Programming in Objective-C
Fifth Edition
Developer's Library
http://www.apeth.com/iOSBook/
#include <stdio.h>

int main(int argc, const char * argv[])
{
    printf("Hello, World!\n");
    return 0;
}

statements

printf("Hello, World!\n");
variables

int n;
Primitive Data Types

char
double
float
int
long
unsigned int
...
printf

%s
%d
%lu
%lld
%f
...

Boolean Expressions

!  >  >=  ==  <=  <  &&  ||
Conditions

if  else
Loops
for (initialization; condition; increment) {
  statements
}

while (condition) {
    statements
}

do {
    statements
} while (condition);
Casting
Pointers

char *
double *
float *
int *
long *
char **
Pointers

* and &
struct
enum
Memory Management

malloc  free
Objective-C
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]){
    @autoreleasepool {
        NSLog(@"Hello, World!");
    }
    return 0;
}
Xcode

Xcode provides everything developers need to create great applications for Mac, iPhone, and iPad. Xcode 4 has been streamlined to help you write better apps. It has unified user interface design, coding, testing, and debugging all within a single window. The Xcode IDE analyzes the details of your project to identify mistakes in both syntax and logic, it can ev...

What's New in Version 4.6.3

- Fixes an issue where debugging in the iOS Simulator could hang on OS X 10.8.4.

Information

Category: Developer Tools
Updated: Jan 13, 2013
Version: 4.6.3
Price: Free
Size: 1.65 GB
Language: English
Seller: Apple Inc.
© 1999-2013 Apple Inc.

Rated 4+
Requirements:
OS X 10.7.4 or later
// main.c
// SwapFailure
//
// David J. Malan
// Harvard University
// malan@harvard.edu
//
// Fails to swap two variables' values.
//
#include <stdio.h>

// function prototype
void swap(int a, int b);

int main(int argc, const char * argv[]) {
    int x = 0;
    int y = 1;
    printf("x is %d\n", x);
    printf("y is %d\n", y);
    printf("Swapping x and y...\n");
    swap(x, y);
    printf("Success!\n");
    printf("x is %d\n", x);
    printf("y is %d\n", y);
    return 0;
}
Data Types

BOOL

id (and nil)

...
Foundation Data Types

NSInteger

NSPoint

NSRect

NSSize

NSSize

NSUInteger

...
Classes and Objects
@interface Foo: NSObject {
    // instance variables
}

// declarations of methods
@end
@implementation Foo

// definitions of methods
@end
Instance Variables

@protected
@private
@public
Class Methods

+alloc;
Messages

Student *student = [Student alloc];
Instance Methods

- (void)init;

- (int)age;

- (void)setAge:(int)age;
Messages

@student init;  
@student age;  
@student setAge:20;
Selectors

alloc

init

age

setAge:
@property

assign
copy
strong
weak

atomic
nonatomic

readonly
readwrite
init... Methods

- (id)initWithName:(NSString *)name andAge:(int)age;
init... Methods

Student *alice = [student initWithName:@"Alice" andAge:20];
Collections

NSArray
NSMUTABLEARRAY

NSDictionary
NSMUTABLEDICTIONARY

NSSet
NSMUTABLESET

...
Fast Enumeration

for (id foo in bar) {
    // do something with foo
}

@interface Foo (Bar)

-(void)baz;

@end
Blocks

^return_type(type1 arg1, type2 arg2) {
    // implementation
}

@interface Student <NSCopying> {
}

...
Protocols

@implementation Student

...

-(id)copyWithZone:(NSZone *)zone {
    Student *s = [Student allocWithZone:zone];
    [s initWithName:_name andAge:_age];
    return s;
}

...

@end
NSException

@try {
  // try something here
}

@catch (NSException *e) {
  // handle exception here
}

@finally {
  // do something here
}
NSError

NSError *e = nil;
if ([foo bar:baz error:&e] == nil)
{
    // handle error
}
to be continued...