# Objective-C - Under the hood

Michael Pan

# Outline

- Pointer & Object
- C Struct for class
- Dynamic feature

# Contacts

E-mail : scentsome@gmail.com

Facebook Group : Developer's Lesson

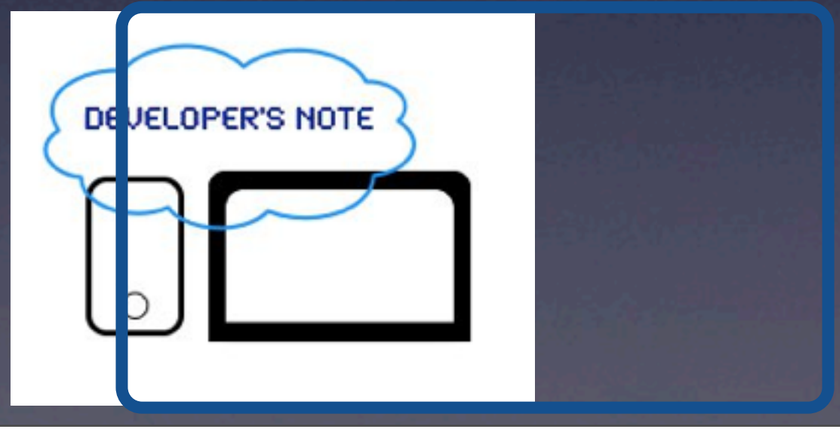
http://www.facebook.com/groups/developerslesson

