**import UIKit**

**import GoogleMobileAds**

**import SwiftyStoreKit**

**@UIApplicationMain**

**class AppDelegate: UIResponder, UIApplicationDelegate {**

**var window: UIWindow?**

**func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {**

**//初始化路由**

**RoutableManager.config()**

**//创建计算器历史数据库**

**CalHistoryDBManager.shareManger().openDB(DBName: "calHis")**

**GADMobileAds.sharedInstance().start(completionHandler: nil)**

**SwiftyStoreKit.completeTransactions(atomically: true) { purchases in**

**for purchase in purchases {**

**switch purchase.transaction.transactionState {**

**case .purchased, .restored:**

**if purchase.needsFinishTransaction {**

**// Deliver content from server, then:**

**SwiftyStoreKit.finishTransaction(purchase.transaction)**

**}**

**case .failed, .purchasing, .deferred:**

**break // do nothing**

**default:**

**break;**

**}**

**}**

**}**

**return true**

**}**

**}**

**import UIKit**

**class BaseViewController: UIViewController {**

**var titleLab:UILabel?**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**// Do any additional setup after loading the view.**

**}**

**func setNav() {**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: .white), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**let topBorder = CALayer()**

**let borderHeight: CGFloat = 0.33**

**topBorder.borderWidth = borderHeight**

**topBorder.borderColor = UIColor.colorWithHexColorString("000000", alpha: 0.1).cgColor**

**topBorder.frame = CGRect(x: 0, y: 43.67, width: CALScreenWidth, height: borderHeight)**

**self.navigationController?.navigationBar.layer.addSublayer(topBorder)**

**let titleView = UILabel()**

**titleView.font = UIFont.pingFangSC\_Semibold(size: 17)**

**titleView.text = "Basic calculator"**

**titleView.lineBreakMode = .byTruncatingMiddle**

**titleView.textAlignment = .center**

**titleView.frame = CGRect(x: 0, y: 0, width: 200, height: 30)**

**titleLab = titleView**

**self.navigationItem.titleView = titleLab**

**self.navigationItem.leftBarButtonItem = UIBarButtonItem(customView: UIBarButtonItemBackView())**

**}**

**}**

**import UIKit**

**class LJ\_BaseTableViewController: UITableViewController {**

**var titleLab:UILabel?**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**func setNav() {**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: .white), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**let topBorder = CALayer()**

