fn7の日記

<[勉強][プログラミング][Objecti... | [雑記][Apple]iPadは想像以上>

2010-02-03

Objective-C 最速基礎文法マスター

勉強, プログラミング, Objective-C

Java基礎文法最速マスター - 何かしらの言語による記述を解析する日記を参考に、Objective-Cのものを書いてみた。

まだまだ歴が浅いので間違っている所があるかもしれません。

基礎

デバッガコンソール出力

Xcodeのデバッガ出力を行うにはNSLog関数を使います。

printfと同様のフォーマット文字列を使いますが、%@はNSStringのインスタンスを表示する時に使用します。

```
NSLog(@"message: %@",messge);
NSLog(@"object: %@",[object description]);
```

コメント

コメントです。

```
// 一行コメント
/*
   複数行コメント
*/
```

変数宣言

変数の宣言です。Cと同じです。変数の宣言時にはデータ型を指定します。

```
int n;
```

データ型はCのデータ型が使えます。

```
int n;
unsigned int n;
char n;
unsigned char n;
long n;
float n;
double n;
```

真偽値はYES,NOが用意されています。

```
BOOL isOK = YES;
BOOL isBAD = NO;
```

Objective-Cのオブジェクトの宣言はポインタの宣言です。

```
NSString *string;
NSArray * array;
NSDictionary* dictinary;
```

オブジェクトの生成とメモリ解放

オブジェクト生成には、alloc+init系とautorelease済みの2通りがあります。

GCなしの場合メモリ管理は参照カウント方式となるので、使用には注意が必要です。

参照カウントをインクリメントするにはretain,デクリメントするにはreleaseを使います。

alloc+init系

初期化時にalloc+init系メソッドを使った場合には 参照カウントが1の状態なので、使用が終わればreleaseして解放する必要があります。

```
NSArray *array = [[NSArray alloc] initWithObjects:@"a",@"b",@"c",nil];
// なんか処理
[array release]; // 解放
```

alloc+init系メソッドを使って生成したオブジェクトもautoreleaseに任せることができます。

```
NSArray *array = [[[NSArray alloc] initWithObjects:@"a",@"b",@"c",nil] autorelease];
```

autorelease済み

```
NSArray *array = [NSArray arrayWithObjects:@"A",@"B",@"C",nil];
```

数値

```
int i = 2
int i = 100000000;

float radius = 1.0f;

double diameter = 2 * M_PI * radius;
```

四則演算

```
num = 1 + 1;
num = 1 - 1;
num = 1 * 2;
num = 1 / 2;
```

商の求め方です。割る数と割られる数が両方とも整数の場合、計算結果の小数点以下が切り捨てられます。

```
num = 1 / 2;  // 0
```

割る数と割られる数のどちらかが小数の場合、計算結果の小数点以下が切り捨てられません。

```
num = 1.0 / 2;    // 0.5
num = 1 / 2.0;    // 0.5
num = 1.0 / 2.0;  // 0.5
```

余りの求め方です。

```
// 余り
mod = 4 % 2 ;
```

インクリメントとデクリメント

```
// インクリメント
num++;
++num;

// デクリメント
num--;
--num;
```

単体で使用する場合には、どちらを使用しても問題ないですが、代入式で使用する場合には注意が必要です。

文字列

文字列の表現

文字列はNSStringを使います。リテラル表現として@""が使えます。

NSStringは変更不可能な文字列です。

```
NSString *string = @"Hello World!";
```

変更可能な文字列は

```
NSMutableString * string = [NSMutableString stringWithString:@"Hello World!"];
```

文字列操作

```
// 結合
NSMutableString * string = [NSMutableString stringWithString:@"aaa"];
[string appendString:@"bbb"]

//分割
NSMutableString * string = [NSMutableString stringWithString:@"aaa,bbb,ccc"];
NSArray *record = [string componentsSeparatedByString:@","];

// 長さ
int length = [@"abcdef" length];

//切り出し
NSString *string = [@"abcd" substringWithRange:NSMakeRange(0,2)];       // ab

//検索
NSString *string = @"abcd";
NSRange range = [string rangeOfString:@"cd"];
NSLog(@"%d:%d",range.location,range.length); //みつからない場合にはlength = 0になる
```