Facebook Page :  Developer's Note

http://www.facebook.com/pages/Taipei-Taiwan/Developers-note/226724001803

Blogger : Developer's Note

http://iosdevelopersnote.blogspot.com/



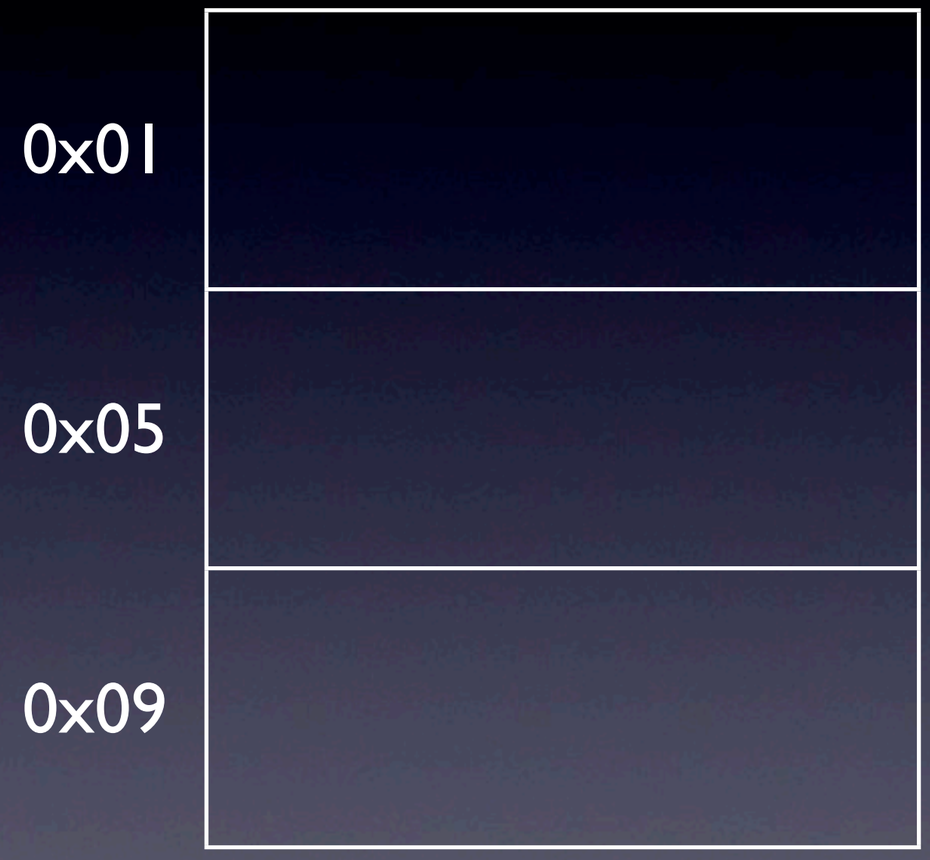
# Book

Objective-C 與 iOS 開發入門

http://www.books.com.tw/exep/prod/booksfile.php?item=0010517912


# Pointer & Object

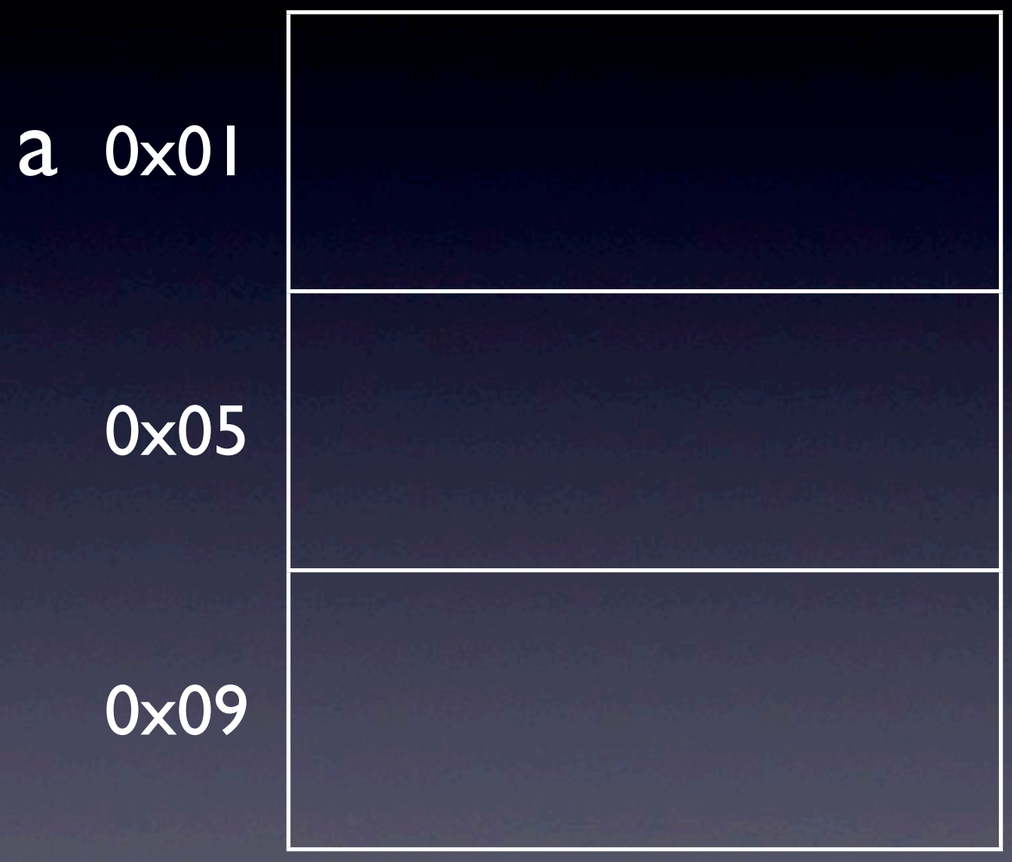
# Lets go from C



# Lets go from C

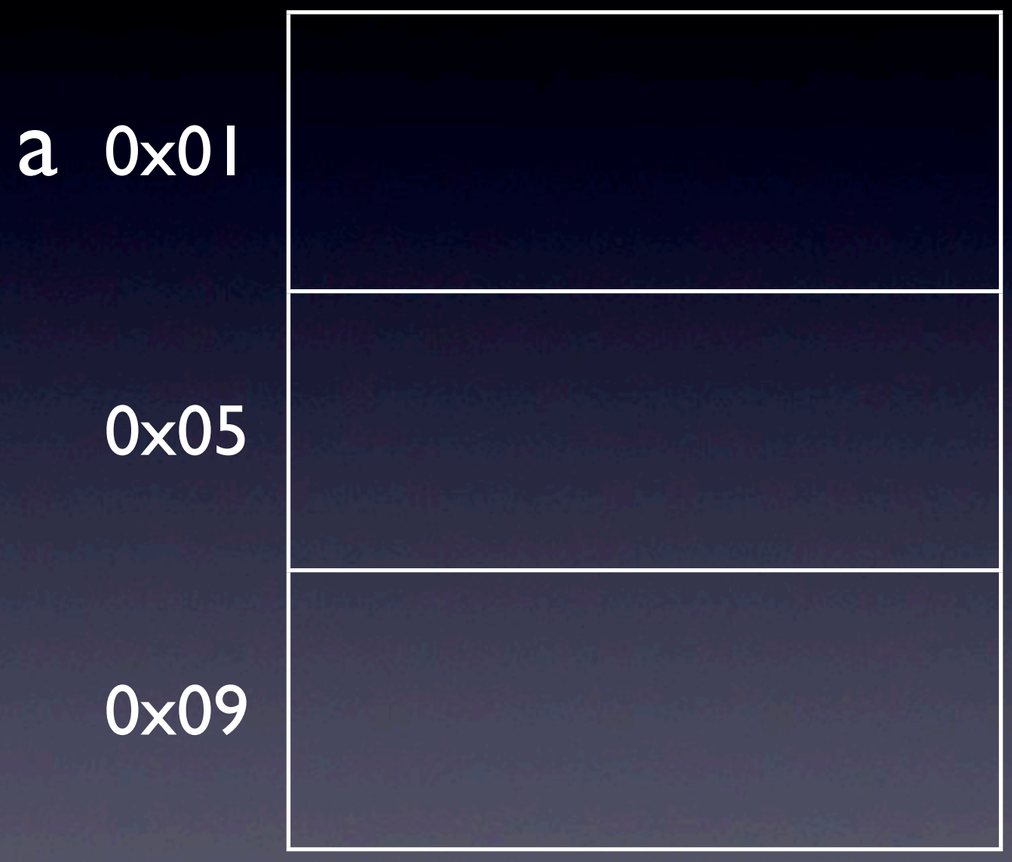

```
int a;
```

# Lets go from C



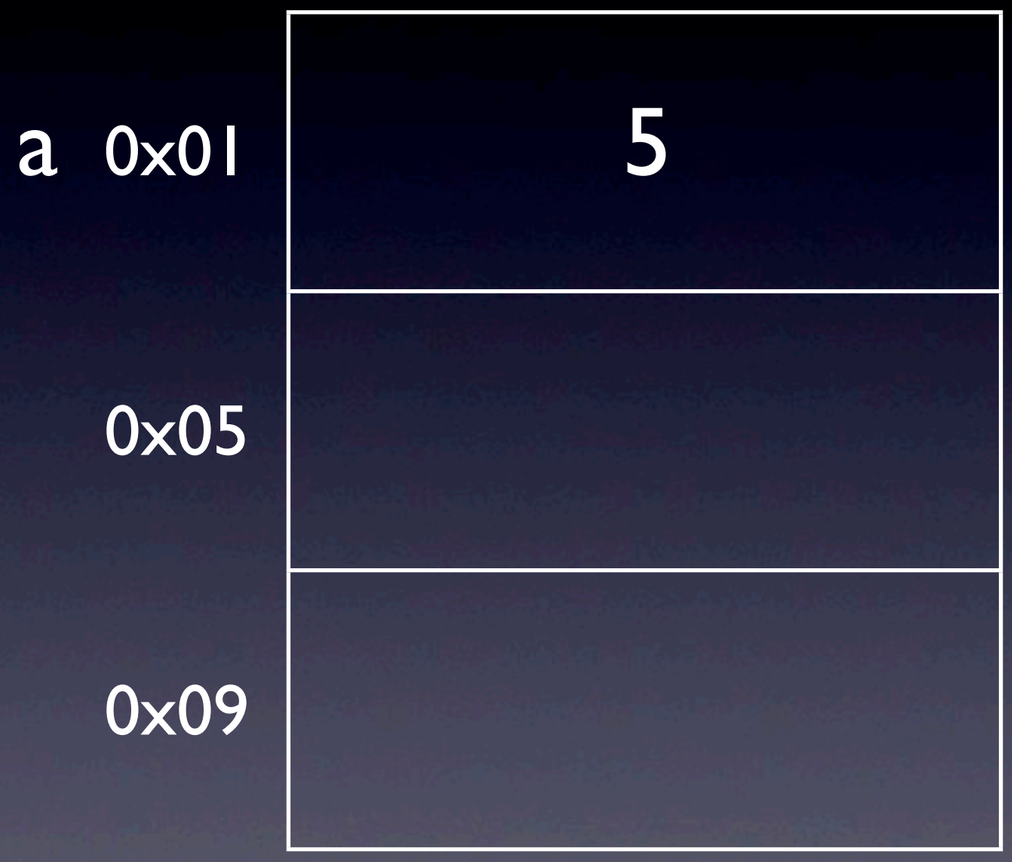

```
int a;
```

# Lets go from C



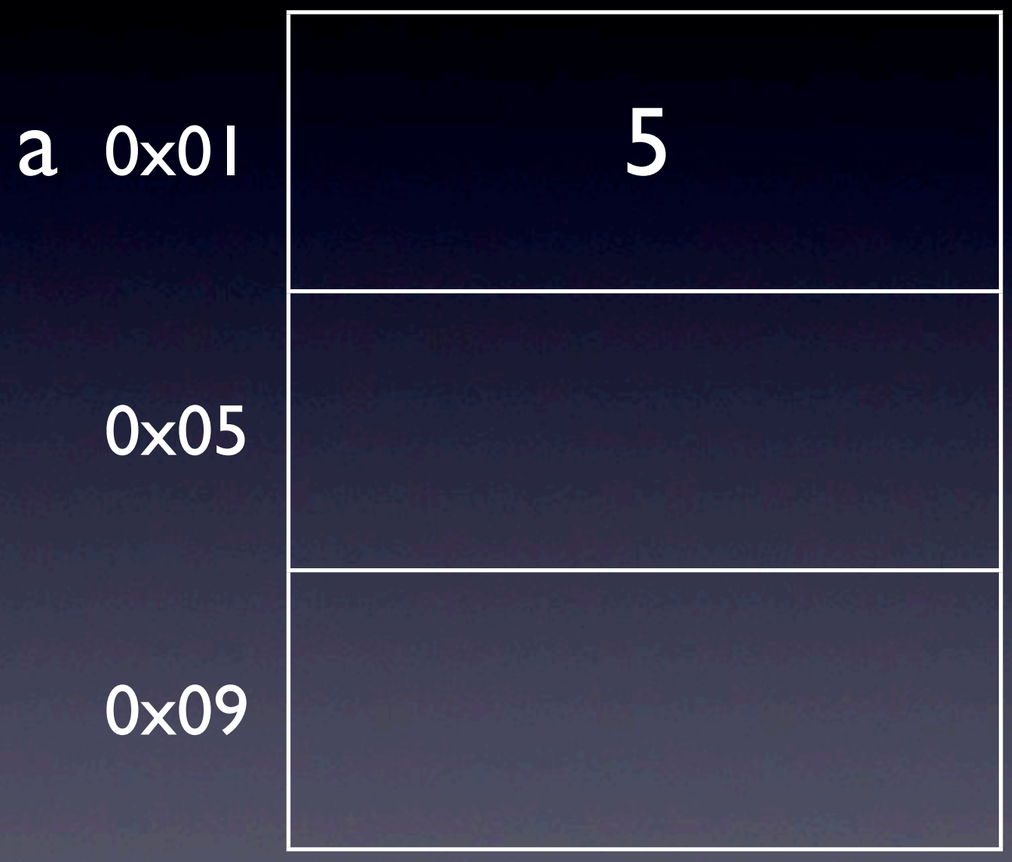

```
int a;
a=5;
```

# Lets go from C




```
int a;
a=5;
```

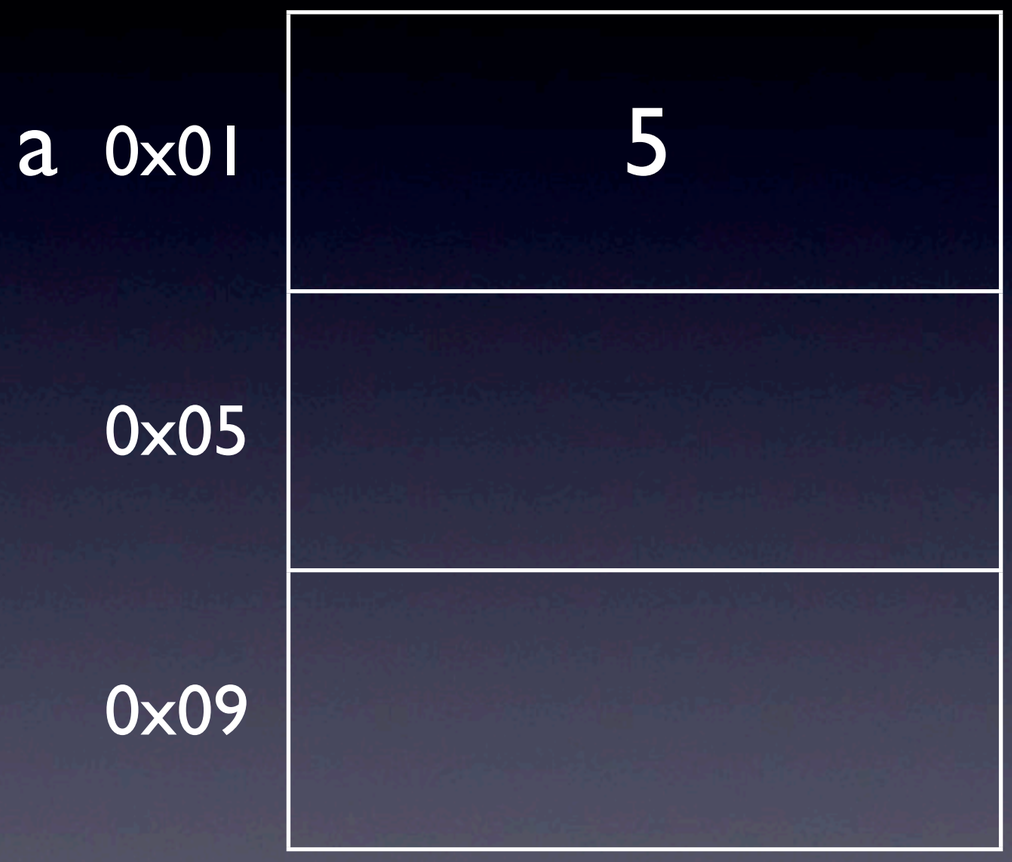
# Pointer




```
int a;
a=5;
```

**& => address of**

# Pointer

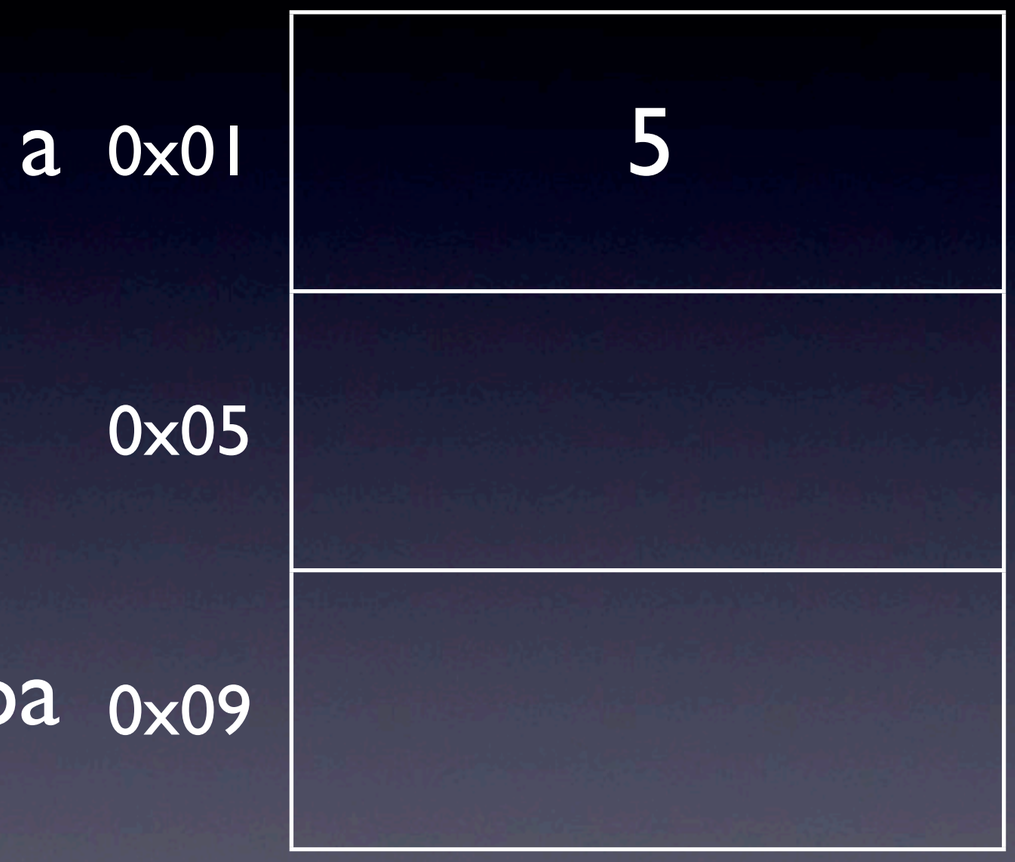



```
int a;
a=5;
```

```
int * pa;
```

