**课 程 设 计 报 告**

**课程设计名称：IOS移动开发设计**

**课程设计题目**：计算器的编写

院（系）：计算机与信息工程学院

专    业：计算机科学与技术（网络编程）

班    级：2014网络编程

学    号：20141105069

姓    名：温馨

指导教师：朝力萌

完成日期：2016.7.13

[https://github.com/wwwwenxin/wx-s-calculator](https://github.com/wwwwenxin/wx-s-calculator" \t "https://set2.mail.qq.com/cgi-bin/_blank)

**目 录**

[第1章 概要设计 3](#_Toc406624046)

[1.1题目的内容与要求 3](#_Toc406624047)

[1.2总体结构](#_Toc406624048) 4

[第2章 详细设计](#_Toc406624049) 5

[2.1主模块](#_Toc406624050) 5

[第3章 调试分析](#_Toc406624051) 20

[第4章 使用说明与执行结果](#_Toc406624052) 22

第1章  概要设计

**1.1题目的内容与要求**

**内容：在 IOS X-code环境下运用新编程语言Swift进行计算器的编写。先创建一个新的X-code工程，选择IOS中Application的Single View Application（单视图应用），语言为Swift。创建好后开始计算器的编写。首先在视图控制器中拖拽text field添加一个输入框，并拖拽Button添加数字键与加减乘除等运算符的按键；运行无误后通过类向导对输入框与各个按钮添加变量进行不同的消息与命令的处理，先定义变量temp，临时储存第一个输入的数字，再定义变量flag，标记运算状态。再对数字和运算符进行绑定，在里面添加**运算**使得按钮在按下的瞬间可以运行代码中所编写的函数，进行计算器的基本功能的运行。运用Swift语言与以往运用其他语言编写程序有些不同，它是一种新的编程语言，用于编写 iOS 和 OS X 应用，程序编写时不需要引入头文件，也不需要在每一句后加分号；使用let声明常量，使用var声明变量。声明时类型是可选的，声明的同时赋值的话，编译器会自动推断类型。运用Swift语言使编程更简单，更灵活，也更有趣。**

**要求：**

**（1）编译语言为Swift；**

**（2）计算器完成后可以实现计算器的加减乘除、开根号、百分号、倒数、清零等基本功能；**

**1.2总体结构**

class ViewController: UIViewController {

    override func viewDidLoad() {

        super.viewDidLoad()

        // Do any additional setup after loading the view, typically from a nib.

    }

    override func didReceiveMemoryWarning() {

        super.didReceiveMemoryWarning()

        // Dispose of any resources that can be recreated.

    }

    var temp=0.0

    var flag:Int=0;

    var point=true;

  //  var num1:Double=0;

   // var num2:Double=0;

  //  var equal:Double=0;

 @IBOutlet weak var wx: UITextField!

 @IBAction func 数字以及各个运算符(sender: AnyObject) {

  }

第2章  详细设计

**2.1主模块**

**2.1.1计算器中数字键的编写**

 @IBAction func wx0(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"";

        }

        else

        {

            wx.text=(wx.text)!+"0";

        }

    }

    @IBAction func wx1(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"1";

        }

        else

        {

            wx.text=(wx.text)!+"1";

        }

    }

    @IBAction func wx2(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"2";

        }

        else

        {

            wx.text=(wx.text)!+"2";

        }

    }

    @IBAction func wx3(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"3";

        }

        else

        {

            wx.text=(wx.text)!+"3";

        }

    }

    @IBAction func wx4(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"4";

        }

        else

        {

            wx.text=(wx.text)!+"4";

        }

    }

    @IBAction func wx5(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"5";

        }

        else

        {

            wx.text=(wx.text)!+"5";

        }

    }

    @IBAction func wx6(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"6";

        }

        else

        {

            wx.text=(wx.text)!+"6";

        }

    }

    @IBAction func wx7(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"7";

        }

        else

        {

            wx.text=(wx.text)!+"7";

