**Objective C Tutorial**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSAutoreleasePool \* pool = [[NSAutoreleasePool alloc] init];

NSLog (@**"hello world"**);

[pool drain];

return 0;

}

**Objective-C Variables**

|  |  |  |
| --- | --- | --- |
| **Type** | **Storage size** | **Value range** |
| char | 1 byte | -128 to 127 or 0 to 255 |
| unsigned char | 1 byte | 0 to 255 |
| signed char | 1 byte | -128 to 127 |
| int | 2 or 4 bytes | -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 |
| unsigned int | 2 or 4 bytes | 0 to 65,535 or 0 to 4,294,967,295 |
| short | 2 bytes | -32,768 to 32,767 |
| unsigned short | 2 bytes | 0 to 65,535 |
| long | 4 bytes | -2,147,483,648 to 2,147,483,647 |
| unsigned long | 4 bytes | 0 to 4,294,967,295 |

**Arithmetic Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Adds two operands |
| - | Subtracts second operand from the first |
| \* | Multiplies both operands |
| / | Divides numerator by denominator |
| % | Modulus Operator and remainder of after an integer division |
| ++ | Increment operator increases integer value by one |
| -- | Decrement operator decreases integer value by one |

**Relational Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | Checks if the values of two operands are equal or not. |
| != | Checks if the values of two operands are not equal. |
| > | Checks if the value of left operand is greater than the value of right operand. |
| < | Checks if the value of left operand is less than the value of right operand. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand. |

**Logical Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| && | Logical AND operator. If both the operands are non zero then condition becomes true. |
| || | Logical OR Operator. If any of the two operands is non zero then condition becomes true. |
| ! | Logical NOT Operator. Reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make false. |

**Assignment Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Assignment operator, Assigns values from right side operands to left side operand |
| += | Add AND assignment operator, It adds right operand to the left operand and assigns the result to left operand |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assigns the result to left operand |
| \*= | Multiply AND assignment operator, It multiplies right operand with the left operand and assigns the result to left operand |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assigns the result to left operand |
| <<= | Left shift AND assignment operator |
| >>= | Right shift AND assignment operator |
| &= | Bitwise AND assignment operator |
| ^= | bitwise exclusive OR and assignment operator |
| |= | bitwise inclusive OR and assignment operator |

**BAB 1**

* **LOOP**
* **NESTED LOOP**
* **WHILE LOOP**
* **BREAK STATEMENT**
* **CONTINUE STATEMENT**
* **IF STATEMENT**
* **SWITCH**

**Loop**

#import <Foundation/Foundation.h>

int main ()

{

/\* **for** loop execution \*/

int a;

**for**( a = 10; a < 20; a = a + 1 )

{

NSLog(@**"value of a: %d\n"**, a);

}

return 0;

}

The code above generates the following result.



**Nested for loop**

#import <Foundation/Foundation.h>

int main ()

{

int i;

int j;

i = 0;

**do**

{

NSLog (@**"Outer loop %i"**, i);

**for** (j = 0; j < 3; j++)

{

NSLog (@**" Inner loop number %i"**, j);

}

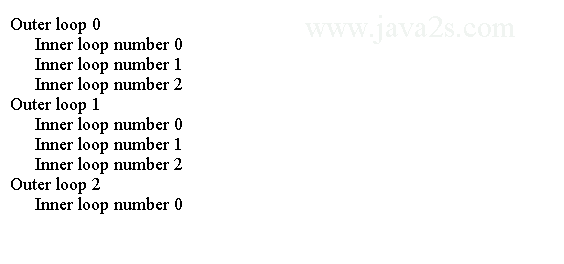
i++;

} **while** (i < 3);

return 0;

}

The code above generates the following result.



**While Loop**

#import <Foundation/Foundation.h>

int main ()

{

int a = 10;

**while**( a < 20 )

{

NSLog(@**"value of a: %d\n"**, a);

a++;

}

return 0;

}

The code above generates the following result 

#import <Foundation/Foundation.h>

int main ()

{

/\* local variable definition \*/

int a = 10;

/\* **do** loop execution \*/

**do**

{

NSLog(@**"value of a: %d\n"**, a);

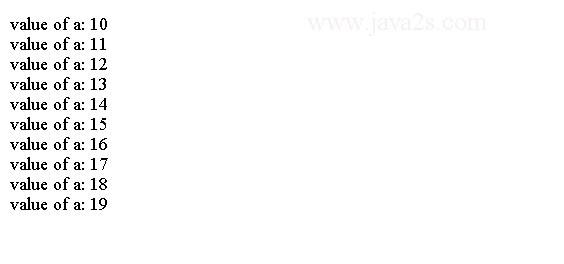
a = a + 1;

}**while**( a < 20 );

return 0;

}

The code above generates the following result.