**& => address of**

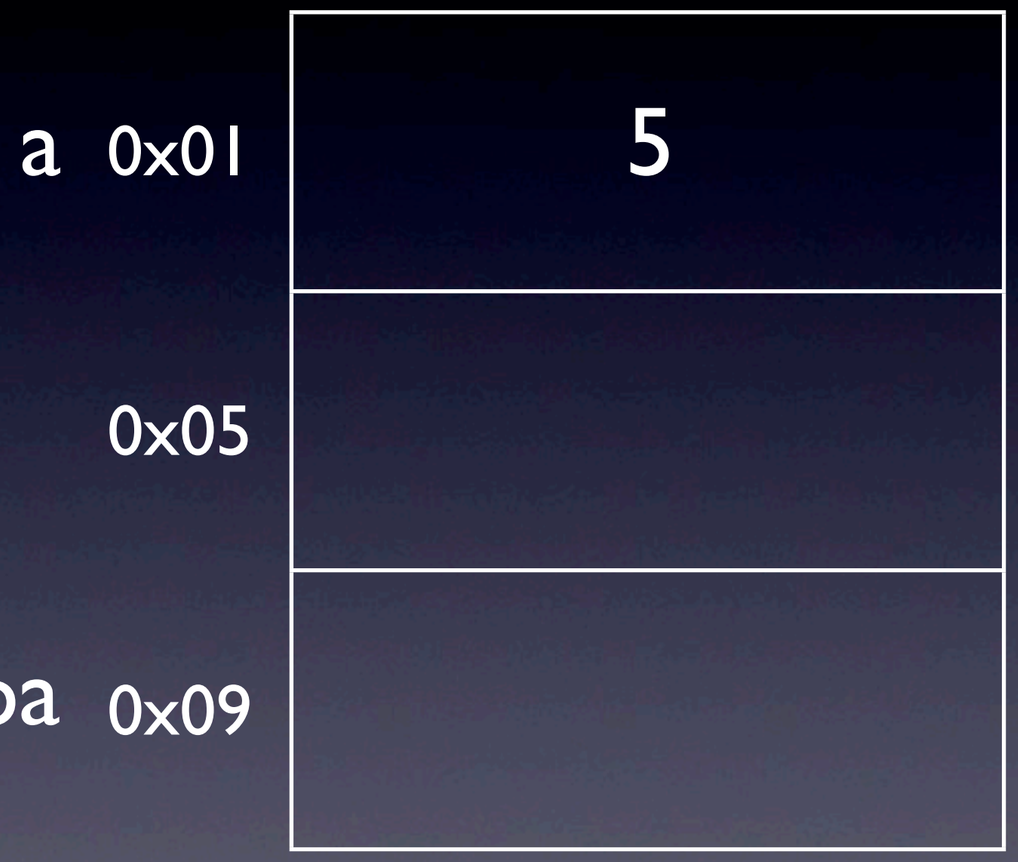
# Pointer




```
int a;
a=5;
int * pa;
```

**& => address of**

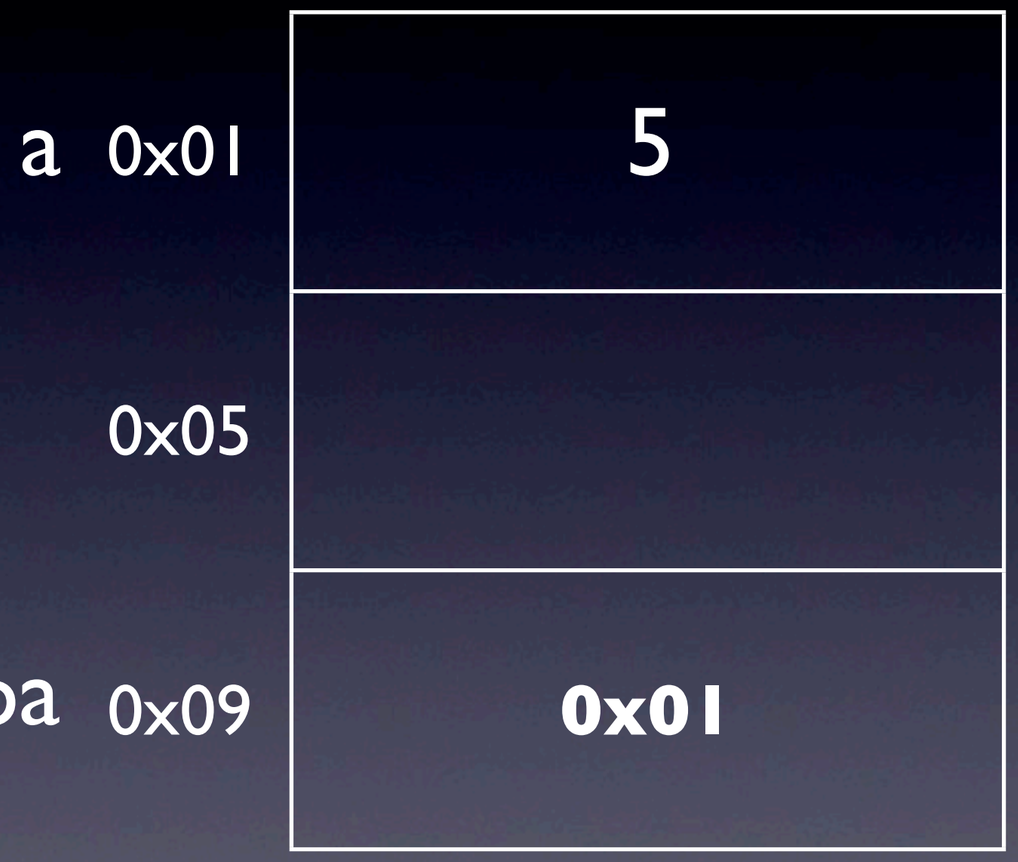
# Pointer




```
int a;
a=5;
int * pa;
pa = &a;
```

**& => address of**

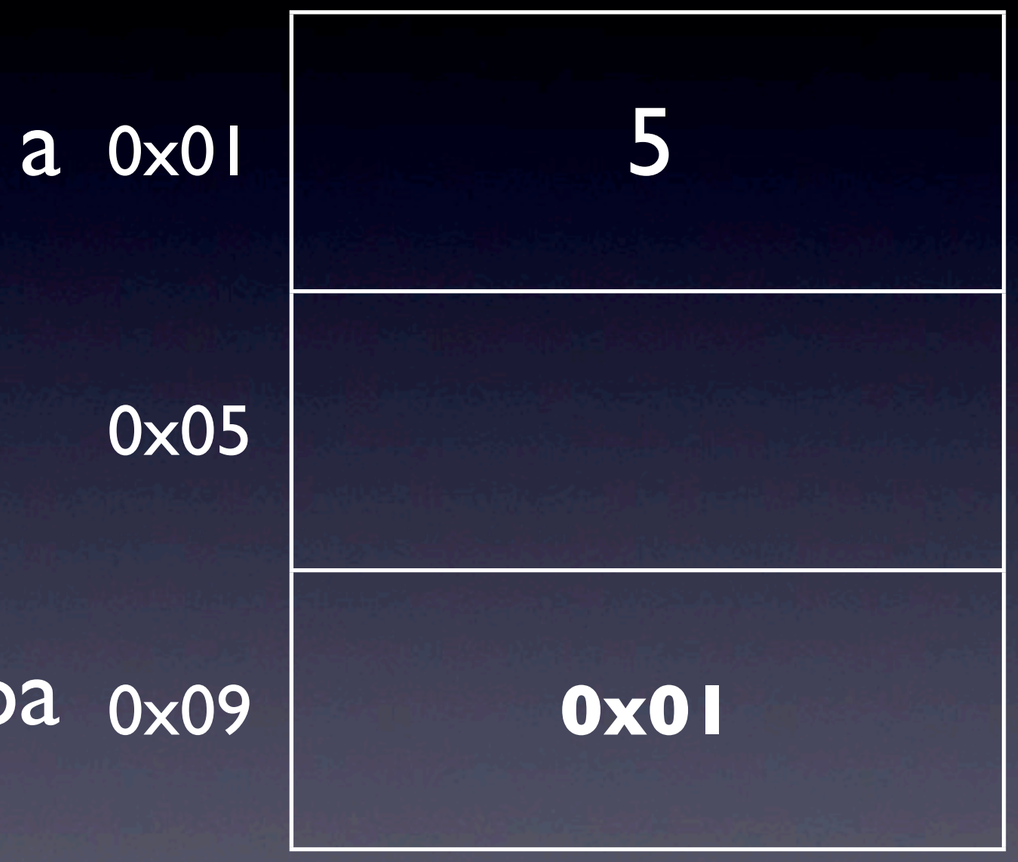
# Pointer




```
int a;
a=5;
int * pa;
pa = &a;
```

**& => address of**

# Pointer - More



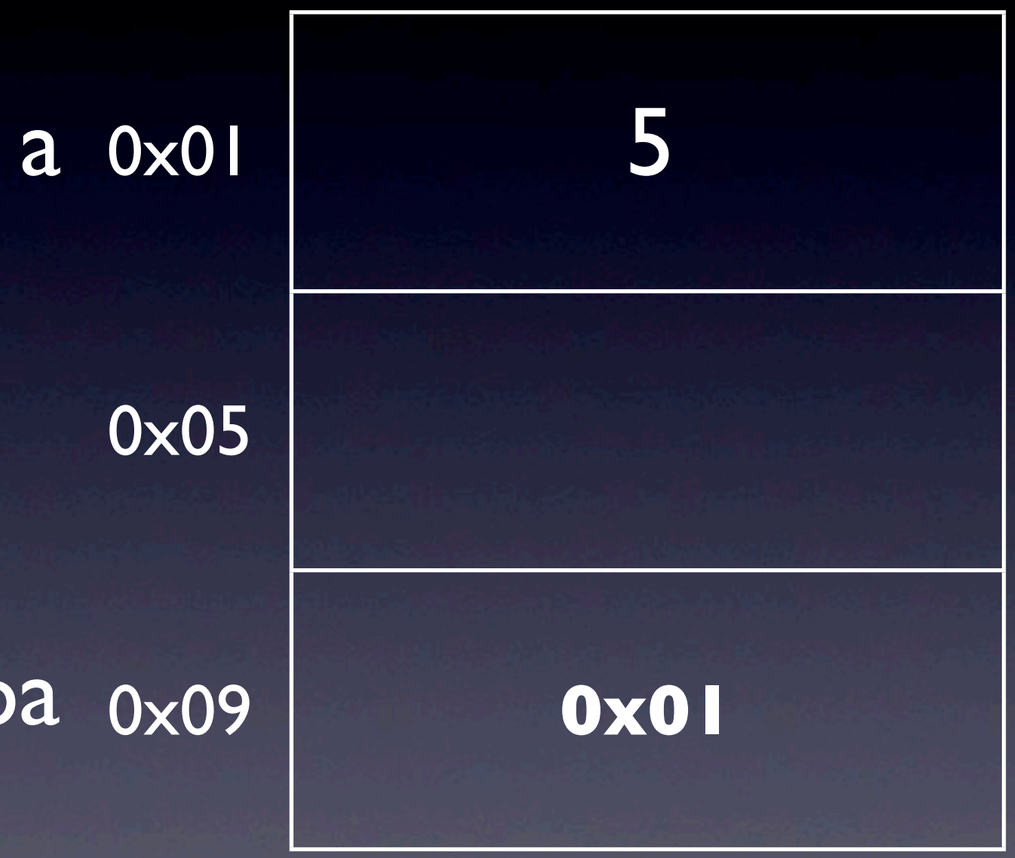

```
int a;
a=5;
int * pa;
pa = &a;
printf(“%d”,*pa);
```