        }

    }

    @IBAction func wx8(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"8";

        }

        else

        {

            wx.text=(wx.text)!+"8";

        }

    }

    @IBAction func wx9(sender: AnyObject) {

        if(wx.text=="0")

        {

            wx.text=(wx.text)!+"9";

        }

        else

        {

            wx.text=(wx.text)!+"9";

        }

    }

**2.1.2计算器中加的编写**

@IBAction func add(sender: AnyObject) {

        if flag==0{

            temp = (Double)(wx.text!)!

            wx.text=""

        }

        flag=1

    }

定义一个变量flag，赋初始值为0，将text文本中输入的内容强制转化为Double类型，将它储存在临时变量 temp中，将text文本置空，如果点击“+”按钮，自动将flag值变为1

**2.1.3计算器中减法的编写**

@IBAction func sub(sender: AnyObject) {

        if flag==0{

            temp = (Double)(wx.text!)!

            wx.text=""

        }

        flag=2

    }

定义一个变量flag，赋初始值为0，将text文本中输入的内容强制转化为Double类型，将它储存在临时变量 temp中，将text文本置空，如果点击 “-” 按钮，自动将flag值变为2

**2.1.4计算器中乘法的编写**

 @IBAction func muliply(sender: AnyObject) {

        if flag==0{

            temp = (Double)(wx.text!)!

            wx.text=""

        }

        flag=3

    }

定义一个变量flag，赋初始值为0，将text文本中输入的内容强制转化为Double类型，将它储存在临时变量 temp中，将text文本置空，如果点击“\*”按钮，自动将flag值变为3

**2.1.5计算器中除法的编写**

@IBAction func divide(sender: AnyObject) {

        if flag==0{

            temp = (Double)(wx.text!)!

            wx.text=""

        }

        flag=4

    }

定义一个变量flag，赋初始值为0，将text文本中输入的内容强制转化为Double类型，将它储存在临时变量 temp中，将text文本置空，如果点击“/”按钮，自动将flag值变为4

**2.1.6计算器中等号的编写**

@IBAction func equal(sender: AnyObject) {

        switch flag{

        case 1:

            temp=temp+(Double)(wx.text!)!

            wx.text="\(temp)"

        case 2:

            temp=temp-(Double)(wx.text!)!

            wx.text="\(temp)"

        case 3:

            temp=temp\*(Double)(wx.text!)!

            wx.text="\(temp)"

        case 4:

            if (Double)(wx.text!)!==0

            {

                wx.text="0不能为除数"

            }

            else{

                temp=temp/(Double)(wx.text!)!

                wx.text="\(temp)"

                }

        default:

            break

            }

    }

在这里要运用到一个选择语句，根据flag标记上一次所点的运算键，执行相应运算。

**2.1.7计算器中全清键的编写**

@IBAction func clean(sender: AnyObject) {

        wx.text=""

        temp=0

        flag=0

    }

**2.1.8计算器中负号的编写**

@IBAction func minussign(sender: AnyObject) {

        temp=(Double)(wx.text!)!

        temp=(-temp)

        wx.text="\(temp)"

    }

**2.1.9计算器中百分号的编写**

@IBAction func percent(sender: AnyObject) {

        var x=0.0

        if  flag==1||flag==2||flag==3||flag==4{

            switch  flag{

            case 1:

                x=temp\*(Double)(wx.text!)!/100

                temp=temp+x

                wx.text="\(temp)"

            case 2:

                x=temp\*(Double)(wx.text!)!/100

                temp=temp-x

                wx.text="\(temp)"

            case 3:

                x=temp\*(Double)(wx.text!)!/100

                temp=temp\*x

                wx.text="\(temp)"

            case 4:

                if(Double)(wx.text!)!==0

                {

                    wx.text="o不能为除数"

                }

                else{

                    x=temp\*(Double)(wx.text!)!/100

                    temp=temp/x

                    wx.text="\(temp)"                }

            default:

                break

            }

        }

        else{

            temp=(Double)(wx.text!)!

