**Arreglos en Objective C**

Un arreglo (vector o matriz, depende del país) es un objeto que contiene colecciones de otros objetos. Los objetos **Array** en Objective-C se manejan con la clase **NSArray**.

La clase **NSArray** contiene una serie de métodos específicamente diseñados para facilitar la creación y manipulación de los arreglos dentro de los programas de Objective-C.

A diferencia de otros lenguajes de programación orientados a objetos (C # es un ejemplo), los objetos contenidos en un arreglo no todos tienen que ser del mismo tipo.

**Arreglos mutables e inmutables**

Los arreglos  en Objective-C vienen en dos sabores: inmutables y mutables. El contenido de un arreglo "inmutable" no puede cambiar en tiempo de ejecución. Los arreglos inmutables se crean como instancia de la clase **NSArray**. Los arreglos "mutables" se crean mediante la clase **NSMutableArray** (una subclase de **NSArray**) y puede ser modificada después de haber sido creado e inicializado.

**Crear un arreglo**

La clase **NSArray** contiene un método de clase llamado **arrayWithObjects** que se puede recurrir para crear un nuevo objeto Array y se inicializa con elementos. Por ejemplo:

NSArray \*aColores;

aColores = [NSArray **arrayWithObjects**: @"Rojo", @"Verde", @"Azul", @"Amarillo", nil];

Es necesario añadir una entrada de tipo *nil* (en otros lenguajes es *null*, pero en Objective-C es *nil*) para que los métodos que se relacionan con los arreglos sepan donde termina este. Si omites la última entrada como *nil*, puede haber errores al momento de ejecutar tu aplicación.

Para crear un arreglo mutable que permitirá a los contenidos de la matriz ser modificada, tenemos que utilizar la clase **NSMutableArray**:

NSMutableArray \*misColores;

misColores = [NSMutableArray arrayWithObjects: @"Rojo", @"Verde", @"Azul", @"Amarillo", nil];

**Número de elementos de un arreglo**

Para saber cuantos elementos tiene un arreglo, utilizamos el método **count**:

NSLog (@"Número de elementos en el arreglo = %lu", [misColores count]);

El índice de los arreglos, al igual que en la mayoría de los lenguaje que provienen de C, inicia en cero. Se puede acceder a los elementos de un arreglo, haciendo pasar la posición de índice a través de un argumento del método **objectAtIndex**.

NSArray \*misColores;

int i;

int cont;

misColores = [NSArray arrayWithObjects: @"Rojo", @"Verde", @"Azul", @"Amarillo", nil];

cont = [misColores count];

for (i = 0; i < cont; i++)

NSLog (@"Elemento %i = %@", i, [misColores **objectAtIndex**: i]);

**Acceder a la información de un arreglo por medio de un ciclo for... in**

NSArray \*misColores;

misColores = [NSArray arrayWithObjects: @"Rojo", @"Verde", @"Azul", @"Amarillo", nil];

for (color in misColores )

NSLog (@"Elemento %@", color);

**Añadir nuevos elementos a los arreglos mutables**

Para añadir nuevos elementos a un arreglo mutable, lo realizaremos por medio del método **addObject** de la clase **NSMutableArray**. Por ejemplo:

[misColores addObject: @"Indigo"];

[misColores addObject: @"Violeta"];

**Insertar elementos en un arreglo mutable por medio de un índice**

Para insertar elementos en cualquier punto de un arreglo mutable (desplaza no elimina), lo podemos efectuar por medio del método ***insertObject****.* Por ejemplo:

[misColores **insertObject**: @"Indigo" atIndex: 1];

[misColores **insertObject**: @"Violeta" atIndex: 3];

**Borrar elementos de un arreglo**

Objective\_C nos proporciona muchos formas de  borrar elementos de un arreglo. Aquí se muestran las principales, pero puede ver la lista completa en la documentación de la librería *Fundation*.

Para borrar el elemento de un índice en específico se utiliza el método ***removeObjectAtIndex***:

[misColores **removeObjectAtIndex**: 0];

Para borrar el primer elemento del arreglo utilice el método ***removeObject***:

[misColores removeObject: @"Rojo"];

Para borrar un elemento por su contenido, utilice el método ***removeObjectIdenticalTo***:

[misColores removeObjectIdenticalTo: @"Red"];

Para borrar todo el contenido de un arreglo, utilice el método ***removeAllObjects***:

[misColores **removeAllObjects**];

Para borrar el último elemento de un arreglo, utilice el método  ***removeLastObject***:

[misColores removeLastObject];

**Ordenar un arreglo**

Existen muchas formas de ordenar un arreglo. La más sencilla es por medio del método *sortedArrayUsingSelector.*

Por ejemplo:

NSMutableArray \*misColores = [NSMutableArray arrayWithObjects: @"rojo", @"verde", @"azul", @"amarillo", nil];

NSArray \*misColoresOrdenados;

misColoresOrdenados = [misColores **sortedArrayUsingSelector**:@selector(localizedCaseInsensitiveCompare:)];

Como puede observarse, este método regresa otro objeto de tipo arreglo y utiliza el método *localizedCaseInsensitiveCompare*. Otros métodos que se pueden utilizar son ***NSOrderedAscending***, ***NSOrderedSame***, ***NSOrderedDescending***.

#import <Foundation/Foundation.h>

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

{

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

NSMutableArray \*array;

array = [[NSMutableArray alloc] init];

int i;

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

{

NSNumber \*newNumber = [[NSNumber alloc] initWithInt:(i \* 3)];

[array addObject:newNumber];

array[i] = newNumber;

}

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

NSNumber \*numberToPrint = [array objectAtIndex:i];

NSLog(@"The number at index %d is %@", i, numberToPrint);

}

// [pool drain];

return 0;

}