# Recap - pointer

Write some thing

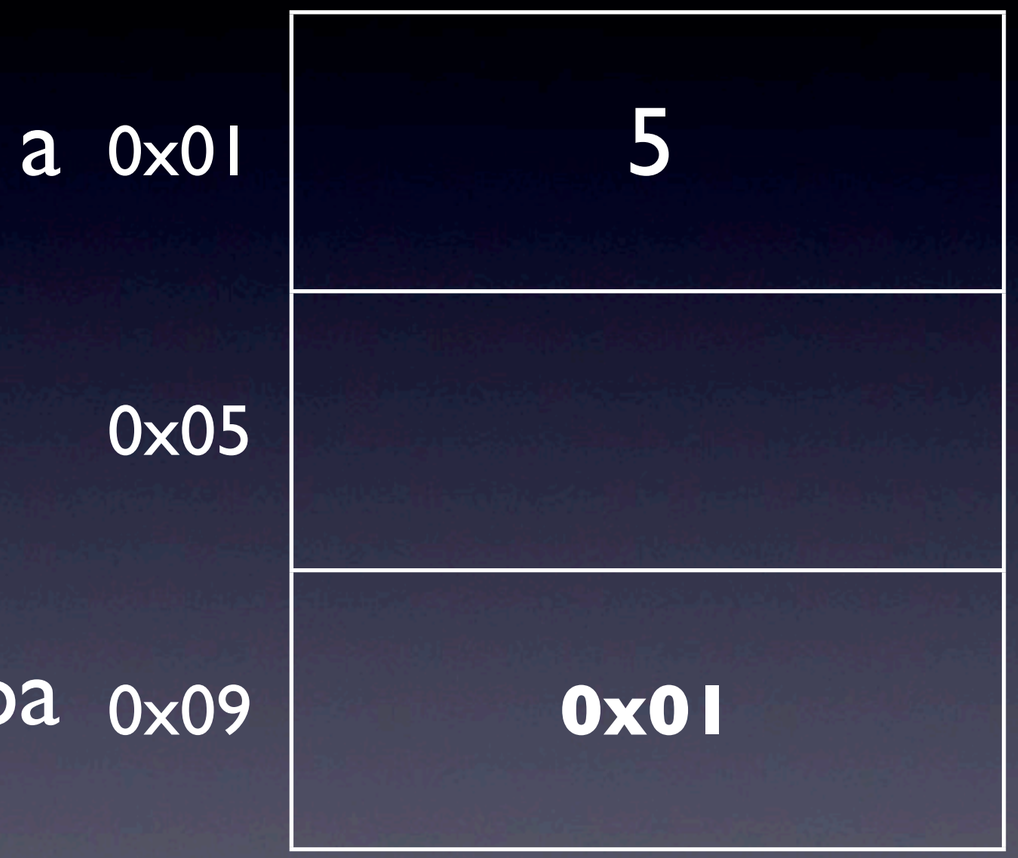
```
#import <Foundation/Foundation.h>

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

    @autoreleasepool {
        int a ;
        a = 5;
        int * pa ;
        pa = &a;
        printf("%d",*pa);

    }
    return 0;
}
```

# Pointer - More



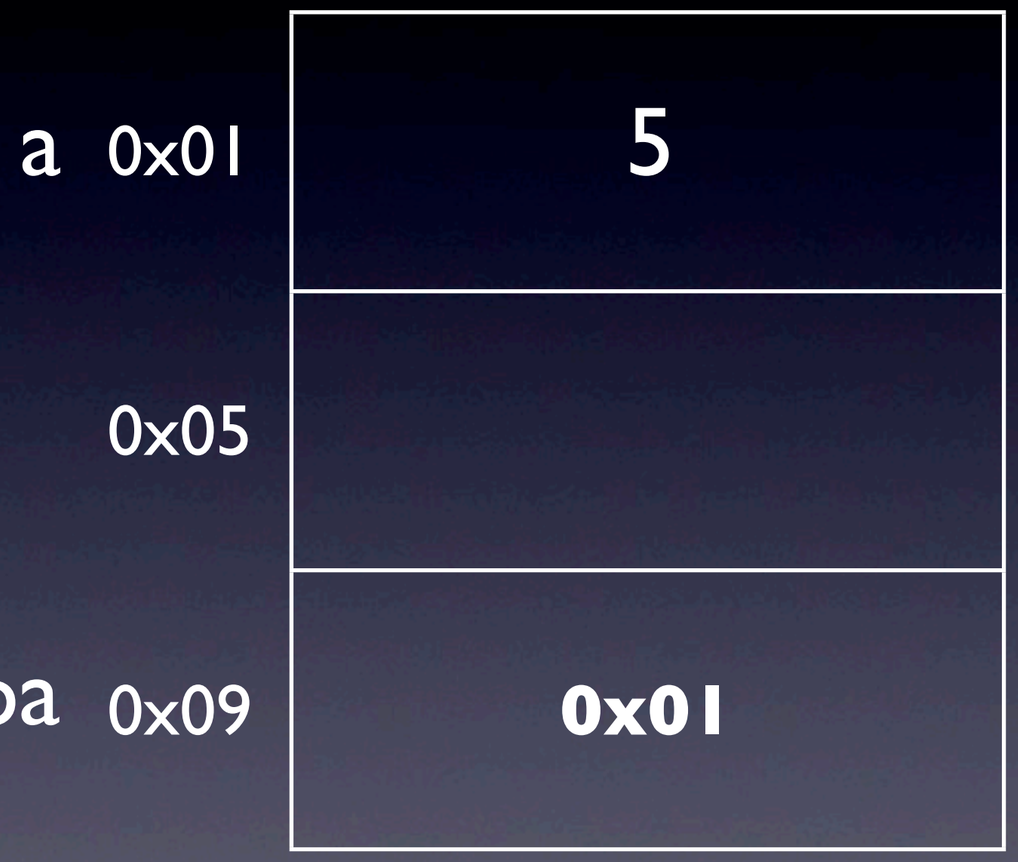

```
int a;
a=5;
int * pa;
pa = &a;
```

# Pointer - More



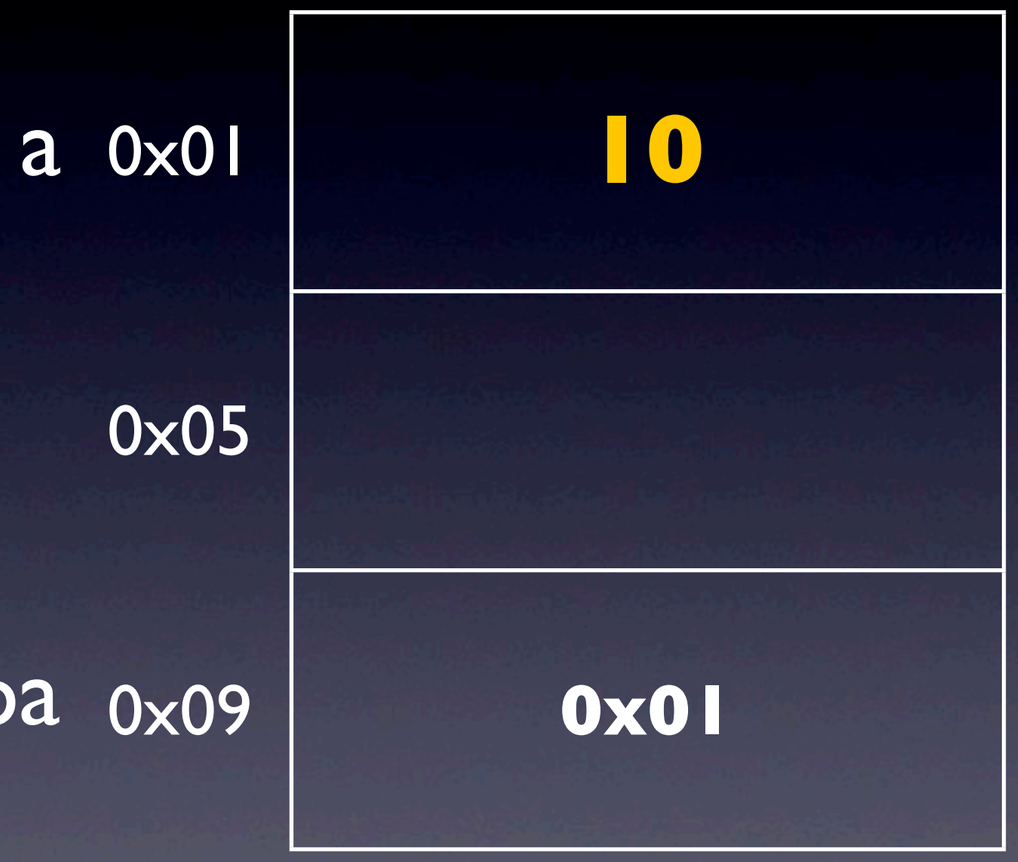

```
int a;
a=5;
int * pa;
pa = &a;
*pa = 10
```

# Pointer - More




```
int a;
a=5;
int * pa;
pa = &a;
*pa = 10
printf(“%d”,a);
```

# Pointer - More




```
int a;
a=5;
int * pa;
pa = &a;
*pa = 10
printf(“%d”,a);
```

# Struct

- More complicated type

```
struct Date {
    int day;
    int month;
    int year;
};

struct Date date = {3,10,1970};

printf("The day is %d, %d/%d", date.year, date.day, date.month);
```

# Struct - Memory

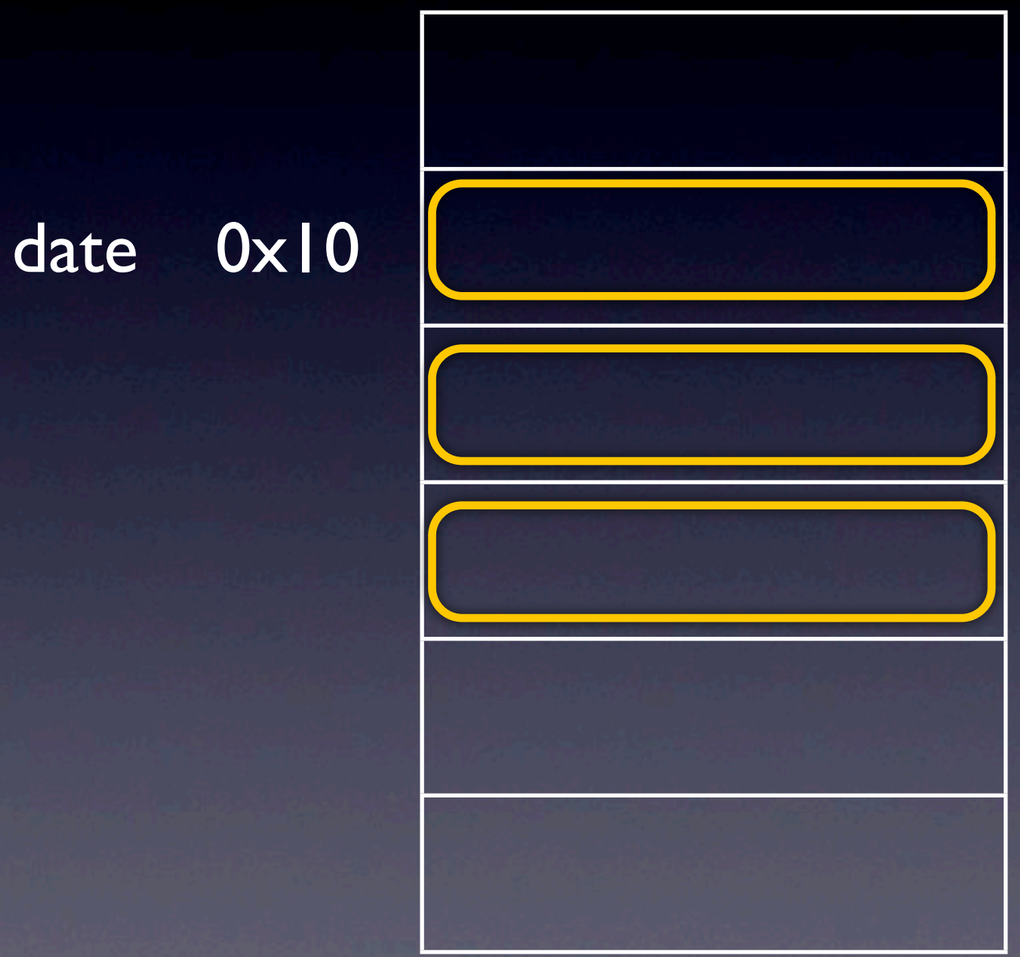

```
struct Date {
    int day;
    int month;
    int year;
};
```