**let borderHeight: CGFloat = 0.33**

**topBorder.borderWidth = borderHeight**

**topBorder.borderColor = UIColor.colorWithHexColorString("000000", alpha: 0.1).cgColor**

**topBorder.frame = CGRect(x: 0, y: 43.67, width: CALScreenWidth, height: borderHeight)**

**self.navigationController?.navigationBar.layer.addSublayer(topBorder)**

**let titleView = UILabel()**

**titleView.font = UIFont.pingFangSC\_Semibold(size: 17)**

**titleView.text = "Basic calculator"**

**titleView.lineBreakMode = .byTruncatingMiddle**

**titleView.textAlignment = .center**

**titleView.frame = CGRect(x: 0, y: 0, width: 200, height: 30)**

**titleLab = titleView**

**self.navigationItem.titleView = titleLab**

**self.navigationItem.leftBarButtonItem = UIBarButtonItem(customView: UIBarButtonItemBackView())**

**}**

**}**

**import UIKit**

**class CalHistoryDBManager: NSObject {**

**private static let manger:CalHistoryDBManager = CalHistoryDBManager()**

**class func shareManger()->CalHistoryDBManager{**

**return manger**

**}**

**var db:FMDatabase?**

**func openDB(DBName:String) {**

**let path = (NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, true).last ?? "") + "/\(DBName)"**

**//        print("数据库的路径=\(path)")**

**db = FMDatabase(path: path)**

**if !db!.open() {**

**print("打开数据库失败")**

**return**

**}**

**createTable(tableName: "CalHis")**

**createTable(tableName: "SenCalHis")**

**}**

**// MARK: 3、创建表**

**func createTable(tableName:String) {**

**// 1.编写SQL语句**

**// id: 主键  name和age是字段名**

**let sql = """**

**CREATE TABLE IF NOT EXISTS \(tableName)(id INTEGER PRIMARY KEY AUTOINCREMENT,**

**result TEXT,**

**formula TEXT,**

**time TEXT**

**);**

**"""**

**//            print(sql)**

**if db!.executeUpdate(sql, withArgumentsIn: []) {**

**print("创建表成功")**

**}else{**

**print("创建表失败")**

**}**

**}**

**func insertResult(tableName:String,result:String,formula:String) {**

**let sql = """**

**INSERT INTO \(tableName)(result,**

**formula,**

**time**

**)**

**VALUES(**

**'\(result)',**

**'\(formula)',**

**'\(0)')**

**"""**

**if db!.open() {**

**if db!.executeUpdate(sql, withArgumentsIn: []) {**

**print("插入成功")**

**}else{**

**print("插入失败")**

**}**

**}**

**db!.close()**

**}**

**func selectAllResult(tableName:String)->Array<Dictionary<String,String>>{**

**let sql = """**

**SELECT \* FROM "\(tableName)"**

**"""**

**var resArr:Array<Dictionary<String,String>> = []**

**if db!.open() {**

**do {**

**let res =  try db!.executeQuery(sql, values: [])**

**while res.next() {**

**let dic = ["result":res.string(forColumn: "result") ?? "0","formula":res.string(forColumn: "formula") ?? "0","time":res.string(forColumn: "time") ?? "0"]**

**resArr.append(dic)**

**}**

**} catch  {**

**}**

**}**

**db!.close()**

**return resArr.reversed()**

**}**

**func deleteAllResult(tableName:String) {**

**let sql = """**

**DELETE FROM \(tableName)**

**"""**

**print(sql)**

**if db!.open() {**

**if db!.executeUpdate(sql, withArgumentsIn: []) {**

**print("删除成功")**

**}else{**

**print("删除失败")**

**}**

**}**

**db!.close()**

**}**

**}**

**import UIKit**

**protocol NormalKeyBoardDelgate {**

**func keyClick(type:KeyType)**

**}**

**class NormalKeyBoard: UIView,NibLoadable {**

**var delegate:NormalKeyBoardDelgate?**

**@IBAction func keyBoardTap(\_ sender: KeyBoardBtn) {**

**delegate?.keyClick(type: sender.btnType ?? .digit(0))**

**}**

**}**

**import UIKit**

**class SystemKeyBoard: UIInputView,NibLoadable {**

**var delegate:NormalKeyBoardDelgate?**

**var dataType:DataType = .Binary{**

**didSet{**

**switch dataType {**

**case .Binary:**

**setBinaryBoard()**

**case .Octal:**

**setOctalBoard()**

**case .Decimal:**

**setDecimalBoard()**

**case .Hex:**

**setHexBoard()**

**}**

**}**

**}**

**@IBOutlet var binaryBtns: [KeyBoardBtn]!**

**@IBOutlet var octalBtns: [KeyBoardBtn]!**

**@IBOutlet var decimalBtns: [KeyBoardBtn]!**

**@IBOutlet var HexBtns: [KeyBoardBtn]!**

**@IBAction func keyBoardTap(\_ sender: KeyBoardBtn) {**

**delegate?.keyClick(type: sender.btnType ?? .digit(0))**

**}**

**func setBinaryBoard() {**

**octalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**decimalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**HexBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**binaryBtns.forEach({$0.isUserInteractionEnabled = true;$0.alpha = 1.0})**

**}**

**func setOctalBoard() {**

**binaryBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**decimalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**HexBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**octalBtns.forEach({$0.isUserInteractionEnabled = true;$0.alpha = 1.0})**

**}**

**func setDecimalBoard() {**

**octalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**binaryBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**HexBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**decimalBtns.forEach({$0.isUserInteractionEnabled = true;$0.alpha = 1.0})**

**}**

**func setHexBoard() {**

**octalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**decimalBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**binaryBtns.forEach({$0.isUserInteractionEnabled = false;$0.alpha = 0.4})**

