) modified :(char \*) path
  - └ Return last modified date/time
- + (DDateTime \*) accessed :(char \*) path
  - └ Return last accessed date/time

### Objectmethods

- init.....Init to empty file object
- init :(char \*) name :(char \*) mode.....Open file
- free.....Free the object (close the file)
- (int) error.....Return the last error
- (int) fileno.....Return file descriptor
- (BOOL) isAtty ..... Check for terminal
- (BOOL) isOpen ..... Check for open file
- (BOOL) open :(char \*) name :(char \*) mode .... Open file
- (BOOL) isEof.....Check for end-of-file
- (char) readChar ..... Read a character
- (DText \*) readLine ..... Read a line
- (DText \*) readText .....Read all text
- (DText \*) readText :(long) len ..... Read len text
- (BOOL) seek :(ulong) off :(int) org.....Move position
- (BOOL) skip :(ulong) off ..... Skip forward
- (unsigned long) tell ..... Return current position
- (BOOL) writeChar :(char) ch ..... Write character
- (BOOL) writeLine :(char \*) text ..... Write line
- (BOOL) writeText :(char \*) text.....Write text
- (uchar) readByte.....Read a byte
- (DData \*) readData :(ulong) length
  - └ Read a data string
- (double) readDouble.....Read a double
- (long) readLong.....Read a long
- (short) readShort .....Read a short
- (BOOL) writeByte :(uchar) byte.....Write a byte
- (BOOL) writeData :(uchar \*) text :(ulong) length
  - └ Write a data string
- (BOOL) writeDouble :(double) nr ..... Write a double
- (BOOL) writeLong :(long) nr ..... Write a long
- (BOOL) writeShort :(short) nr ..... Write a short
- (DList \*) readLines.....Read all lines in a list
- (BOOL) writeLines :(DList \*) list ..... Write list to file
- (BOOL) flush ..... Flush the output buffers
- (BOOL) truncate :(long) size .....Truncate file
- close ..... Close the file

## DFixedPoint

### Constants

DFP\_MAX\_POINT.....Maximum value for point

### Methods

- init.....Init to fixed point number (FPN) zero
- init :(unsigned) point..Init fixed point number with point
- init :(long) value :(unsigned) point.....Init FPN

- (unsigned) point ..... Return number of bits for point
- point :(unsigned) point
  - | Change the point, change the precision
- set :(long) value .... Set the FPN (using the current point)
- set :(long) value :(unsigned) point ..... Set the FPN
- (long) get ..... Return the fixed point number
- move :(DFixedPoint \*) other .. Set FPN from another object
- norm ..... Normalize the fixed point number
- add :(DFixedPoint \*) other ..... Add with another FPN
- add :(DFixedPoint \*) src1 :(DFixedPoint \*) src2
  - | Add two FPNs
- sub :(DFixedPoint \*) other .... Subtract with another FPN
- sub :(DFixedPoint \*) src1 :(DFixedPoint \*) src2
  - | Subtract two FPNs
- mul :(DFixedPoint \*) other .... Multiply with another FPN
- mul :(DFixedPoint \*) src1 :(DFixedPoint \*) src2
  - | Multiply two FPNs
- div :(DFixedPoint \*) other ..... Divide with another FPN
- div :(DFixedPoint \*) src1 :(DFixedPoint \*) src2
  - | Divide two FPNs
- (DText \*) toText ..... Convert object to text string
- (int) compare :(DFixedPoint \*) other
  - | Compare with another FPN
- (double) toDouble ..... Convert to a double

## DFraction

### ClassMethods

- (int) gcd :(int) a :(int) b .... Greatest Common Divider
- (int) lcm :(int) a :(int) b .... Least Common Multiplier

### ObjectMethods

- init ..... Init zero fraction
- init :(int) num :(int) denom ..... Init fraction
- (int) denominator ..... Return the denominator
- denominator :(int) denom ..... Set the denominator
- (int) numerator ..... Return the numerator
- numerator :(int) num ..... Set the numerator
- set :(int) num :(int) denom ..... Set the fraction
- move :(DFraction \*) other ..... Set with other fraction
- add :(DFraction \*) other ..... Add fractions
- add :(DFraction \*) fr1 :(DFraction \*) fr2
  - | Add two fractions
- sub :(DFraction \*) other ..... Subtract fractions
- sub :(DFraction \*) fr1 :(DFraction \*) fr2
  - | Subtract two fractions