# Struct - Memory


```
struct Date date ;
```

```
struct Date {
    int day;
    int month;
    int year;
};
```

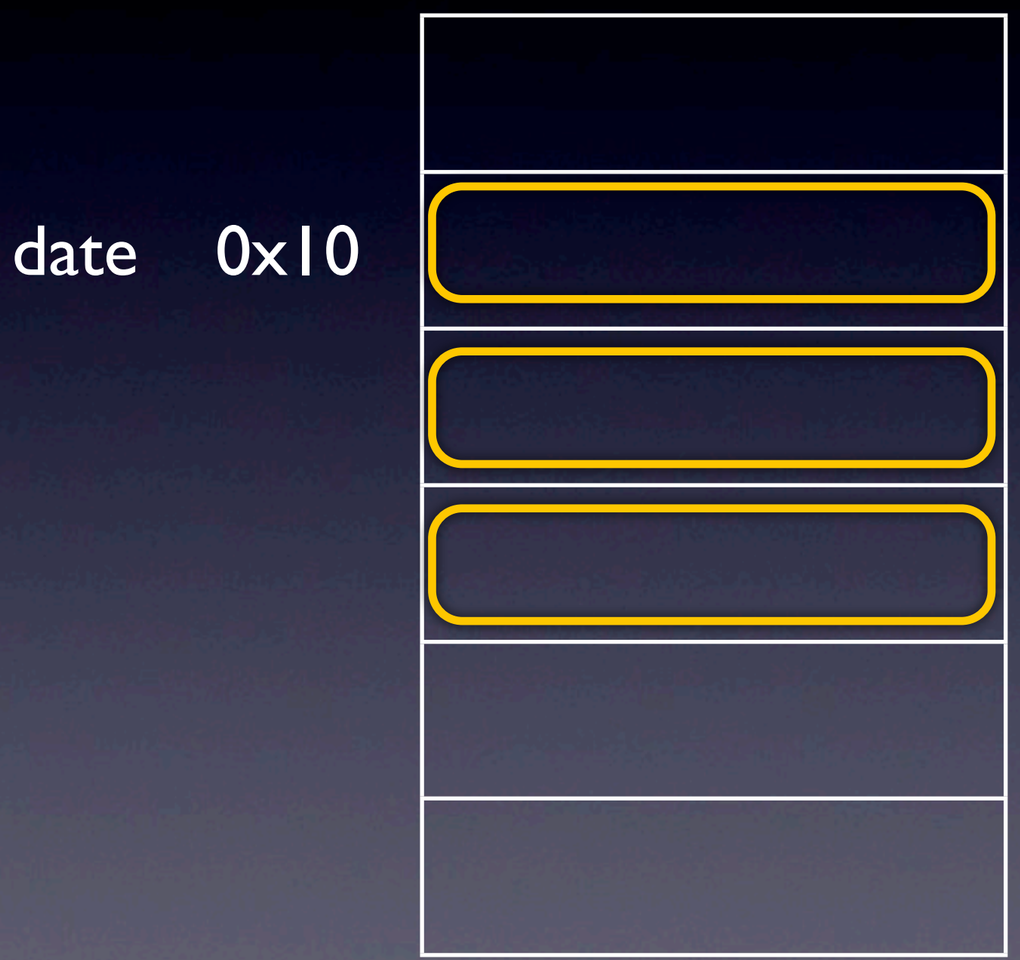
# Struct - Memory




```
struct Date date ;
```

```
struct Date {
    int day;
    int month;
    int year;
};
```

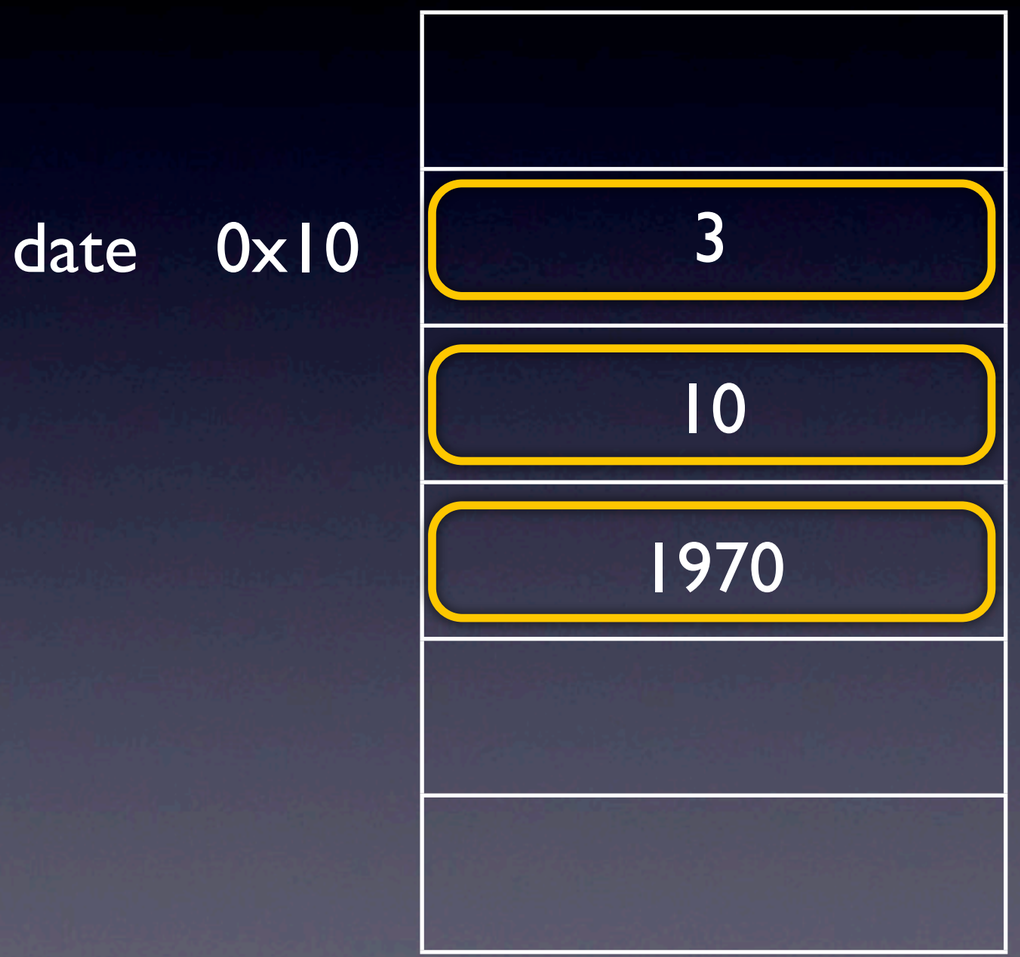
# Struct - Memory




```
struct Date date ;

date = {3,10,1970};
```

```
struct Date {
    int day;
    int month;
    int year;
};
```

# Struct - Memory




```
struct Date date ;

date = {3,10,1970};
```

```
struct Date {
    int day;
    int month;
    int year;
};
```

# Function

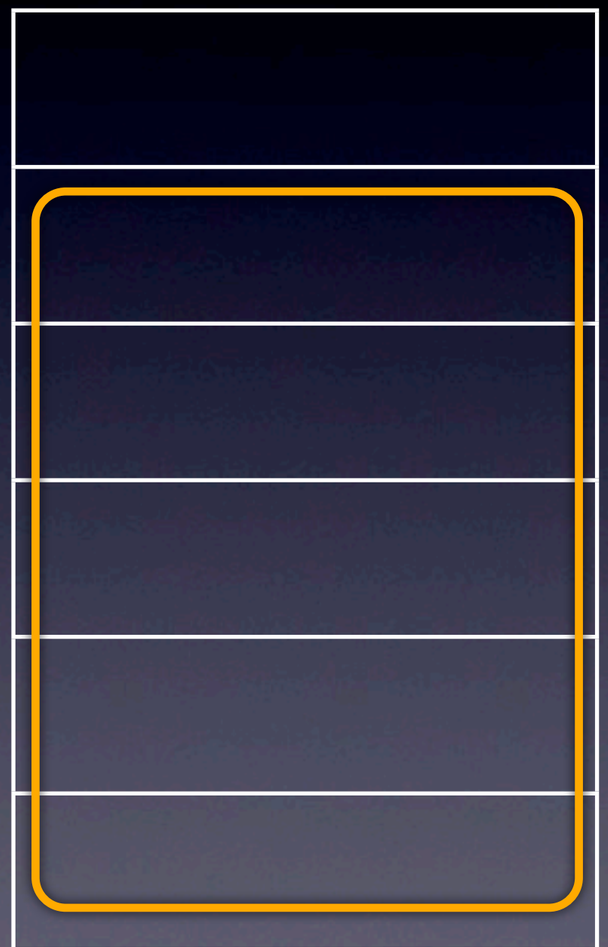
- we know main function
- make another function

```
18 int add(int a, int b){
19     int result = a+b;
20     return result;
21 }
22
23 int main(int argc, const char * argv[])
24 {
25
26     @autoreleasepool {
27         int sum = add(5,6);
28         printf("%d",sum);
29     }
30     return 0;
31 }
```