**HexBtns.forEach({$0.isUserInteractionEnabled = true;$0.alpha = 1.0})**

**}**

**}**

**import UIKit**

**enum KeyType {**

**case digit(Int)**

**case negative**

**case clear**

**case delete**

**case dot**

**case hex(String)**

**init(string:String) {**

**switch string {**

**case "1","2","3","4","5","6","7","8","9","0":**

**self = .digit(Int(string)!)**

**case "-":**

**self = .negative**

**case "c":**

**self = .clear**

**case "d":**

**self = .delete**

**case "A","B","C","D","E","F":**

**self = .hex(string)**

**case ".":**

**self = .dot**

**default:**

**self = .dot**

**}**

**}**

**public var description: String {**

**switch self {**

**case .digit(let num):**

**return "\(num)"**

**case .dot:**

**return "."**

**case .clear:**

**return "c"**

**case .negative:**

**return "-"**

**case .hex(let str):**

**return str**

**case .delete:**

**return "d"**

**}**

**}**

**}**

**@IBDesignable class KeyBoardBtn: UIButton {**

**@IBInspectable @objc var calculatorType:String{**

**set{**

**self.btnType = KeyType(string: newValue)**

**}**

**get{**

**return self.btnType?.description ?? "0"**

**}**

**}**

**var btnType:KeyType?**

**}**

**import UIKit**

**import GoogleMobileAds**

**class MainViewController: UIViewController,UICollectionViewDelegateFlowLayout,UICollectionViewDataSource,StoryboardLoadable, GADInterstitialDelegate {**

**var modelArr:Array<CategoryModel> = []**

**@IBOutlet weak var collectionView: UICollectionView!**

**var flowLayout:UICollectionViewFlowLayout?**

**var interstitial: GADInterstitial!**

**@objc  class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let vc = MainViewController.loadStoryboard(identifier: "MainViewController")**

**vc.hidesBottomBarWhenPushed = true**

**return vc**

**}**

**override func viewWillAppear(\_ animated: Bool) {**

**Routable.sharedRouter()?.navigationController = self.navigationController**

**loadData()**

**}**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**interstitial = createAndLoadInterstitial()**

**setNav()**

**makeCollection()**

**NotificationCenter.default.addObserver(forName: NSNotification.Name("purchaseRes"), object: nil, queue: nil) { notifi in**

**if notifi.userInfo?["res"] as? Bool ?? false {**

**self.loadData()**

**}**

**}**

**// Do any additional setup after loading the view.**

**}**

**@objc func loadData() {**

**modelArr = []**

**categoryArr.forEach { item in**

**let model = CategoryModel(dic: item)**

**if PurchaseTool.sharedInstance.getCurVipStatus() {**

**model.status = .normal**

**}else{**

**switch model.index {**

**case 0:**

**model.status = .normal**

**case 1,2,3:**

**model.status = .ads**

**default:**

**model.status = .purchase**

**}**

**}**

**modelArr.append(model)**

**}**

**collectionView.reloadData()**

**}**

**func createAndLoadInterstitial() -> GADInterstitial {**

**let interstitial = GADInterstitial(adUnitID: "ca-app-pub-3940256099942544/4411468910")**

**interstitial.delegate = self**

**interstitial.load(GADRequest())**

**return interstitial**

**}**

**func interstitialDidDismissScreen(\_ ad: GADInterstitial) {**

**interstitial = createAndLoadInterstitial()**

**}**

**func setNav(){**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: .white), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**let topBorder = CALayer()**

**let borderHeight: CGFloat = 0.33**

**topBorder.borderWidth = borderHeight**

**topBorder.borderColor = UIColor.colorWithHexColorString("000000", alpha: 0.1).cgColor**

**topBorder.frame = CGRect(x: 0, y: 43.67, width: CALScreenWidth, height: borderHeight)**

**self.navigationController?.navigationBar.layer.addSublayer(topBorder)**

**let titleView = UILabel()**

**titleView.font = UIFont.pingFangSC\_Semibold(size: 17)**

**titleView.text = NSLocalizedString("Super calculator", comment: "")**

**titleView.lineBreakMode = .byTruncatingMiddle**

**self.navigationItem.titleView = titleView**

**let btn = UIBarButtonItem(title: NSLocalizedString("setting", comment: ""), style: .done, target: self, action: #selector(setting))**