- mul :(DFraction \*) other ..... Multiply fractions
- mul :(DFraction \*) fr1 :(DFraction \*) fr2
  - | Multiply two fractions
- div :(DFraction \*) other ..... Divide fractions
- div :(DFraction \*) fract :(DFraction \*) div
  - | Divide two fractions
- invert ..... Invert the fraction
- norm ..... Normalize the fraction
- (DText \*) toText ..... Convert to (new) DText
- (double) toDouble ..... Convert to a double
- (int) compare :(DFraction \*) other
  - | Compare with another fraction

## DInt

### Methods

- init ..... Init to zero int
- init :(int) number ..... Init to number
- (int) get ..... Return the int
- set :(int) number ..... Set the int
- (int) compare :(DInt \*) other ..... Compare to other
- (int) fromString :(char \*\*) cstr ..... Set int from string
- (DText \*) toText ..... Convert to new text string
- (DData \*) toData ..... Convert to new data string
- (int) toBigEndian ..... Return int in big endian order
- (int) toLittleEndian ..... Return int in little endian order

## DIntArray

### Methods

- init ..... Init an empty integer array
- init :(const int \*) ints :(ulong) length . Init with array
- deepen ..... Deepen the copied object
- free ..... Free the object
- (BOOL) isEmpty ..... Check if the array is empty
- (ulong) length ..... Return the length of the array
- (const int \*) array ..... Return the array
- size :(ulong) size ..... Insure the array size
- extra :(unsigned) extra .... Set the extra size during resize
- (DText \*) toText ..... Convert to text string
- (int) fromString :(char \*\*) cstr .... Set array from string
- clear ..... Clear the array
- set :(const int \*) ints :(ulong) length ... Set with array
- put :(long) index :(int) element ..... Put value in array
- (int) get :(long) index ..... Get value from array

- insert :(long) index :(int) value ... Insert value in array
- delete :(long) index ..... Delete element from array
- insert :(long) from :(long) to :(const int \*) ints
  - | :(ulong) length ..... Insert array in part of array
- (DIntArray \*) get :(long) from :(long) to
  - | Return sub array
- delete :(long) from :(long) to ..... Delete range in array
- append :(const int \*) ints :(ulong) length
  - | Append an array
- prepend :(const int \*) ints :(ulong) length
  - | Prepend an array
- push :(int) value ..... Push value at end of array
- (int) pop ..... Pop value from end of array
- (int) tos ..... Return value at end of array
- (BOOL) enqueue :(int) value .... Put value at start of array
- (int) dequeue ..... Pop value from end of array
- (int) compare :(DIntArray \*) other
  - | Compare with another array object
- (int) bcompare :(const int \*) ints :(ulong) length
  - | Compare with an array
- (ulong) count :(int) search :(long) from :(long) to
  - | Count search in array
- (long) index :(int) search :(long) from :(long) to
  - | Find smallest index for search
- (long) rindex :(int) search :(long) from :(long) to
  - | Find biggest index for search
- (long) sum :(long) from :(long) to ..... Calculate sum
- (int) max :(long) from :(long) to ... Determine max value
- (int) min :(long) from :(long) to ... Determine min value
- (double) average :(long) from :(long) to
  - | Calculate average
- (double) variance :(long) from :(long) to
  - | Calculate variance
- (double) standardDeviation :(long) from :(long) to
  - | Calculate standard deviation
- sort :(long) from :(long) to .. Sort the array (low to high)
- invert :(long) from :(long) to ... Invert (mirror) the array

## DLong

### Methods

- init ..... Init to zero long
- init :(long) number ..... Init to number
- (long) get ..... Return the long number
- set :(long) number ..... Set the long number

- (long) compare :(DLong \*) other..Compare to other object
- (int) fromString :(char \*\*) cstr.....Set long from string
- (DText \*) toText ..... Convert to new text string
- (DData \*) toData ..... Convert to new data string
- (long) toBigEndian ..... Return long in big endian order
- (long) toLittleEndian .... Return long in little endian order

## DLRnd

### Methods

- init.....Init non-seed random generator
- init :(ulong) seed ..... Init seeded random generator
- seed :(ulong) seed ..... Set the seed for generator
- (double) nextDouble.....Generate a random double
- (double) nextDouble :(double) from :(double) to  
  - └ Generate a ranged random double
- (int) nextInt ..... Generate a random integer
- (int) nextInt :(int) from :(int) to  
  - └ Generate a ranged random integer
- (long) nextLong ..... Generate a random long
- (long) nextLong :(long) from :(long) to  
  - └ Generate a ranged random long