**Break Statement**

#import <Foundation/Foundation.h>

int main ()

{

/\* local variable definition \*/

int a = 10;

/\* **while** loop execution \*/

**while**( a < 20 )

{

NSLog(@**"value of a: %d\n"**, a);

a++;

**if**( a > 15)

{

/\* terminate the loop using **break** statement \*/

**break**;

}

}

return 0;

}

The code above generates the following result.



**Break out of for**

#import <Foundation/Foundation.h>

int main ()

{

int i;

**for** (i = 0; i < 5; i++)

{

NSLog (@**"The value of i = %i"**, i);

**if** (i == 2)

{

**break**;

}

}

return 0;

}

The code above generates the following result.



**Continue Statement**

#import <Foundation/Foundation.h>

int main ()

{

/\* local variable definition \*/

int a = 10;

/\* **do** loop execution \*/

**do**

{

**if**( a == 15)

{

/\* skip the iteration \*/

a = a + 1;

**continue**;

}

NSLog(@**"value of a: %d\n"**, a);

a++;

}**while**( a < 20 );

return 0;

}



**Skipping for**

#import <Foundation/Foundation.h>

int main ()

{

int i;

**for** (i = 0; i < 5; i++)

{

**if** ((i % 2) != 0)

{

**continue**;

}

NSLog (@**"The value of i = %i"**, i);

}

return 0;

}

The code above generates the following result.



**If statement Example**

#import <Foundation/Foundation.h>

int main ()

{

int a = 10;

**if**( a < 20 )

{

NSLog(@**"a is less than 20\n"** );

}

NSLog(@**"value of a is : %d\n"**, a);

return 0;

}

**If else Example**

#import <Foundation/Foundation.h>

int main ()

{

int a = 100;

**if**( a < 20 )

{

NSLog(@**"a is less than 20\n"** );

}

**else**

{

NSLog(@**"a is not less than 20\n"** );

}

NSLog(@**"value of a is : %d\n"**, a);

return 0;

}

**If else if statement Example**

#import <Foundation/Foundation.h>

int main ()

{

int a = 100;

**if**( a == 10 )

{

NSLog(@**"Value of a is 10\n"** );

}

**else** **if**( a == 20 )

{

NSLog(@**"Value of a is 20\n"** );

}

**else** **if**( a == 30 )

{

NSLog(@**"Value of a is 30\n"** );

}

**else**

{

NSLog(@**"None of the values is matching\n"** );

}

NSLog(@**"Exact value of a is: %d\n"**, a );

return 0;

}

**Nested if statement Example**

#import <Foundation/Foundation.h>

int main ()

{

int a = 100;

int b = 200;

**if**( a == 100 )

{

**if**( b == 200 )

{

NSLog(@**"Value of a is 100 and b is 200\n"** );

}

}

NSLog(@**"Exact value of a is : %d\n"**, a );

NSLog(@**"Exact value of b is : %d\n"**, b );

return 0;

}

The code above generates the following result.



**Switch**

#import <Foundation/Foundation.h>

int main ()

{

char grade = **'B'**;

**switch**(grade)

{

**case** **'A'** :

NSLog(@**"A!\n"** );

**break**;

**case** **'B'** :

**case** **'C'** :

NSLog(@**"B\n"** );

**break**;

**case** **'D'** :

NSLog(@**"D\n"** );

**break**;

**case** **'F'** :

NSLog(@**"F\n"** );

**break**;

**default** :

NSLog(@**"Invalid grade\n"** );

}

NSLog(@**"Your grade is %c\n"**, grade );

return 0;

}

**Switch statement with number**

#import <Foundation/Foundation.h>

int main ()

{

int X = 2;

**switch** (X)

{

**case** 1:

NSLog (@**"X = 1"**);

**break**;

**case** 2:

NSLog (@**"X = 2"**);

**break**;

**default**:

NSLog (@**"Default code"**);

**break**;

}

return 0;

}

The code above generates the following result.

http://www.java2s.com/Tutorials/Objective_CImage/myResult/S/SWITCH_STATEMENT_WITH_NUMBER__DD45E3A961C8302065CB.PNG

**BAB 2**

* **NS-NUMBER**
* **ARRAYS**
* **STRING**
* **STRING OPERATION**
* **NSDATE**

**NSNUMBER**

#import <Foundation/Foundation.h>

int main ()

{

NSNumber \*myNumber;

myNumber = [NSNumber numberWithFloat:3.47];

NSLog (@**"The value in NSNumber = %@"**, myNumber);

return 0;

}

The code above generates the following result.

http://www.java2s.com/Tutorials/Objective_CImage/myResult/D/DESCRIPTION__67EFD5DF6EB57A7FA556.PNG

The following code shows how to use NSNumber to multiply two numbers and returns the product.