**self.navigationItem.rightBarButtonItems = [btn]**

**self.navigationItem.rightBarButtonItem?.tintColor = UIColor.colorWithHexColorString("FF9734")**

**}**

**@objc func setting(){**

**let vc = LJ\_AboutTableViewController.loadStoryboard(name: "Main")**

**navigationController?.pushViewController(vc, animated: true)**

**}**

**func makeCollection() {**

**self.collectionView.delegate = self**

**self.collectionView.dataSource = self**

**}**

**func collectionView(\_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {**

**return modelArr.count**

**}**

**func collectionView(\_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {**

**let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cateCell", for: indexPath) as? MainCollectionViewCell**

**cell?.imageView.image = UIImage(named: modelArr[indexPath.row].cateImageName)**

**cell?.titleLab.text = NSLocalizedString(modelArr[indexPath.row].cateTitle, comment: "")**

**cell?.status = modelArr[indexPath.row].status**

**return cell!**

**}**

**func collectionView(\_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {**

**let model = modelArr[indexPath.row]**

**let ind = model.cateController**

**let title = NSLocalizedString(model.cateTitle, comment: "")**

**//传入的cateImageName就是json文件名**

**let fileName = model.cateImageName**

**if !PurchaseTool.sharedInstance.getCurVipStatus() {**

**self.loadData()**

**}**

**switch model.status {**

**case .normal:**

**if ind == "LJ\_UnitTableViewController" {**

**Routable.sharedRouter()?.open("\(ind)/\(title)/\(fileName)")**

**}else if ind == "LJ\_DataTableViewController"{**

**Routable.sharedRouter()?.open("\(ind)/\(title)/\(fileName)/1")**

**}else if ind == "LJ\_DateTableViewController"{**

**Routable.sharedRouter()?.open("\(ind)/\(title)")**

**}else{**

**Routable.sharedRouter()?.open(ind)**

**}**

**case .ads:**

**if ind == "LJ\_UnitTableViewController" {**

**Routable.sharedRouter()?.open("\(ind)/\(title)/\(fileName)")**

**}else if ind == "LJ\_DataTableViewController"{**

**Routable.sharedRouter()?.open("\(ind)/\(title)/\(fileName)/1")**

**}else if ind == "LJ\_DateTableViewController"{**

**Routable.sharedRouter()?.open("\(ind)/\(title)")**

**}else{**

**Routable.sharedRouter()?.open(ind)**

**}**

**if interstitial.isReady {**

**interstitial.present(fromRootViewController: self)**

**} else {**

**print("Ad wasn't ready")**

**}**

**case .purchase:**

**let vc = LJ\_PurchaseTableViewController.loadStoryboard(name: "Main")**

**self.present(vc, animated: true, completion: nil)**

**print("show purchase")**

**}**

**}**

**func interstitialWillDismissScreen(\_ ad: GADInterstitial) {**

**}**

**}**

**import UIKit**

**class MainCollectionViewCell: UICollectionViewCell {**

**@IBOutlet weak var imageView: UIImageView!**

**@IBOutlet weak var titleLab: UILabel!**

**@IBOutlet weak var statusImage: UIImageView!**

**var status:ItemStatus = .purchase{**

**didSet{**

**statusImage.isHidden = false**

**switch status {**

**case .normal:**

**statusImage.isHidden = true**

**case .ads:**

**statusImage.image = UIImage(named: "status\_ads")**

**case .purchase:**

**statusImage.image = UIImage(named: "status\_purchase")**

**}**

**}**

**}**

**}**

**import UIKit**

**enum ItemStatus {**

**case normal**

**case ads**

**case purchase**

**}**

**class CategoryModel: NSObject {**

**var cateImageName = ""**

**var cateTitle = ""**

**var cateController = ""**

**var status:ItemStatus = .purchase**

**var index = 0**

**init(dic:Dictionary<String,Any>) {**

**cateImageName = dic["cateImageName"] as? String ?? ""**

**cateTitle = dic["cateTitle"] as? String ?? ""**

**cateController = dic["cateController"] as? String ?? ""**

**index = dic["index"] as? Int ?? 0**

**}**

**}**

**import UIKit**

**class LJ\_AboutTableViewController: UITableViewController,StoryboardLoadable {**

**@IBOutlet weak var versionLab: UILabel!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**versionLab.text = "\(Bundle.main.infoDictionary!["CFBundleShortVersionString"] as! String)"**

**self.title = NSLocalizedString("setting", comment: "")**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor.colorWithHexColorString("F9F9F9")**

