The main purpose of Objective-C is to add object orientation to the C programming language

IDE

<https://franklingu.github.io/programming/2013/12/30/objective-C-in-linux-(ubuntu)/>

$ gcc --version

$ sudo apt-get install **gobjc** // install GNU objective-c compiler

$ sudo apt-get install **gnustep**

$ sudo apt-get install **gnustep-devel** //GNUStep dev libraries (equivalent to Cocoa on macosx)

命令行方式：

hello.m

#import <Foundation/Foundation.h>

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

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

NSLog (@"hello world");

[pool drain];

return 0;

}

$ gcc hello.m -o hello `gnustep-config --objc-flags` -lgnustep-base -lobjc // compile

$ ./hello // 运行可执行程序

**// same to C**

Data Types: char, int, float, double, void

const int width = 5;

double mean = (double) 17 / 5 ; // Type Casting, 转型和C语言一模一样

double balance[] = {1.0, 2.0, 3.0, 4.0, 5.0}; //Arrays

double\* ptr = balance;

\*(ptr+i), balance[i];

if (ptr) {} // if ptr is not null

if (!ptr) {} // if ptr is null

Arithmetic Operators: +, -, \*, /, %, ++, --

Relational Operators: ==, !=, >, <, >=, <=

Logical Operators: &&, ||, !

Bitwise Operators: &, |, ^, ~, <<, >>

Assignment Operators: =, +=, -=, \*=, /=, %=, <<=, >>=, &=, ^=, |=

Misc Operatos: sizeof(width), ptr = &value, \*ptr, ?=

Loops: for, while, do…while, break, continue

Decision Making: if…else, switch , exp1 ? exp 2 : exp 3;

typedef struct Book {

NSString \*title;

NSString \*author;

NSString \*subject;

int book\_id;

} Book;

Book book;

book.title = @"Objective-C Programming";

book.author = @"Nuha Ali";

book.subject = @"Objective-C Programming Tutorial";

book.book\_id = 6495407;

Preprocessors: #define, #include, #undef, #ifdef, #ifndef, #if, #else, #elif, #endif, #error, #pragma, \_\_DATE\_\_, \_\_TIME\_\_, \_\_FILE\_\_, \_\_LINE\_\_, \_\_STDC\_\_

**Strings**

“hello,” “ dear”

“hello, \

dear”

NSString\* s1 = @"hello"; // NSString

NSLog(@"Greeting message: %@\n", s1 );

NSString\* s3 = [[NSString alloc] initWithFormat:@"%@ %@",s1,s2]; //s1 s2

Square \*square = [[Square alloc]init]; // NSArray

Rectangle \*rectangle = [[Rectangle alloc]init];

NSArray \*shapes = [[NSArray alloc]initWithObjects: square, rectangle,nil];

id object1 = [shapes objectAtIndex:i];

[object1 printArea];

NSLog(@"Hello, World! \n"); // 同printf()

**Method**

// method declaration

- (return\_type) function\_name:( argumentType1 )argumentName1

joiningArgument2:( argumentType2 )argumentName2 ...

joiningArgumentn:( argumentTypen )argumentNamen;

// method definition

- (return\_type) method\_name:( argumentType1 )argumentName1

joiningArgument2:( argumentType2)argumentName2 ...

joiningArgumentn:( argumentTypen)argumentNamen {

body of the function

}

- (int) max:(int)num1 andNum2:(int)num2; // method declaration

- (int) max:(int) num1 secondNumber:(int) num2 { // method definition

return num1 > num2 ? num1 : num2;

}

**Blocks like closures or lambdas**

returntype (^blockName)(argumentType); // block declaration

returntype (^blockName)(argumentType)= ^{…}; // block implementation

typedef void (^**CompletionBlock**)();

@interface SampleClass:NSObject

- (void)performActionWithCompletion:(**CompletionBlock**)completionBlock;

@end

@implementation SampleClass

- (void)performActionWithCompletion:(**CompletionBlock**)completionBlock {

NSLog(@"Action Performed");

completionBlock();

}

// client

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

[sampleClass performActionWithCompletion:^{

NSLog(@"Completion is called to intimate action is performed.");

}];