# Function - Memory

add

result

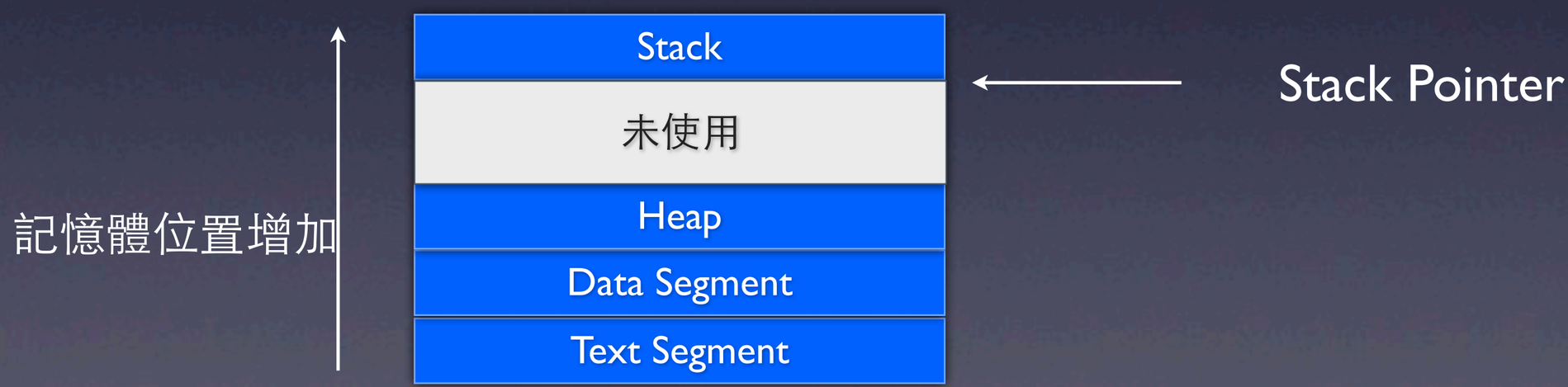


```
int add(int a, int b){
    int result = a+b;
    return result;
}
```

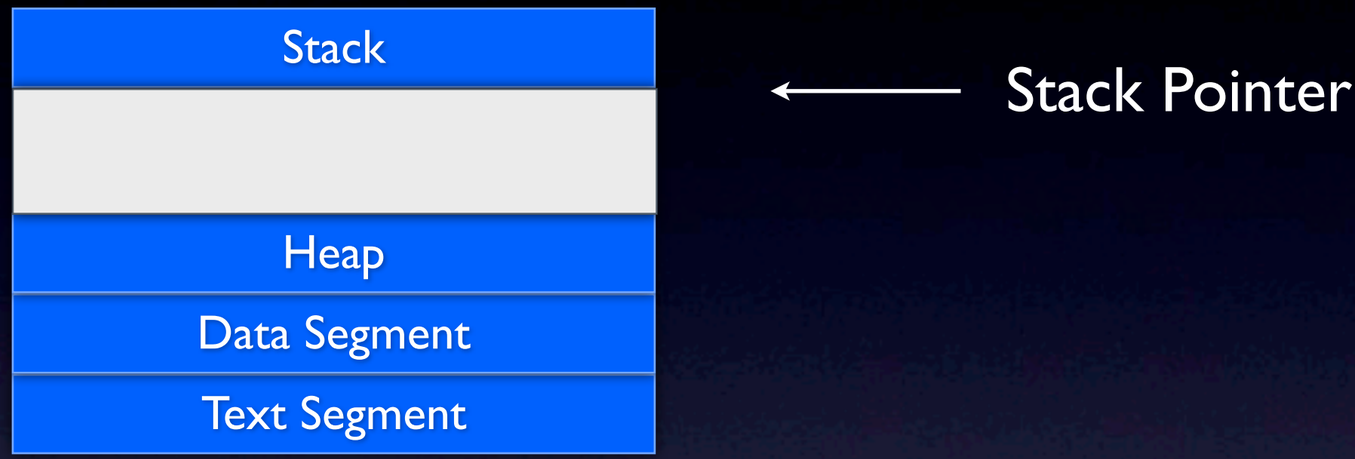
in **Stack**

# Objective-C 程式記憶體配置

- Text Segment - 唯讀執行區
- Data Segment - 可讀寫區包含 global 變數
- Heap - 可依程式需要產生和消除記憶體(動態配置- malloc)
- Stack - 為了 **function** 產生，可變動大小，包含區域變數

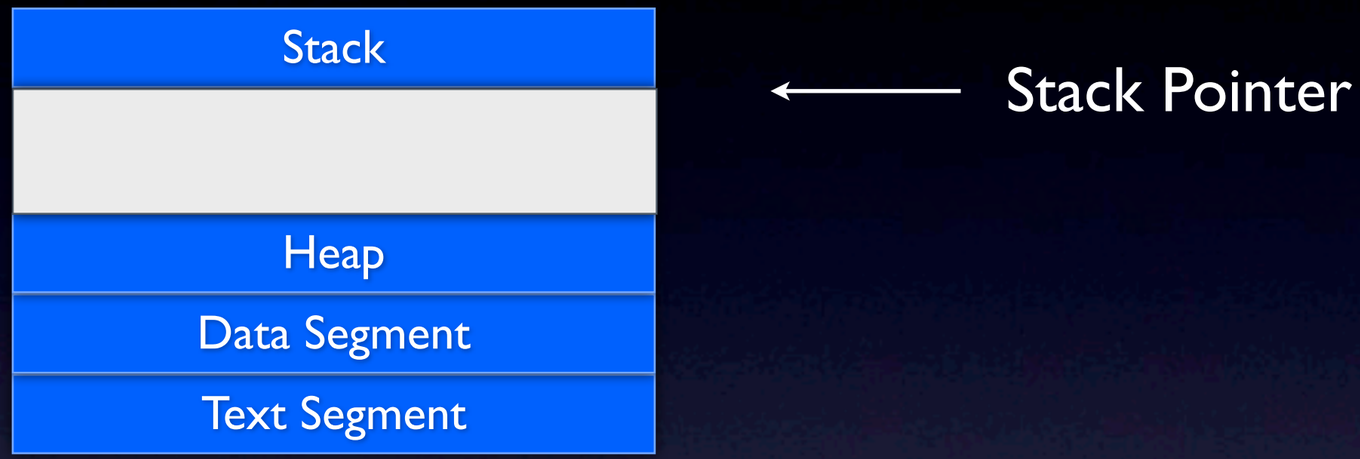


# Stack 與 Function



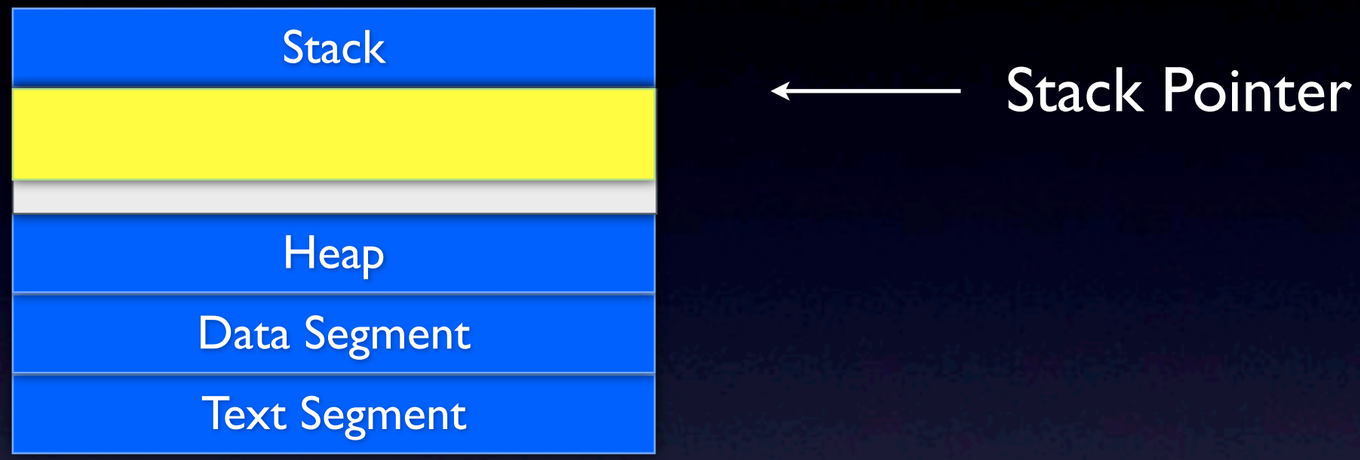

```
int main (int argc, const char * argv[]) {
    // ... 省略



}
```

# Stack 與 Function



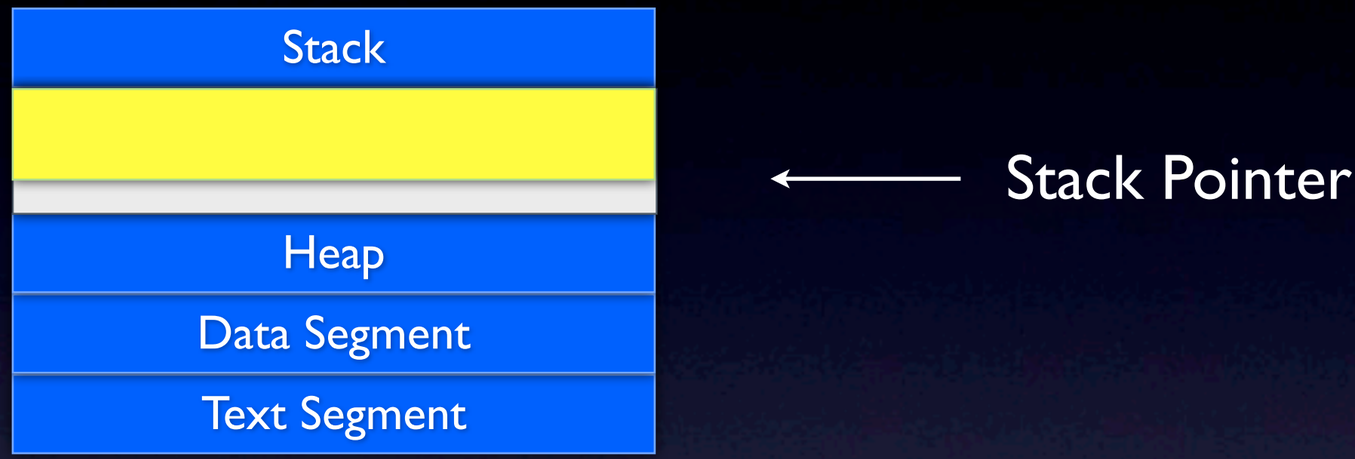

```
int main (int argc, const char * argv[]) {
    // ... 省略
    int sum = add(5, 6);


}
```

# Stack 與 Function



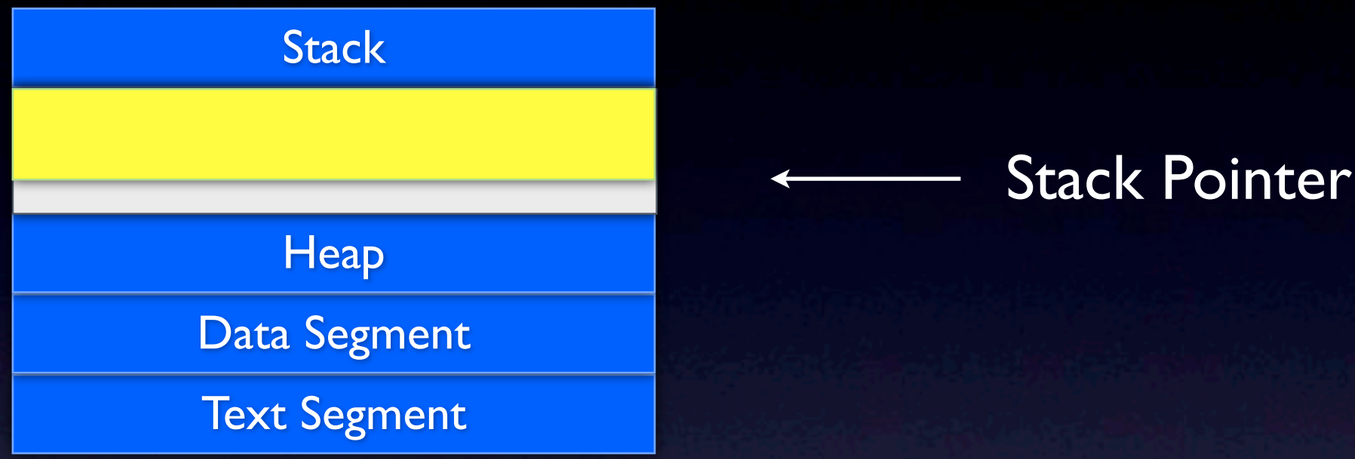

```
int main (int argc, const char * argv[]) {
    // ... 省略
    int sum = add(5, 6);


}
```