**return view**

**}**

**override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 10**

**}**

**override func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {**

**if indexPath.section == 0 {**

**if indexPath.row == 0 {**

**let vc = LJ\_WebViewController()**

**vc.urlStr = aboutUrl**

**vc.titleStr = "About";**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}else if indexPath.row == 1{**

**let image = UIImage(named: "1.png")**

**let url = URL(string: "https://apps.apple.com/cn/app/id\(appid)")!**

**///  applicationActivities 可以没有元素，系统会自动选择合适的平台**

**let activityController = UIActivityViewController(activityItems: ["Ghost Detection Machine", image!,url], applicationActivities: [])**

**activityController.completionWithItemsHandler = {**

**(type, flag, array, error) -> Swift.Void in**

**print(type ?? "")**

**}**

**present(activityController, animated: true) { }**

**}else{**

**let appUrl = "https://itunes.apple.com/cn/app/id\(appid)?action=write-review"**

**if UIApplication.shared.canOpenURL(URL(string: appUrl)!){**

**UIApplication.shared.open(URL(string: appUrl)!, options: [:], completionHandler: nil)**

**}            }**

**}else{**

**if indexPath.row == 0 {**

**let vc = LJ\_WebViewController()**

**vc.urlStr = privacyUrl**

**vc.titleStr = "Privacy Policy";**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}else if indexPath.row == 1{**

**let vc = LJ\_WebViewController()**

**vc.urlStr = termsUrl**

**vc.titleStr = "Terms of Use";**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**}**

**}**

**}**

**import UIKit**

**import WebKit**

**import SnapKit**

**class LJ\_WebViewController: UIViewController,WKNavigationDelegate  {**

**lazy var progressView:UIProgressView = {**

**let progressView = UIProgressView()**

**progressView.trackTintColor = UIColor.white**

**progressView.progressTintColor = UIColor(red: 66/255.0, green: 192/255.0, blue: 46/255.0, alpha: 1)**

**return progressView**

**}()**

**var webView:WKWebView?**

**var urlStr:String = ""**

**var titleStr:String = ""**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**initToolbar()**

**initWebView()**

**loadWebView()**

**// Do any additional setup after loading the view.**

**}**

**func initToolbar(){**

**let toolBar = UIView(frame: CGRect(x: 0, y: 0, width: CALScreenWidth, height: CAL\_Navi\_Height))**

**toolBar.backgroundColor = UIColor(red: 255/255.0, green: 255/255.0, blue: 255/255.0, alpha: 1)**

**let lineView = UIView(frame: CGRect(x: 0, y: CAL\_Navi\_Height - 1, width: CALScreenWidth, height: 1))**

**let goBackButton = UIButton(type: .custom)**

**goBackButton.setImage(UIImage(named: "navback"), for: .normal)**

**goBackButton.addTarget(self, action: #selector(backClicked), for: .touchUpInside)**

**let titleLab = UILabel()**

**if self.titleStr != "" {**

**titleLab.text = self.titleStr**

**}**

**titleLab.textColor = UIColor(red: 0/255.0, green: 0/255.0, blue: 0/255.0, alpha: 1)**

**titleLab.font = UIFont.boldSystemFont(ofSize: 17)**

**toolBar.addSubview(goBackButton)**

**toolBar.addSubview(titleLab)**

**toolBar.addSubview(lineView)**

**self.view.addSubview(toolBar)**

**goBackButton.snp.makeConstraints { make in**

**make.left.equalTo(toolBar.snp\_leftMargin).offset(6)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-4)**

**make.width.height.equalTo(30)**

**}**

**titleLab.snp.makeConstraints { make in**

**make.centerX.equalTo(toolBar.snp\_centerXWithinMargins)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-7)**

**make.height.equalTo(22)**

**}**

**}**

**func initWebView() {**

**self.progressView.frame = CGRect(x: 0, y: CAL\_Navi\_Height, width: CALScreenWidth, height: 3)**

**self.view.addSubview(progressView)**

**self.view.backgroundColor = UIColor.white**

**self.webView = WKWebView(frame: CGRect(x: 16, y: CAL\_Navi\_Height + 12, width: CALScreenWidth - 32, height: CALScreenHeight - CALScreenWidth - 12))**

