01、第一个OC的类

**Car.h**

//因为使用了Nsobject

#import <Foundation/Foundation.h>

/\*

类名：car

属性：轮胎个数、时速（速度）

行为：跑

\*/

//类的声明

//声明对象的属性和方法

//:Nsobject目的是让Car这个类具备创建对象的能力

@interface Car : NSObject

{

//用来声明对象属性（实例变量或成员变量）

@public //可以让外部的指针间接访问对象内部的成员变量

int wheels; //轮胎个数

int speed; //时速

}

//方法（行为）:方法名、参数、返回值（声明、实现）

//只要是OC对象的方法，必须以 - 开头

//OC方法中任何数据类型必须用小括号（）括住

//OC方法中的小括号（）：括住数据类型

-(void)run;

@end

**Car.m**

#import "Car.h"

//类的实现

@implementation Car

-(void)run

{

//方法的实现（说清楚方法里面有什么代码）

NSLog(@"car is running!");

}

@end

**main.m**

#import <Foundation/Foundation.h>

#import "Car.h"

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

//OC中想执行一些行为，就写上一个中括号[行为执行者 行为名称]

//利用类来创建对象

//执行Car这个类的new行为来创建新对象

//[Car new]会创建出用一个新对象，并且会返回新对象的地址

Car \*p = [Car new];

p->wheels = 4;

p->speed = 220;

NSLog(@"wheels is %d and speed is %d",p->wheels,p->speed);

[p run];

return 0;

}

02 OC对象和函数

**car.h**

#import <Foundation/Foundation.h>

@interface Car : NSObject

{

@public

int wheels;

int speed;

}

-(void)run;

@end

**car.m**

#import "Car.h"

@implementation Car

-(void)run

{

NSLog(@"%d个轮子，速度为%d的车子跑起来了",wheels,speed);

}

@end

**main.m**

/\*

方法和函数的区别：

方法：

1、对象方法都是以-开头

2、对象方法的声明必须写在@interface和@end之间

对象方法的事项必须写在@implement和@end之间

3、对象方法只能由对象来调用

4、对象方法归类/对象所有

函数：

1、函数能写在文件中的任意位数，函数归文件所有

2、函数调用不依赖于对象

3、函数内部不能直接通过成员变量名访问某个对象的成员变量

\*/

#import <Foundation/Foundation.h>

#import "Car.h"

void test(int w,int s)

{

w = 20;

s = 200;

}

void test1(Car \*newC)

{

newC->wheels = 5;

}

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

Car \*c = [Car new];

c->wheels = 4;

c->speed = 250;

//test(c->wheels,c->speed);

test1(c);

[c run];

return 0;

}

03 类的合理设计

**student.h**

#import <Foundation/Foundation.h>

#import "Dog.h"

/\*

学生

成员变量：性别、生日、体重、最喜欢的颜色、狗（体重、吃、跑）

方法：吃、跑步、遛狗（让狗跑）、喂狗（让狗吃）

\*/

typedef enum{

SexMan,

SexWoman

} Sex;

typedef struct{

int year;

int month;

int day;

} Date;

typedef enum{

ColorBlack,

ColorRed,

ColorGreen

} Color;

@interface Student : NSObject

{

@public

Sex sex;

Date birthday;

double weight;

Color favColor;

char \*name;

Dog \*dog;

}

-(void)eat;

-(void)run;

-(void)print;

-(void)walkDog;

-(void)feedDog;

@end

**student.m**

**#import "Student.h"**

**@implementation Student**

**-(void)eat**

**{**

**//每吃一次，体重加1**

**weight += 1;**

**NSLog(@"学生吃完这次后体重是%f",weight);**

**}**

**-(void)run**

**{**

**weight -= 1;**

**NSLog(@"学生跑完这次后的体重是%f",weight);**

**}**

**-(void)print**

**{**

**NSLog(@"性别=%d，喜欢的颜色=%d，姓名=%s，生日=%d-%d-%d",sex,favColor,name,birthday.year,birthday.month,birthday.day);**

**}**

**-(void)walkDog**

**{**

**//调用狗的run方法**

**[dog run];**

**}**

**-(void)feedDog**

**{**

**//调用狗的eat方法**

**[dog eat];**

**}**

**@end**

**dog.h**

#import <Foundation/Foundation.h>

@interface Dog : NSObject

{

@public

double weight;

}

-(void)run;

-(void)eat;

@end

**dog.m**

**#import "Dog.h"**

**@implementation Dog**

**-(void)eat**

**{**

**//每吃一次，体重加1**

**weight += 1;**

**NSLog(@"狗吃完这次后体重是%f",weight);**

**}**

**-(void)run**

**{**

**weight -= 1;**

**NSLog(@"狗跑完这次后的体重是%f",weight);**

**}**

**@end**

**main.m**

#import <Foundation/Foundation.h>

#import "Student.h"

#import "Dog.h"

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

Student \*s = [Student new];

Dog \*d = [Dog new];

d->weight = 20;

s->dog = d;

s->weight = 50;

s->sex = SexMan;

//Date d = {2011,9,10};

//s->birthday = d;

s->birthday.year = 2011;

s->birthday.month = 9;

s->birthday.day = 10;

s->favColor = ColorBlack;

s->name = "jack";

[s eat];

[s eat];

[s run];

[s run];

[s print];

[s walkDog];

[s feedDog];

return 0;

}

**04 匿名对象**

**car.h**

#import <Foundation/Foundation.h>

@interface Car : NSObject

{

@public

int speed;

}

-(void)run;

@end

**car.m**

#import "Car.h"

@implementation Car

-(void)run

{

NSLog(@"速度为%d的车子跑起来了",speed);

}

@end

**main.m**

#import <Foundation/Foundation.h>

#import "Car.h"

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

// //有名字的对象

// Car \*c = [Car new];

// c->speed = 250;

// [c run];

//匿名对象，了解即可，不要写匿名对象这样的代码，对内存管理没有好处

[Car new]->speed = 220;

[[Car new] run];

}

**05 方法的声明和实现**

**calculator.h**

#import <Foundation/Foundation.h>

/\*

计算器类

方法：

1、返回PI值

2、计算某个整数的平方

3、计算两个整数的和

\*/

@interface calculator : NSObject

//方法名：pi

-(double)pi;

//OC方法中，一个参数对应一个冒号

//方法名:square:(冒号也是方法名的一部分)

-(int)square:(int)num;

//方法名：sumWithNum1:andNum2:

-(int)sumWithNum1:(int)num1 andNum2:(int)num2;

@end

**calcator.m**

#import "calculator.h"

@implementation calculator

-(double)pi

{

return 3.14;

}

-(int)square:(int)num

{

return num\*num;

}

-(int)sumWithNum1:(int)num1 andNum2:(int)num2

{

return num1 + num2;

}

@end

**main.m**

#import <Foundation/Foundation.h>

#import "calculator.h"

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

calculator \*cal = [calculator new];

double a = [cal pi];

NSLog(@"pi is %f",a);

int b = [cal square:10];

NSLog(@"square is %d",b);

int c = [cal sumWithNum1:10 andNum2:5];

NSLog(@"sum is %d",c);

return 0;

}

}