# Stack 與 Function



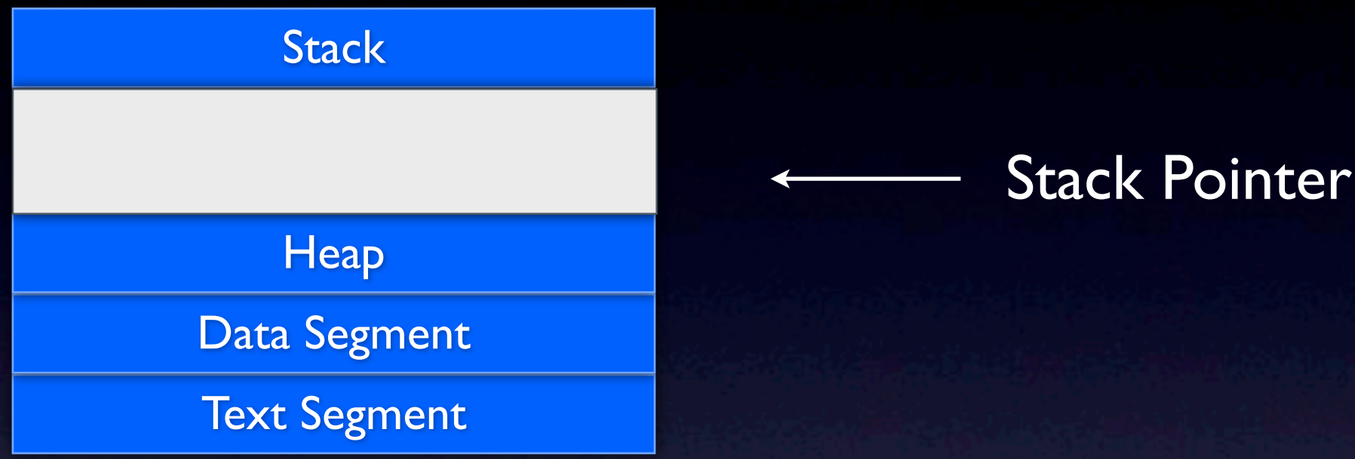

```
int main (int argc, const char * argv[]) {
    // ... 省略
    int sum = add(5, 6);


}
```

# Stack 與 Function



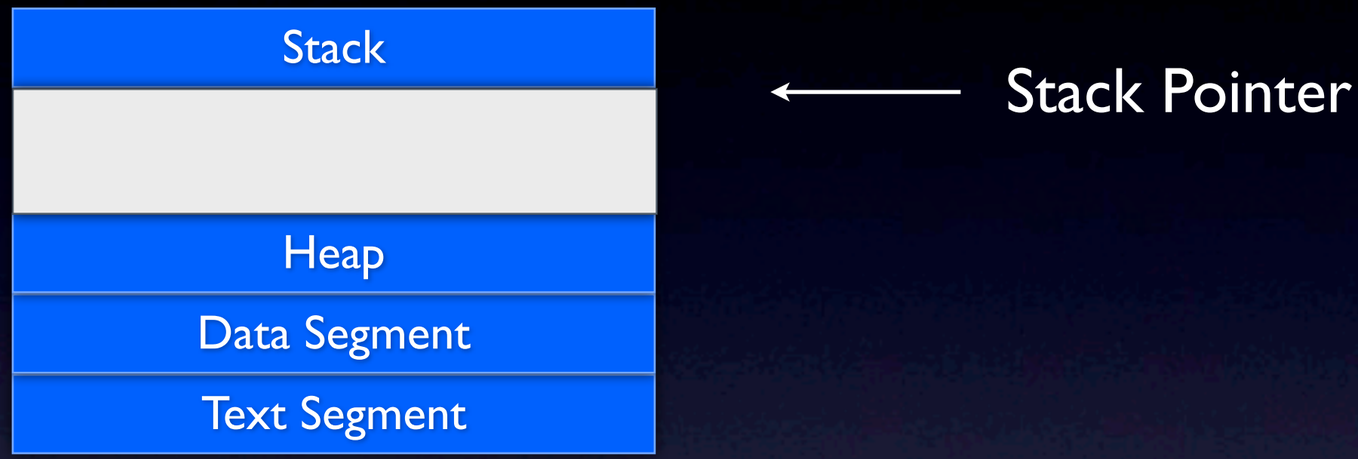

```
int main (int argc, const char * argv[]) {
    // ... 省略

    int sum = add(5, 6);

    // 下一行程式碼
}
```

# Stack 與 Function



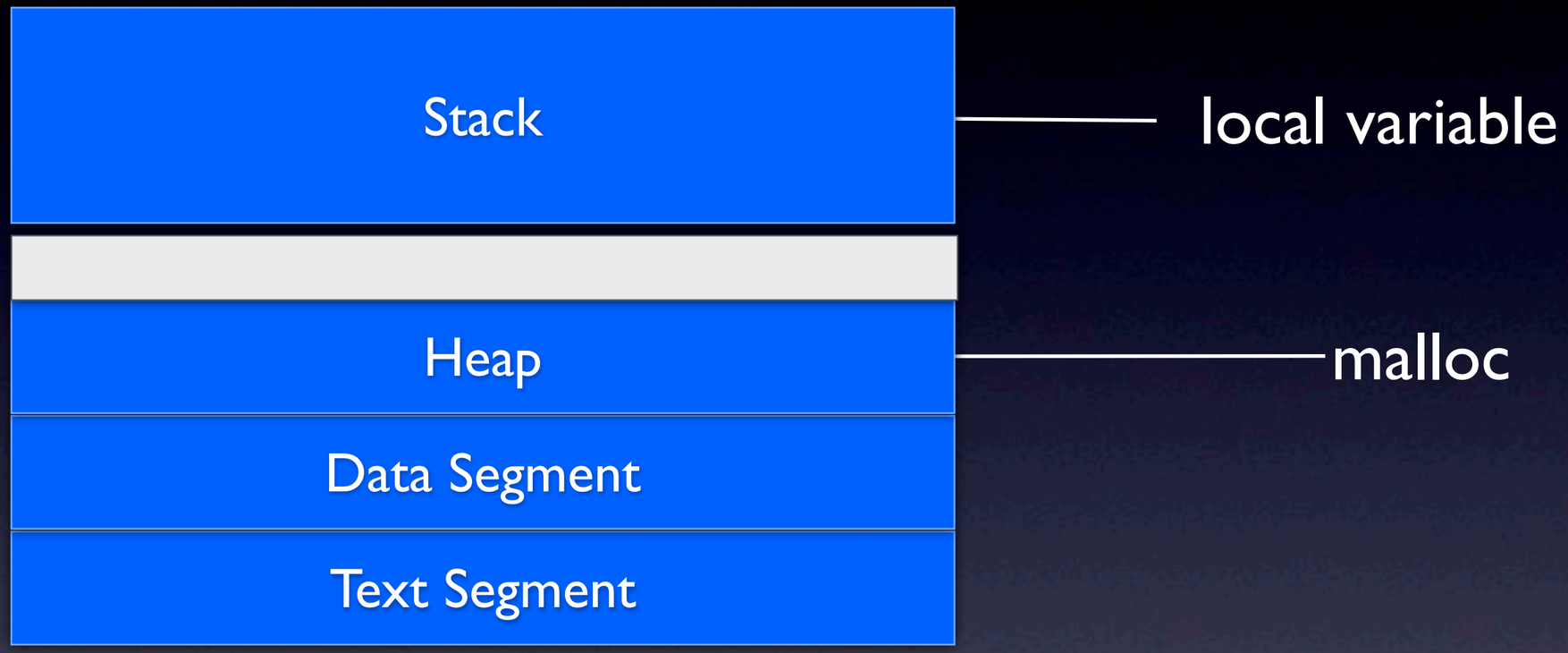

```
int main (int argc, const char * argv[]) {
    // ... 省略

    int sum = add(5, 6);

    // 下一行程式碼
}
```

# Stack 與 Function




```
int main (int argc, const char * argv[]) {
    // ... 省略

    int sum = add(5, 6);

    // 下一行程式碼
}
```

# Stack 與 Heap

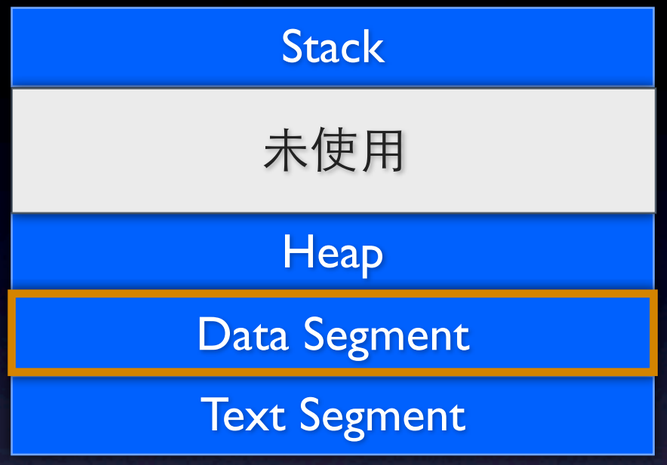

Stack
local variable
Heap
malloc
Data Segment
Text Segment

# Pointer - 動態配置記憶體

```
#include <stdlib.h>

int * p_a;
p_a = malloc(sizeof(int));
*p_a = 8;
printf("value in pointer a is %d\n", *p_a);
```

# Static variable

- in Data Segment




```
int callStaticValue();

int main (int argc, const char * argv[]) {
  callStaticValue();
  callStaticValue();
  printf("static variable is %d", callStaticValue());
  return 0;
}

int callStaticValue(){
   static int sVar=0; // static 變數初始化只會做一次
   return sVar++;  // 一直加1
}
```

# Struct + function

- it is what we call **Object**

```
struct Date {
 int day;
 int month;
 int year;
 void (*formatedDate)( struct Date  date);
};

void formatedFunction(struct Date  date){
    printf("The day is %d, %d/%d",date.year, date.month, date.day );
}
```

```
// in main()
struct Date  today = {29, 4, 2013};
today.formatedDate = formatedFunction;
today.formatedDate(today);
```

# Objective-C class - C struct

- /usr/include/objc/objc.h

```
typedef struct objc_class *Class;
typedef struct objc_object {
    Class isa;
} *id;

typedef struct objc_selector  *SEL;

typedef id (*IMP)(id, SEL, ...);
```