配列

宣言

```
NSArray *array;
```

配列の生成

```
NSArray *array;
array = [NSArray arrayWithObjects: @"a",@"b",@"c",nil];
NSMutableArray *array = [NSMutableArray array]; // 変更可能な配列を宣言と同時に代入
```

要素の参照と代入

```
// 要素の参照
[array objectAtIndex:0];
[array objectAtIndex:1];

// 代入(NSMutableArrayのインスタンス)
[array removeObjectAtIndex:1]; // 一度要素を削除して
[array insertObject:@"1" atIndex:1]; // 削除したとこに挿入
```

配列の要素数

```
int array_num = [array count];
```

辞書

宣言

```
NSDictionary *dictinary;
dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
  @"value1",@"key1",
  @"value2",@"key2",
  nil,
  ];
NSMutableDictionary *dictionary = [NSMutableDictionary dictionary]; // 変更可能な辞書を宣言と同時に代入
>|objc|
```

要素の参照と代入

```
//要素の参照
id val1 = [dictionary objectForKey:@"key1"];
id val2 = [dictionary objectForKey:@"key2"];
// 代入(NSMutableDictionaryのインスタンス)
>|objc|
[dictionary setObject:@"value1" forKey:@"key1"];
```

辞書の操作

・キーの取得

```
NSArray *keys = [dictionary allkeys]
```

・値の取得

```
NSArray *values = [dictionary allValues];
```

・キーの削除(NSMutableDictionaryのインスタンス)

```
[dictionary removeObjectForKey:@"key1"];
```

制御文

if文

```
if( 条件 ){

}
```

if 〜 else文

```
if (条件){

}else{

}
```

if 〜 else if 文

```
if ( 条件 ){

}else if( 条件 ){

}
```

while文

```
int i = 0;
while(i < 5){
        // なんか処理
  i++
}
```

for文

```
for(int i=0;i<5;i++){
  //処理
}
```

関数定義

C言語の関数定義も可能。基本的にクラスのメソッドとして定義します。

また、既存のクラスを拡張する方法としてカテゴリというものがあります。

```
@interface ClassA : NSObject{
        NSString *name_;
}
@property (nonatomic,copy) NSString *name_;
-(void) helloWorld:(NSString *)name{
  NSLog(@"hello %@ world! from %@",name,self.name_);
}
@end
```

ファイル入出力

```
NSString *inputFilePath = [INPUTFILEPATH stringByExpandingTildeInPath];
NSString *outputFilePath = [OUTPUTFILEPATH stringByExpandingTildeInPath];
NSFileManager *fm = [NSFileManager defaultManager];
if(![fm fileExistsAtPath:outputFilePath]){
  [fm createFileAtPath:outputFilePath contents:nil attributes:nil];
}
NSFileHandle *input = [NSFileHandle fileHandleForReadingAtPath:inputFilePath];
NSFileHandle *output = [NSFileHandle fileHandleForWritingAtPath:outputFilePath];
@try{
  NSData *data;
  while((data = [input readDataOfLength:1024]) && 0 < [data length]){
    [output writeData:data];
  }
} @finally {
  [input closeFile];
  [output closeFile];
}
```

知っておいたほうがよい文法

高速列挙

NSArrayやNSDictionaryは高速列挙という方法によって要素の列挙が簡単にできます

```
NSArray *array = [NSArray arrayWithObjects: @"A",@"B",@"C",nil];
for(id i in array){
  //なんか処理
}


NSDictionary *dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
    @"value1",@"key1",
    @"value2",@"key2",
    @"value3",@"key3",
    nil
  ];
// キーでループする場合
for(id i in [dictionary keyEnumerator]){
  // なんか処理
}
// 値でループする場合
for(id i in [dictionary objectEnumerator]){
  // なんか処理
}
```

一時変数はidで受けてもいいし、キーの型が判明している時には型を指定した方が良いでしょう。

```
for(NSString *k in [dictionary keyEnumerator]){
  // 処理
}
```