**self.webView?.backgroundColor = UIColor.clear**

**self.webView?.navigationDelegate = self**

**self.webView?.addObserver(self, forKeyPath: "estimatedProgress", options: .new, context: nil)**

**self.view.addSubview(self.webView!);**

**}**

**func loadWebView() {**

**if self.urlStr != "" {**

**let request = URLRequest(url: URL(string: self.urlStr)!)**

**self.webView?.load(request)**

**}**

**}**

**@objc func backClicked(){**

**self.dismiss(animated: true, completion: nil)**

**}**

**func webView(\_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction, decisionHandler: @escaping (WKNavigationActionPolicy) -> Void) {**

**progressView.setProgress(0.0, animated: false)**

**if navigationAction.targetFrame == nil {**

**webView .load(navigationAction.request)**

**}**

**decisionHandler(WKNavigationActionPolicy.allow)**

**}**

**func webView(\_ webView: WKWebView, didFinish navigation: WKNavigation!) {**

**progressView.setProgress(0.0, animated: false)**

**}**

**override func observeValue(forKeyPath keyPath: String?, of object: Any?, change: [NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {**

**if (keyPath == "estimatedProgress") {**

**progressView.isHidden = self.webView?.estimatedProgress == 1**

**progressView.setProgress(Float(self.webView!.estimatedProgress), animated: true)**

**}**

**}**

**}**

**import UIKit**

**class LJ\_PurchaseTableViewController: UITableViewController,StoryboardLoadable {**

**@IBOutlet weak var textView: UITextView!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setTextView()**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 0.001**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = .white**

**return view**

**}**

**@IBAction func purchaseTap(\_ sender: UITapGestureRecognizer) {**

**PurchaseTool.sharedInstance.payWithProduct(productID: "com.product.purchase.year") { res in**

**self.dismiss(animated: true, completion: nil)**

**}**

**}**

**@IBAction func closeClick(\_ sender: UIButton) {**

**self.dismiss(animated: true, completion: nil)**

**}**

**@IBAction func restoreClick(\_ sender: UIButton) {**

**}**

**func setTextView() {**

**let str1 = NSLocalizedString("pircha\_1", comment: "")**

**let str2 = NSLocalizedString("terms\_of\_use", comment: "")**

**let str3 = NSLocalizedString("purcha\_2", comment: "")**

**let str4 = NSLocalizedString("privacy\_policy", comment: "")**

**let str5 = NSLocalizedString("purcha\_3", comment: "")**

**let str:NSString = (str1 + str2 + str3 + str4 + str5) as NSString**

**let range1 = str.range(of: str2)**

**let range2 = str.range(of: str4)**

**let parStyle = NSMutableParagraphStyle()**

**parStyle.lineSpacing = 0.5**

**let att = NSMutableAttributedString(string: str as String, attributes: [NSAttributedString.Key.font:UIFont.pingFangSC\_Regular(size: 10),NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("000000", alpha: 0.7),NSAttributedString.Key.paragraphStyle:parStyle])**

**let valueStr1 = "TermsofService://\(str2)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**let valueStr2 = "PrivacyPolicy://\(str4)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**att.addAttributes([NSAttributedString.Key.link : valueStr1 as Any,], range: range1)**

**att.addAttributes([NSAttributedString.Key.link : valueStr2 as Any], range: range2)**

**textView.linkTextAttributes = [NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("FF7F24")]**

**textView.attributedText = att**

**}**

**func textView(\_ textView: UITextView, shouldInteractWith URL: URL, in characterRange: NSRange) -> Bool {**

**if URL.scheme == "TermsofService" {**

**self.clickTerms()**

**return false**

**}else if URL.scheme == "PrivacyPolicy"{**

**self.clickPrivacy()**

**return false**

**}**

**return true**

**}**

**func clickTerms() {**

**print("TermsofService")**

**let vc = LJ\_WebViewController()**

**vc.urlStr = termsUrl**

**vc.titleStr = NSLocalizedString("terms\_of\_use", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**func clickPrivacy() {**

**print("PrivacyPolicy")**

**let vc = LJ\_WebViewController()**

**vc.urlStr = privacyUrl**

**vc.titleStr = NSLocalizedString("privacy\_policy", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**}**

**import UIKit**

**class SenCalViewController: BaseViewController,StoryboardLoadable {**

**@objc  class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let vc = SenCalViewController.loadStoryboard(identifier: "ScientificCalculat")**