#import <Foundation/Foundation.h>

@**interface** SampleClass:NSObject

- (NSNumber \*)multiplyA:(NSNumber \*)a withB:(NSNumber \*)b;

@end

@implementation SampleClass

- (NSNumber \*)multiplyA:(NSNumber \*)a withB:(NSNumber \*)b

{

float number1 = [a floatValue];

float number2 = [b floatValue];

float product = number1 \* number2;

NSNumber \*result = [NSNumber numberWithFloat:product];

return result;

}

@end

int main()

{

NSAutoreleasePool \* pool = [[NSAutoreleasePool alloc] init];

SampleClass \*sampleClass = [[SampleClass alloc]init];

NSNumber \*a = [NSNumber numberWithFloat:10.5];

NSNumber \*b = [NSNumber numberWithFloat:10.0];

NSNumber \*result = [sampleClass multiplyA:a withB:b];

NSString \*resultString = [result stringValue];

NSLog(@**"The product is %@"**,resultString);

[pool drain];

return 0;

}

**Convert Float NSNumber to String**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

float fNumber = 12;

NSString \*floatToString = [NSString stringWithFormat:@**"%f"**, fNumber];

NSLog(@**"floatToString = %@"**, floatToString);

NSNumber \*number = [NSNumber numberWithFloat:30];

NSString \*numberToString = [number stringValue];

NSLog(@**"numberToString = %@"**, numberToString);

return 0;

}

The code above generates the following result.



**String to Number**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSString \*aFloatValue = @**"12.50"**;

float f = [aFloatValue floatValue];

float result = f \* 2 + 45;

NSLog(@**"f = %f and result = %f"**, f, result);

NSNumber \*aFloatNumber = [NSNumber numberWithFloat:[aFloatValue floatValue]];

NSLog(@**"aFloatNumber = %@"**, aFloatNumber);

return 0;

}

The code above generates the following result.



**Format a Number**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSNumber \*numberToFormat = [NSNumber numberWithFloat:9.99];

NSLog(@**"numberToFormat = %@"**, numberToFormat);

NSNumberFormatter \*numberFormatter = [[NSNumberFormatter alloc] init];

numberFormatter.numberStyle = NSNumberFormatterCurrencyStyle;

NSLog(@**"Formatted for currency: %@"**, [numberFormatter stringFromNumber:

numberToFormat]);

numberFormatter.numberStyle = NSNumberFormatterSpellOutStyle;

NSLog(@**"Formatted for spelling out: %@"**, [numberFormatter stringFromNumber

:numberToFormat]);

return 0;

}

**ARRAYS**

#import <Foundation/Foundation.h>

int main ()

{

int n[ 10 ];

int i,j;

**for** ( i = 0; i < 10; i++ )

{

n[ i ] = i + 100; /\* set element at location i to i + 100 \*/

}

**for** (j = 0; j < 10; j++ )

{

NSLog(@**"Element[%d] = %d\n"**, j, n[j] );

}

return 0;

}

**Creating an Array**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*object1 = @**"Hello"**;

NSString \*object2 = @**"world!"**;

NSNumber \*object3 = [NSNumber numberWithInt:45];

NSArray \*myArray;

myArray= [NSArray arrayWithObjects: object1, object2, object3, nil];

NSLog(@**"Array contents = %@"**,[myArray componentsJoinedByString:@**", "**]);

return 0;

}

The code above generates the following result.

http://www.java2s.com/Tutorials/Objective_CImage/myResult/C/CREATING_AN_ARRAY__D26BE0EC10C1DD66F71B.PNG

**Accessing All Items in an Array**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*object1 = @**"Hello"**;

NSString \*object2 = @**"world!"**;

NSString \*object3 = @**"Good-bye"**;

NSArray \*myArray;

myArray= [NSArray arrayWithObjects: object1, object2, object3, nil];

**for** (NSString \*randomVariable in myArray)

{

NSLog (@**"Array element = %@"**, randomVariable);

}

return 0;

}

The code above generates the following result.



**Array for loop**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*object1 = @**"Hello"**;

NSString \*object2 = @**"world!"**;

NSNumber \*object3 = [NSNumber numberWithInt:45];

NSArray \*myArray;

myArray= [NSArray arrayWithObjects: object1, object2, object3, nil];

int i;

**for** (i = 0; i < [myArray count]; i++)

{

NSLog (@**"Element %i = %@"**, i, [myArray objectAtIndex: i]);

}

return 0;

}

The code above generates the following result.