            temp=temp/100

            wx.text="\(temp)"}

    }

**2.1.10计算器中平方的编写**

@IBAction func square(sender: AnyObject) {

        temp=(Double)(wx.text!)!

        temp=temp\*temp

        wx.text="\(temp)"

    }

**2.1.11计算器中根号的编写**

@IBAction func root(sender: AnyObject) {

        temp=(Double)(wx.text!)!

        temp=sqrt(temp)

        wx.text="\(temp)"

    }

**2.1.12计算器中清除键的编写**

@IBAction func back(sender: AnyObject) {

        var s=""

        s = wx.text!

        s.removeAtIndex(s.endIndex.predecessor())

        wx.text=s

    }

**2.1.13计算器中倒数的编写**

@IBAction func reciprocal(sender: AnyObject) {

        temp=(Double)(wx.text!)!

        temp=1/temp

        wx.text="\(temp)"

    }

**2.1.14计算器中小数点的编写**

 @IBAction func decimal(sender: AnyObject) {

        if point==true{

        if( wx.text=="0")

        {

            wx.text=""

            wx.text=(wx.text)!+"."

        }

        else

        {

            wx.text=(wx.text)!+".";

        }

            point=false

        }

    }

第3章  调试分析

3.1调试（debugging）是指去掉程序中的错误（通常被称为bugs）的过程。一个错误可能非常简单，例如拼错一个单词或者漏掉一个分好；也可能比较复杂，例如使用一个指向并不存在地址的指针。

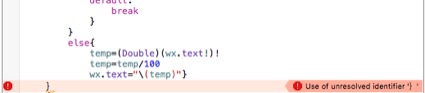
计算器中测试结果：



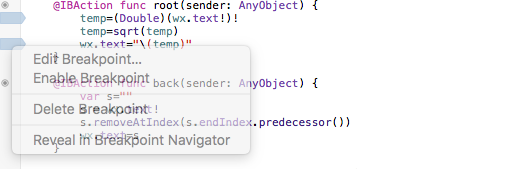
3.2 调试中出现的问题

在IOS移动开发设计当中，通常会遇到以下问题：

1.对于代码的准确性要求很高，即使一个中文格式的大括号，也会使程序无法运行



2.调试程序时要注意断点的存在会是程序调试失败



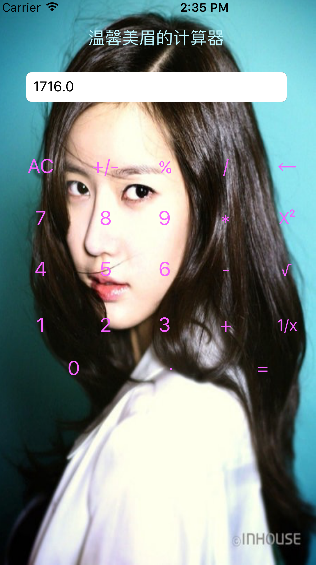
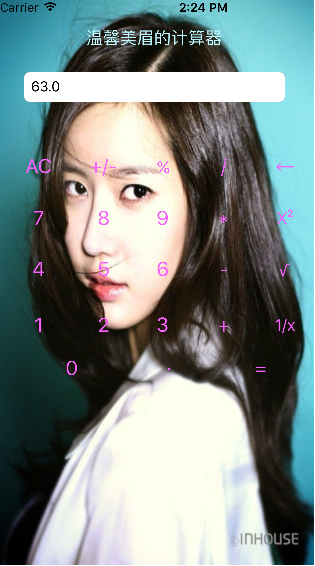
3.Android和IOS作为移动端操作系统和web端是完全不同的两个领域，所以要避免将web端的经验带入移动端应用程序