**vc.hidesBottomBarWhenPushed = true**

**return vc**

**}**

**@IBOutlet weak var calTF: UITextField!**

**@IBOutlet weak var historyView: UIView!**

**@IBOutlet weak var historyLab: UILabel!**

**@IBOutlet weak var calResultLab: UILabel!**

**@IBOutlet weak var degBtn: CalculatorBtn!**

**var calculatorBrain:CalculatorBrain?**

**var randNumArr:Array<Double> = []**

**var inputStr:String = ""{**

**didSet{**

**calTF.attributedText = formatInputStr(string: inputStr)**

**}**

**}**

**var result:String = ""**

**var isDeg = false{**

**didSet{**

**if !isDeg {**

**degBtn.setBackgroundImage(UIImage(named: "btn／deg"), for: .normal)**

**}else{**

**degBtn.setBackgroundImage(UIImage(named: "btn／rand"), for: .normal)**

**}**

**}**

**}**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setNav()**

**self.titleLab?.text = NSLocalizedString("Scientific calculator", comment: "")**

**calculatorBrain = CalculatorBrain()**

**calResultLab.adjustsFontSizeToFitWidth = true**

**self.calTF.delegate = self**

**calTF.inputView = UIView()**

**// Do any additional setup after loading the view.**

**}**

**@IBAction func btnClick(\_ sender: CalculatorBtn) {**

**if self.calResultLab.text == "Error!" {**

**}**

**guard let type = sender.calculatorType else {**

**return**

**}**

**//切换角度弧度**

**if type == "w" {**

**isDeg = !isDeg**

**return**

**}**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**if !inputStr.formulaCanAppend(Character(type.description)){**

**print("不可拼接")**

**return**

**}**

**switch CalculatorButton(str: sender.calculatorType ?? "") {**

**case .command(.clear):**

**calculatorBrain?.inputMut.removeAll()**

**self.calResultLab.text = ""**

**inputStr = ""**

**randNumArr = []**

**break**

**case .command(.flip):///正负号处理**

**calculatorBrain?.inputMut.append("p")**

**break**

**case .sc(.sin),.sc(.cos),.sc(.tan),.sc(.cot),.sc(.sinh),.sc(.cosh),.sc(.tanh),.sc(.lg),.sc(.ln):**

**calculatorBrain?.inputMut.append(type)**

**if (calculatorBrain?.calculationCheck(str: "(") ?? false){**

**calculatorBrain?.inputMut.append("(")**

**}**

**break**

**case .sc(.random):**

**calculatorBrain?.inputMut.append(type)**

**randNumArr.append(Double(arc4random())/10000000000.0)**

**calResultLab.text = "\(randNumArr.last ?? 0.0)".formateToMoneyFount()**

**//        case .sc(.square):**

**//            calculatorBrain?.inputMut.append(type)**

**case .sc(\_):**

**calculatorBrain?.inputMut.append(type)**

**case .op(.equal):**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**if calculatorBrain?.inputMut.count == 0 {**

**return**

**}**

**if !inputStr.isRightFormula(){**

**self.calResultLab.text = "Error!"**

**return**

**}**

**if inputStr.contains("y") {**

**for i in 0 ..< randNumArr.count {**

**let range = inputStr.range(of: "y")**

**inputStr = inputStr.replacingOccurrences(of: "y", with: "\(randNumArr[i])", options: String.CompareOptions(rawValue: 0), range: range)**

**}**

**}**

**let temp = calculatorBrain?.calculation(str: inputStr, isRadina: !isDeg)**

**result = temp?.removeLastZero() ?? "Error"**

**self.calResultLab.text = result.formateToMoneyFount()**

**self.historyLab.text = result.formateToMoneyFount()**

**CalHistoryDBManager.shareManger().insertResult(tableName: "SenCalHis", result: result.formateToMoneyFount(), formula: formatInputStr(string: inputStr).mutableString as String)**