## DRnd

### Methods

- init.....Init non-seed random generator
- init :(ulong) seed ..... Init seeded random generator
- seed :(ulong) seed ..... Set the seed for generator
- (double) nextDouble.....Generate a random double
- (double) nextDouble :(double) from :(double) to  
  - └ Generate a ranged random double
- (int) nextInt ..... Generate a random integer
- (int) nextInt :(int) from :(int) to  
  - └ Generate a ranged random integer
- (long) nextLong ..... Generate a random long
- (long) nextLong :(long) from :(long) to  
  - └ Generate a ranged random long

## DScore

### Methods

- init ..... Init non-ranged score
- init :(double) min :(double) max ..... Init ranged score
- init :(int) value ..... Init discrete score
- (double) min.....Return the minimum range value

- (double) max ..... Return the maximum range value
- (int) length.....Return the number of values in score
- (double) sum ..... Return the sum of the values
- (double) sumSquared.....Return the squared sum
- (double) percentage ..... Return percentage in distribution
- (BOOL) range :(double) min :(double) max  
  - └ Set the range for the score
- (BOOL) range :(int) value  
  - └ Set the discrete range for the score
- (void) reset ..... Reset the length and sums
- distribution :(int) length  
  - └ Set number values in distribution
- (BOOL) update :(double) value....Feed a value in the score
- (double) mean ..... Calculate the mean of the values
- (double) average.....Calculate the average of the values
- (double) variance ..... Calculate the variance of the values
- (double) standardDeviation..Calculate the SD of the values

## DShort

### Methods

- init.....Init to zero short
- init :(short) number.....Init to number
- (short) get ..... Return the short
- set :(short) number.....Set the short to number
- (int) compare :(DShort \*) other ..... Compare with other
- (int) fromString :(char \*\*) cstr....Set short from string
- (DText \*) toText ..... Convert to new text string
- (DData \*) toData ..... Convert to new data string
- (short) toBigEndian ..... Return short in big endian order
- (short) toLittleEndian .. Return short in little endian order

## DText

### Methods

- init ..... Init an empty string
- init :(char \*) cstr.....Init with c-string
- copy.....Copy a text string
- free.....Free a text string
- size :(ulong) size.....Insure the length of string
- extra :(unsigned) extra ..... Extra space during resizing
- (int) error ..... Return the last error
- (double) toDouble ..... Convert string to double
- (int) toInt ..... Convert string to int
- (long) toLong ..... Convert string to long

- format :(char \*) fmt,.....Set the string with format
- clear ..... Clear the text string
- set :(char \*) cstr ..... Set the string with c-string
- set :(char \*) cstr :(long) from :(long) to  
  - └ Set the string with c-substring
- set :(char) ch :(long) nr ..... Set number of characters
- put :(long) index :(char) ch .....Set character in string
- (char) get :(long) index .....Get character from string
- delete :(long) index ..... Remove character from string
- insert :(long) from :(long) to :(char \*) cstr  
  - └ Insert c-substring in text string
- insert :(long) from :(long) to :(char)ch :(long)nr  
  - └ Insert number of characters in text string
- (DText \*) get :(long) from :(long) to  
  - └ Return new substring from string
- delete :(long) from :(long) to  
  - └ Delete substring from string
- (BOOL) isEmpty .....Test for empty string
- (ulong) length .....Return the length of the string
- (char \*) cstring ..... Return the c-string from the string
- (DText \*) toText .....Convert string to new DText object
- (DData \*) toData ..... Convert string to new DData object
- (DText \*) readText ..... Read all text from string
- (DText \*) readText :(long) lenght  
  - └ Read length text from string
- (DText \*) readLine .....Read text till EOL from string
- (char) readChar ..... Read character from string
- (ulong) tell ..... Return current position in string
- (BOOL) seek :(ulong) off :(int) origin  
  - └ Move position in string
- (BOOL) skip :(ulong) off .....Skip positions in string
- (BOOL) isEof ..... Test for end position in string
- (DText \*) scanText :(char) sep .. Scan text until separator
- (DText \*) scanText :(char \*) seps :(char \*) sep  
  - └ Scan text until one of seps
- (BOOL) cmatch :(char \*) cstr..Match with string, case sens.
- (BOOL) imatch :(char \*) cstrMatch with string, case insens.
- (int) scanInt :(int) wrong ..... Scan text for int
- (int) skipChar :(char) ch ..... Skip character
- (int) skipWhiteSpace.....Skip whitespace
- (BOOL) writeText :(char \*) text ..... Write text to string
- (BOOL) writeLine :(char \*) text ..... Write line to string
- (BOOL) writeChar :(char) ch ..... Write character to string
- append :(char \*)cstr.....Append c-string to string
- prepend :(char \*) cstr ..... Prepend c-string to string