第4章  使用说明与执行结果

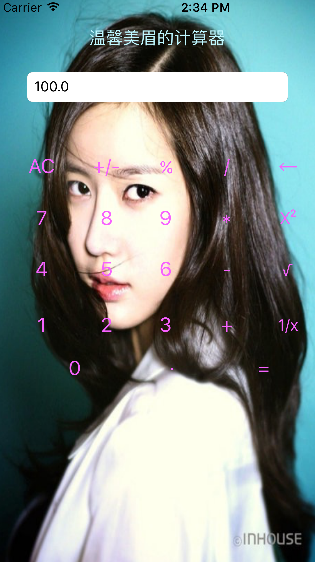
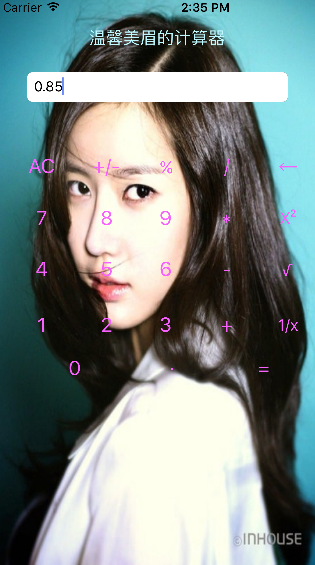
**一．运行操作及结果:**



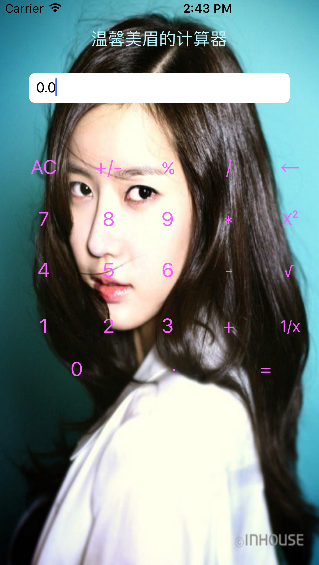
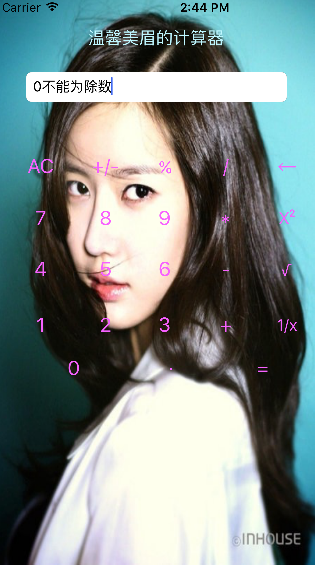
**执行7\*9： 执行809+907：**



执行10000开根号： 执行85% ：

执行0 / 30： 执行10 / 0 ：

**二．课程设计小结**

**在实验中发现——分析——总结的问题有：**

本次试验一开始让我想到的是曾经使用Visual Studio 2010进行编写计算器，但是因为涉及到一个新的编程语言Swift，所以编程语言及编译环境完全改变了。不过操作方式与有些函数同上次实验类似。在编译途中，我发现运用Swift语言要比C语言更加灵活，它基于C和Objective-C，而却没有C的一些兼容约束，是第一套具有与脚本语言同样的表现力和趣味性的系统编程语言。

开始编写的时候，思路混乱，对于知识点的记忆不够牢固，经常一个按钮绑了多个变量导致程序总是出错误，只能编译一点运行一下，从而编译速度特别慢。但在老师的指导和同学的交流下，编译过程越来越顺利，遇到一些小问题自己查阅资料就可以解决。

本次实验让我更加明白对于编程要有足够的耐心和细心。不需要害怕出错，只要反复实验，从中吸取经验，问题就会越来越少。

**参考文献：**

**[1] SwiftV课堂-中国最大的Swift视频学习站 http：//[www.swiftv.cn](http://www.swiftv.cn/" \t "https://set2.mail.qq.com/cgi-bin/_blank)**

**[2] 百度百科[http://baike.baidu.com/](http://baike.baidu.com/" \t "https://set2.mail.qq.com/cgi-bin/_blank)**