# SEL

```
struct objc_selector
{
  void *sel_id;
  const char *sel_types;
};
```

# Objective-C class - C struct

- /usr/include/objc/runtime.h

```
struct objc_class {
    Class isa;

#if !__OBJC2__
    Class super_class                                       OBJC2_UNAVAILABLE;
    const char *name                                        OBJC2_UNAVAILABLE;
    long version                                            OBJC2_UNAVAILABLE;
    long info                                               OBJC2_UNAVAILABLE;
    long instance_size                                      OBJC2_UNAVAILABLE;
    struct objc_ivar_list *ivars                            OBJC2_UNAVAILABLE;
    struct objc_method_list **methodLists                   OBJC2_UNAVAILABLE;
    /* ignore */
#endif

} OBJC2_UNAVAILABLE;
/* Use `Class` instead of `struct objc_class *` */
/* ignore */
typedef struct {
    const char *name;
    const char *value;
} objc_property_attribute_t;
```

# Method

```
struct objc_method {
    SEL method_name                                          OBJC2_UNAVAILABLE;
    char *method_types                                       OBJC2_UNAVAILABLE;
    IMP method_imp                                           OBJC2_UNAVAILABLE;
}                                                            OBJC2_UNAVAILABLE;

struct objc_method_list {
    struct objc_method_list *obsolete                        OBJC2_UNAVAILABLE;

    int method_count                                         OBJC2_UNAVAILABLE;
#ifdef __LP64__
    int space                                                OBJC2_UNAVAILABLE;
#endif
    /* variable length structure */
    struct objc_method method_list[1]                        OBJC2_UNAVAILABLE;
}                                                            OBJC2_UNAVAILABLE;
```

# About Message

- Add method runtime

# Add method runtime

```
struct objc_method {
  SEL method_name;
  char *method_types;
  IMP method_imp;
};
```

```
struct objc_method_list {
  struct objc_method_list *obsolete;
  int method_count;
  struct objc_method method_list[1];
};
```

# Add method

```
#import <objc/objc-class.h>

// create a class with no methods
@interface EmptyClass : NSObject { }
@end

@implementation EmptyClass
@end

// define the function to add as a method
id sayHello ( id self, SEL _cmd,... )
{
  NSLog (@"Hello");
}
```

```
void addMethod ()
{
  // create the method

  struct objc_method myMethod;
  myMethod.method_name = sel_registerName("sayHello");
  myMethod.method_imp  = sayHello;

  // build the method list.
  // this memory needs to stick around as long as the
  // methods belong to the class.
  struct objc_method_list * myMethodList;
  myMethodList = malloc (sizeof(struct objc_method_list));
  myMethodList->method_count = 1;
  myMethodList->method_list[0] = myMethod;

  // add method to the class
  class_addMethods ( [EmptyClass class], myMethodList );

  // try it out
  EmptyClass * instance = [[EmptyClass alloc] init];
  [instance sayHello];
  [instance release];
}
```

# Method under the hood

```
MyClass * aObj;
int para = 5;
[aObj setCount:para];
```

```
objc_msgSend( aObj, @selector( setCount: ), para );
```

經由 isa 找到 Class structure

從table 找到相對應的 function

| selector | function |
|---|---|
| **setCount:** | ... |
| count | ... |

# Demo

- DemoNCCU13 - MyCObject

# Dynamic feature

- Dot operation
- KVC

# Convenient Way

- Since Objective-C 2.0
- Dot Syntax

```
NSString *name = person.name;
//             name = [person name]

      person.name = @"Michael";
//  [person setName:@"Michael"]
```

- Cascade

```
      person.bill.name = @"Andy";
// [[person bill] setName:@"Andy"]
      person.bill.name ;
// [[person bill] name]
```

# Property means method

```
@interface Person : NSObject

@property int age;
@property (strong) NSString * name;

@end

@implementation Person
@end
```

```
Person * person = [[Person alloc] init];
person.name = @”Michael”;
person.age = 35;
```

# Key-Value Coding

# Another access method

- Read
  - person.name;
  - [person name];
  - [person **valueForKey**:@”name”];

- Write
  - person.name = @”michael”;
  - [person setName:@”michael”];
  - [person **setValue**:@”michael” **forKey**:@”name”];

# Finding rule

- Find the action **-name** or **-setName**
- Find real variable **_name** and then **name**

**forKey**:@"name"

# Accessible Types

- attribute
  - scalar // auto-boxing for KVC
  - NSString
  - Boolean
  - Immutable object : NSColor, NSNumber
- to-many relationship - objects in NSArray, NSSet

# Recap

```
@interface MyObject : NSObject {
    NSString * name;
    NSString * _name;
}

@property (retain, readonly) NSString * _name;
@property (retain, readonly) NSString * name;
@end
```

```
@implementation MyObject
  @synthesize name, _name;
@end
```

```
MyObject * myo = [MyObject new];
[myo setValue:@"michael" forKey:@"name"];
NSLog(@" kvc %@",[myo valueForKey:@"name"]);
```

Result : kvc (null) ?

0. no setter
1. find out _name = @”michael”
2. find out name’s getter, return name

下底線開頭的變數要小心使用

# KeyPath

[myCar valueForKeyPath: @"**engine.vendor**"]

```
@interface MyCar : NSObject {
    Engine * engine;
}

@end
```

```
@interface Engine : NSObject {
    NSString * vendor;
}

@end
```

# Array Operation

NSNumber * count;

count = [myCars valueForKeyPath: @"**@avg.price**"]

**@sum**.xxx
**@min**.xxx
**@max**.xxx

```
@interface MyCar : NSObject {
    int price;
}

@end
```

myCars = | myCar1 | myCar2 | myCar3 |

# Recap

```
@interface Engine : NSObject {
    int  price;
}

@end
```

```
// 計算 engines array裡平均的price
NSArray * engines = [???:myengine1, myengine2, myengine3, nil];
NSLog(@" average price is %@",[engines valueForKeyPath:@"???"]);
```

# nil for scalar value ?

[engine setValue: nil forKey:@"price"]

**error !!**

overwrite **-setNilValueForKey:**

```
- (void) setNilValueForKey: (NSString *) key {
    if ([key isEqualToString: @"price"]) {
        price = 0;
    } else {
        [super setNilValueForKey: key];
    }
}
```

# Handle UndefinedKey

```
- (void) setValue: (id) value forUndefinedKey: (NSString *) key {
    // do something
}
```

```
- (id) valueForUndefinedKey:(NSString *)key {
    // do something
}
```

# Recap

實作

- (void) setValue: (id) value forUndefinedKey: (NSString *) key;

- (id) valueForUndefinedKey:(NSString *)key;

# Reference

Introduction to Key-Value Coding Programming Guide

http://goo.gl/rrfmy

# 實例 - 解析 RSS

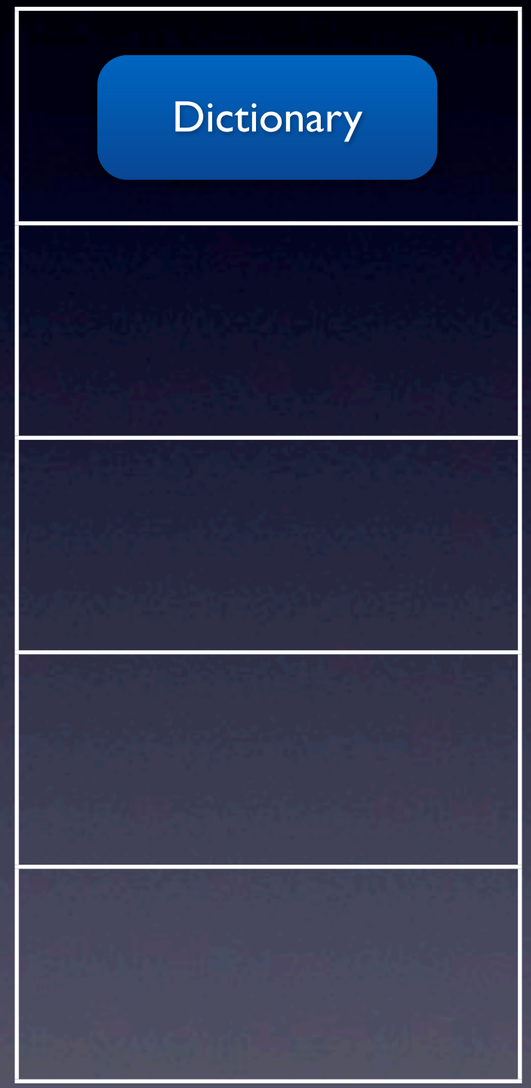
網路上免費資源 - 多用來呈現新聞

# With XML - RSS-like

```
<car>
    <item>
        <price> 99 </price>
        <year> 1922 </year>
    </item>
    <item>
        <price> 30 </price>
        <year> 1990 </year>
    </item>
    <item>
        <price> 82 </price>
        <year> 1980 </year>
    </item>
    <item>
        <price> 75 </price>
        <year> 1920 </year>
    </item>
    <item>
        <price> 60 </price>
        <year> 1910 </year>
    </item>
</car>
```

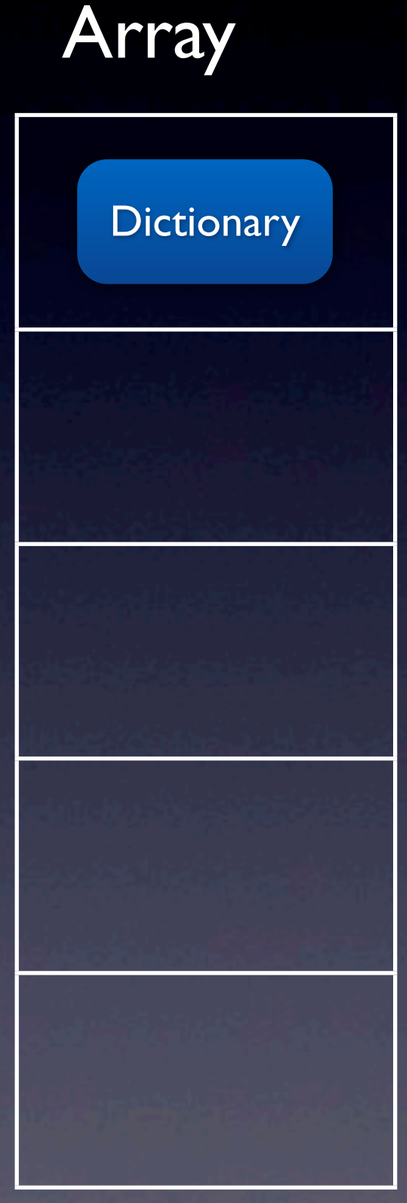
| key | value |
|---|---|
| price | 99 |
| year | 1922 |

Array



# Array and KVC

```
<car>
    <item>
        <price> 99 </price>
        <year> 1922 </year>
    </item>
    <item>
        <price> 30 </price>
        <year> 1990 </year>
    </item>
    <item>
        <price> 82 </price>
        <year> 1980 </year>
    </item>
    <item>
        <price> 75 </price>
        <year> 1920 </year>
    </item>
    <item>
        <price> 60 </price>
        <year> 1910 </year>
    </item>
</car>
```




```
(
    99,
    30,
    82,
    75,
    60,
)
```

[array **valueForKey**:@”price”]

# Demo

ValueAndPredicate

# Question