Categories and Extensions //类似于动态编程，给现有class添加方法

@interface NSString(MyAdditions)

+(NSString \*)getCopyRightString;

@end

@implementation NSString(MyAdditions)

+(NSString \*)getCopyRightString {

return @"Copyright TutorialsPoint.com 2013";

}

@end

NSString \*copyrightString = [NSString getCopyRightString];

**Class**

// Objective C

@class Stash; // forward declaration

//同java, C#一样，单继承，根基类，实现基本功能：memory allocation and initialization

@interface Box: NSObject

{

double length; //成员变量 default = protected

double breadth;

}

@property(nonatomic, readwrite) double height; // 同java, C#

@property(strong) Stash \* stashPtr; //指针类型strong, unsafe\_unretained, weak

-(double) volume; //定义成员函数

-(void) fun: (int) a (int) b;

-static void static\_fun(int a, int b);

@end

@implementation Box

-(id) init {

self = [super init];

length = 1.0; // or self->length = 1.0;

breadth = 1.0;

return self;

}

-(double) volume {

return length\*breadth\*height;

}

@end

// C++

class Box : Shape

{

double length; // default = private

double breadth;

public:

Box();

class Stash\* stashPtr; // forward declaration

void fun(int a, int b);

static void static\_fun(int a, int b);

}

void Box:: Box()

{

length = 1.0; // or self->length = 1.0;

breadth = 1.0;

}

Void Box::fun(int a, int b) {

…

Shape::fun(); // parent call

}

// client

// Objective-C uses [ ] to call member functions and passes parameters separated with :

Box \*box = [[Box alloc]init];

box.height = 5.0;

double volume = [box volume] ;

**[**ptr fun**:**5**:**3**]**; // call foo member with arguments 5 and 3

[box static\_fun:5:3]; // call static function of NSOBject with arguments 4 and 3

Box \* ptr = new Box (); // C++

ptr->foo(5,3);

Box::static\_fun(5,3);

**Protocol**

// Objective-C: Protocol

@protocol Shape

- (void) fun;

@end

@interface Box: NSObject<Shape>

@end

@implementation Box

- (void) fun { ... }

@end

// C++: Abstract Class

class Box {

virtual void fun() = 0;

};

class Box: public NSObject, public Shape {

void fun() { ... }

}

Downcasting

// C++

CPPObject\* ptr = ...; // some pointer

foo\* f = **dynamic\_cast**<foo\*>(ptr);

if (f) f->somefunction(5,3);

// Objective-C

NSObject\* ptr = ...; // some pointer

if ([ptr **isKindOfClass**:[foo class]]

[ptr somefunction:5:3];

**What C++ has and Objective-C hasn't**

* Static objects. Objects in Objective-C cannot be instantiated statically or in the stack. Only pointers.
* Multiple inheritance
* Namespaces
* Templates
* Operator overloading
* STL and algorithms ;
* Methods can be protected or private (in Obj-C, only public)
* const/mutable items
* friend methods
* References
* Anonymous function signatures (without a variable name)

**Foundation Framework**

The framework was developed by NeXTStep (NS)

#import <Foundation/Foundation.h>

NSArray,

NSDictionary

NSSet

NSMutableSet

NSMutableArray

NSMutableDictionary

NSArray \*array = [[NSArray alloc] initWithObjects:@"string1", @"string2",@"string3",nil];

for(NSString \*aString in array) {

NSLog(@"Value: %@",aString);

}

for(NSString \*aString in [array reverseObjectEnumerator]) { //逆序遍历

NSLog(@"Value: %@",aString);

}

NSString, NSScanner

NSDate, NSTimeZone, and NSCalendar

NSFileManager

NSException

**Memory Management**

* MRR (Manual Retain-Release) //背后是引用计数，但需要手动retain or release

Box \*box = [[Box alloc]init];

[box fun];

int count = [box retainCount]); //此时引用计数为1

[box retain]; //引用计数+1

[box release]; //引用计数-1

[box release]; //引用计数-1

box = nil;

* ARC (Automatic Reference Counting) //背后是引用计数，编译时期自动插入内存管理

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

Box \*box = [[Box alloc]init];

[box fun];

[pool drain];