カテゴリ

LLのように既存のクラスにメソッドを追加することができます。

```
@interface NSString (Decorate)
// カテゴリに属するメソッド定義を行う
- (NSString *) decorateWithString:(NSString *)string;
@end

@implementation NSString (Decorate)
-(NSString *) decorateWithString:(NSString *)string{
  return [NSString stringWithFormat:@"%@%@%@",string,self,string];
}

@end


NSLog(@"test: %@",[@"[MSG]" decorateWithString:@"**"]); // **[MSG]**
```

プロトコル

クラスがプロトコルに適合しますと宣言された場合、プロトコルで規定されているメソッドを実装しなければならないようにすることができます。

プロパティ

クラスのメンバ変数へのアクセサを提供する機能です。

使用するにあたって知っておくべきことがいくつかありますが、ひとつだけ紹介します。

```
@property (nonatomic, retain) NSArray *array_;
```

上記のようにretainで宣言されたプロパティへの代入はアクセサを通さないとretainが有効になりません。

```
-(id)initWithParam:(NSArray *)array{
  if(self = [super init]){
    array_ = array; // retainされない。
    self.array_ = array; // retainされる。
  }
  return self;
}

-(void)dealloc{
  [array_ release];
  [super dealloc];
}
```

retainされていない状態となるので、自分が保持しているにもかかわらず外部の処理でreleaseされることでオブジェクトが破棄される可能性があります。破棄されてしまうとEXC_BAD_ACCESS もしくはdouble freeが発生します。

初心者には気づきにくいと思われますのでご注意ください。

検証してみた：アクセサ通さないとretainされない件 - fn7の日記

Google Objective-Cスタイルガイド 日本語訳

まだ少ししか読んでないけど、すごく勉強になります。

GoogleObjective-Cスタイルガイド日本語訳

[参考資料]

Cocoaでファイル入出力 -- BONNOH FRACTION 13

NSRangeを作るのなら NSMakeRange関数を使う - iRSSの日記 - iPhoneアプリ開発グループ

Permalink | コメント(0) | 23:24 253 users

コマンドラインでコンパイル

勉強, プログラミング, Objective-C

意外と簡単にコンパイルができるようだ。

Xcodeから適当なプロジェクトを作成してmain.mだけとり出す。

main.mを適当に編集しちゃう。

```
#import <Cocoa/Cocoa.h>

int main(){
  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
  NSLog(@"Hello World!");
  [pool release];
  return 0;
}
```

コンパイルしてエラーがなければ実行する。

```
$ gcc main.m -o main -framework Cocoa && ./main
```

[参考]

Objective-Cの実験コードをmainだけ書いてgccでコンパイルする方法 - 拡張現実ライフ - モバイルデジタルサイネージ編

Permalink | コメント(0) | 23:38

コメントを書く

なまえ メール(非公開) URL

画像認証 画像内の文字列を入力して下さい

投稿

リンク元

- 76 http://d.hatena.ne.jp/seikenn/20100203/programmingMaster
- 47 http://b.hatena.ne.jp/hotentry/it
- 45 http://b.hatena.ne.jp/hotentry
- 42 http://b.hatena.ne.jp/
- 34 http://b.hatena.ne.jp/entrylist
- 28 http://reader.livedoor.com/reader/
- 20 http://www.google.co.jp/reader/view/
- 18 http://www.google.com/reader/view/
- 11 http://www.google.co.jp/reader/view/?hl=ja&tab=wy
- 9 http://ig.gmodules.com/gadgets/ifr?view=home&url=http://www.hatena.ne.jp/tools/gadget/bookmark/bookmark_gadget.xml&nocache=0&up_display_item=10&up_display_thumbnail=false&up_display_summary=true&up_category_all=true&up_category_general=true&up_category_so

おとなり日記

- 2010-02-01 かわおの徒然記 5/49 10%
- 2010-02-03 うなの日記 5/52 9%
- 2010-02-02 プログラマーの脳みそ 10/132 7%
- 2010-02-02 プログラミング言語を作る日記 7/90 7%
- 2010-02-01 notes plastiques 6/86 6%

<[勉強][プログラミング][Objecti... | [雑記][Apple]iPadは想像以上>