- push : (char) ch ..... Push a character to string
- pop : (char) ch ..... Pop a character from string
- multiply : (unsigned) times .. Repeat the string in the string
- capitalize ..... Capitalize the string
- capwords ..... Capitalize the words in the string
- expandtabs : (unsigned) size .... Expand the tabs to spaces
- lower ..... Lowercase characters in string
- upper ..... Uppercase characters in string
- swapcase ..... Change case characters in string
- lstrip ..... Remove leading spaces in string
- rstrip ..... Remove trailing spaces in string
- strip ..... Remove leading and trailing spaces
- ljust : (unsigned) width ..... Left justify string
- rjust : (unsigned) width ..... Right justify string
- center : (unsigned) width ..... Center string
- column : (int) width ..... Put string in column
- zfill : (unsigned) width ..... Left justify string with zero's
- (int) fromString : (char \*\*) cstr .. Set string from c-string
- (int) compare : (DText \*) obj ..... Compare string with obj
- (int) ccompare : (char \*) cstr
  - | Compare string with c-string case sensitive
- (int) icompare : (char \*) cstr
  - | Compare string with c-string case insensitive
- (int) ccompare : (char \*) cstr : (ulong) len
  - | Compare string with first len chars in c-string case sensitive
- (int) icompare : (char \*) cstr : (ulong) len
  - | Compare string with first len chars in c-string case insensitive
- (ulong) count : (char \*) cstr : (long) from : (long) to
  - | Count occurrences in string
- (long) index : (char \*) cstr : (long) from : (long) to
  - | Find first index for c-string
- (long) rindex : (char \*) cstr : (long) from : (long) to
  - | Find last index for c-string
- replace : (char \*) old : (char \*) new : (long) max
  - | Replaces old with new in string, max times

## DValue

### Constants

DVL\_EMPTY ..... Value is empty, typeless

DVL\_CLASS ..... Value is a class

DVL\_OBJECT ..... Value is an object

DVL\_SEL ..... Value is a selector

DVL\_BOOL ..... Value is a boolean

DVL\_INT ..... Value is an integer

DVL\_LONG ..... Value is a long

DVL\_DOUBLE ..... Value is a double

DVL\_STRING ..... Value is a text string

### Methods

- init ..... Init to an empty value
- deepen ..... Deepen the copied value
- free ..... Free the value
- empty ..... Set the value empty, typeless
- (BOOL) isEmpty ..... Check if value is empty
- (int) type ..... Return the type of the value
- (char \*) typeString ..... Return the value type as string
- setClass : (Class) value ..... Set the value to a class
- setObject : (id) value ..... Set the value to an object ref.
- setSel : (SEL) value ..... Set the value to a selector
- setBool : (BOOL) value ..... Set the value to a boolean
- setInt : (int) value ..... Set the value to an integer
- setLong : (long) value ..... Set the value to a long
- setDouble : (double) value ..... Set the value to a double
- setString : (char \*) value ..... Set the value to a text string
- (Class) getClass ..... Return the class value
- (id) getObject ..... Return the object reference
- (SEL) getSel ..... Return the selector value
- (BOOL) getBool ..... Return the boolean value
- (int) getInt ..... Return the integer value
- (long) getLong ..... Return the long value
- (double) getDouble ..... Return the double value
- (char \*) getString ..... Return the text string
- (Class) toClass ..... Convert the value to a class
- (id) toObject ..... Convert the value to a (new) object
- (SEL) toSel ..... Convert the value to a selector
- (BOOL) toBool ..... Convert the value to a boolean
- (int) toInt ..... Convert the value to an integer
- (long) toLong ..... Convert the value to a long
- (double) toDouble ..... Convert the value to a double
- (DText \*) toText ..... Convert the value to a text object

---

Version 0.7.0. This card may be freely distributed under the terms of the GNU general public licence

Copyright © 2003-2005 by Dick van Oudheusden