**Accessing an Item in an Array**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*object1 = @**"Hello"**;

NSString \*object2 = @**"world!"**;

NSNumber \*object3 = [NSNumber numberWithInt:45];

NSArray \*myArray;

myArray= [NSArray arrayWithObjects: object1, object2, object3, nil];

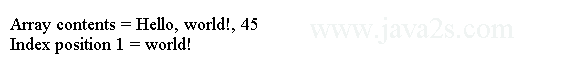
NSLog(@**"Array contents = %@"**,[myArray componentsJoinedByString:@**", "**]);

NSLog (@**"Index position 1 = %@"**, [myArray objectAtIndex:1]);

return 0;

}

The code above generates the following result.



**STRINGS**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*str1 = @**"Hello"**;

NSString \*str2 = @**"World"**;

NSString \*str3;

int len ;

NSAutoreleasePool \* pool = [[NSAutoreleasePool alloc] init];

str3 = [str2 uppercaseString];

NSLog(@**"Uppercase String : %@\n"**, str3 );

str3 = [str1 stringByAppendingFormat:@**"World"**];

NSLog(@**"Concatenated string: %@\n"**, str3 );

len = [str3 length];

NSLog(@**"Length of Str3 : %d\n"**, len );

str3 = [[NSString alloc] initWithFormat:@**"%@ %@"**,str1,str2];

NSLog(@**"Using initWithFormat: %@\n"**, str3 );

[pool drain];

return 0;

}

**STRING OPERATIONS**

**Convert to uppercase and lowercase**

#import <Foundation/Foundation.h>

int main ()

{

NSString \*testString = @**"Greetings from another planet!"**;

NSString \*targetString;

targetString = [testString uppercaseString];

NSLog (@**"All uppercase = %@"**, targetString);

NSLog (@**"\*\*\*\*\*\*\*\*\*\*"**);

targetString = [testString lowercaseString];

NSLog (@**"All lowercase = %@"**, targetString);

NSLog (@**"\*\*\*\*\*\*\*\*\*\*"**);

targetString = [testString capitalizedString];

NSLog (@**"All capitalized strings = %@"**, targetString);

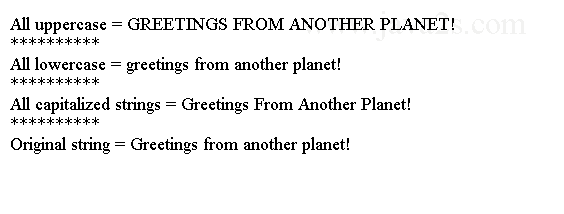
NSLog (@**"\*\*\*\*\*\*\*\*\*\*"**);

NSLog (@**"Original string = %@"**, testString);

return 0;

}

The code above generates the following result.



**NSDATE**

**Create Today's Date**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSDate \*todaysDate = [NSDate date];

NSLog(@**"Today's date is %@"**, todaysDate);

return 0;

}

The code above generates the following result.

http://www.java2s.com/Tutorials/Objective_CImage/myResult/C/CREATE_TODAY_S_DATE__67E5A1B41EDDAB414538.PNG

**Create Custom Dates**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSDateComponents \*dateComponents = [[NSDateComponents alloc] init];

dateComponents.year = 2007;

dateComponents.month = 6;

dateComponents.day = 29;

dateComponents.hour = 12;

dateComponents.minute = 01;

dateComponents.second = 31;

dateComponents.timeZone = [NSTimeZone timeZoneWithAbbreviation:@**"PDT"**];

NSDate \*iPhoneReleaseDate = [[NSCalendar currentCalendar] dateFromComponents:dateComponents];

NSLog(@**"The original iPhone went on sale: %@"**, iPhoneReleaseDate);

return 0;

}

The code above generates the following result.

http://www.java2s.com/Tutorials/Objective_CImage/myResult/C/CREATE_CUSTOM_DATES__FE5D1E737BA1A9EF8ED6.PNG

**Adding and Subtracting Dates**

#import <Foundation/Foundation.h>

int main (int argc, **const** char \* argv[])

{

NSString \*dateString = @**"02/14/2012"**;

NSDateFormatter \*df = [[NSDateFormatter alloc] init];

df.dateFormat = @**"MM/dd/yyyy"**;

NSDate \*valentinesDay = [df dateFromString:dateString];

NSLog(@**"Valentine's Day = %@"**, valentinesDay);

NSDateComponents \*weekBeforeDateComponents = [[NSDateComponents alloc] init];

weekBeforeDateComponents.week = -1;

NSDate \*vDayShoppingDay = [[NSCalendar currentCalendar] dateByAddingComponents:weekBeforeDateComponents toDate:valentinesDay options:0];

NSLog(@**"Shop for Valentine's Day by %@"**, vDayShoppingDay);

return 0;

}