**break**

**case.op(let synmbol):**

**calculatorBrain?.inputMut.append(synmbol.rawValue)**

**break;**

**case.digit(let i):**

**var str = ""**

**for string in calculatorBrain!.inputMut.reversed() {**

**let bool = (Int(string) ?? -1 >= 0 && Int(string) ?? -1 <= 9) || string == "."**

**if bool {**

**str.append(string)**

**}else{**

**break**

**}**

**}**

**if !str.contains(".") && str.length > 14 {**

**return**

**}**

**calculatorBrain?.inputMut.append("\(i)")**

**break**

**case .command(.percent):**

**calculatorBrain?.inputMut.append("%")**

**case .dot:**

**if calculatorBrain?.flag == 0 {**

**calculatorBrain?.inputMut.append(".")**

**calculatorBrain?.flag = 1**

**}else{**

**calculatorBrain?.inputMut.append(".")**

**}**

**default:**

**break**

**}**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**}**

**@IBAction func deleteClick(\_ sender: UIButton) {**

**if calculatorBrain?.inputMut.last == "(" {**

**calculatorBrain?.inputMut.removeLast()**

**if (calculatorBrain?.inputMut.count ?? 0) > 0 {**

**let c = calculatorBrain?.inputMut.last**

**if c?.regexWith(pattern: "[fghjklmq㏑]") ?? false {**

**calculatorBrain?.inputMut.removeLast()**

**}**

**}**

**}else{**

**if (calculatorBrain?.inputMut.count ?? 0) > 0 {**

**calculatorBrain?.inputMut.removeLast()**

**}**

**}**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**}**

**func formatInputStr(string:String) -> NSMutableAttributedString {**

**var forStr = string**

**forStr = forStr.replacingOccurrences(of: "p", with: "-")**

**forStr = forStr.replacingOccurrences(of: "x", with: "EE")**

**forStr = forStr.replacingOccurrences(of: "v", with: "^(-1)")**

**forStr = forStr.replacingOccurrences(of: "f", with: "sin")**

**forStr = forStr.replacingOccurrences(of: "g", with: "cos")**

**forStr = forStr.replacingOccurrences(of: "j", with: "cot")**

**forStr = forStr.replacingOccurrences(of: "y", with: "Rand")**

**forStr = forStr.replacingOccurrences(of: "h", with: "tan")**

**forStr = forStr.replacingOccurrences(of: "k", with: "sinh")**

**forStr = forStr.replacingOccurrences(of: "l", with: "cosh")**

**forStr = forStr.replacingOccurrences(of: "q", with: "lg")**

**forStr = forStr.replacingOccurrences(of: "m", with: "tanh")**

**forStr = forStr.formateToMoneyFount()**

**let ranges = forStr.regexWithSymbol()**

**let att = NSMutableAttributedString(string: forStr)**

**att.addAttributes([NSAttributedString.Key.foregroundColor : UIColor.colorWithHexColorString("505050"),NSAttributedString.Key.font:UIFont.systemFont(ofSize: 18)], range: NSRange(location: 0, length: forStr.length))**

**for range in ranges {**

**att.addAttributes([NSAttributedString.Key.foregroundColor : UIColor.colorWithHexColorString("FF9734")], range: range)**

**if range.location > 0 {**

**att.addAttributes([NSAttributedString.Key.kern : NSNumber(value: 5.0)], range: NSRange(location: range.location - 1, length: range.length + 1))**

**}**

**}**

**return att**

**}**

**@IBAction func historyClick(\_ sender: UITapGestureRecognizer) {**

**let vc = CalculatorHistoryViewController.loadStoryboard(name: "Main")**

**vc.tableName = "SenCalHis"**

**self.navigationController?.present(vc, animated: true, completion: nil)**

**}**

**/\***

**// MARK: - Navigation**

**// In a storyboard-based application, you will often want to do a little preparation before navigation**

**override func prepare(for segue: UIStoryboardSegue, sender: Any?) {**

**// Get the new view controller using segue.destination.**

**// Pass the selected object to the new view controller.**

**}**

**\*/**

**}**

**extension SenCalViewController:UITextFieldDelegate{**

**func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {**

**print(string,range)**

**let selectRange = textField.selectedTextRange**

**if selectRange?.isEmpty ?? false {**

**}else{**

**if string.count == 0{**

**self.calculatorBrain?.inputMut.removeSubrange(Range(range)!)**

**}else{**

**var i = 0**

**for c in string {**

**self.calculatorBrain?.inputMut.insert("\(c)", at: range.location + i)**

**i += 1**

**}**

**}**

**}**

**return true**

**}**

**func textFieldDidEndEditing(\_ textField: UITextField, reason: UITextField.DidEndEditingReason) {**

**}**

**}**

**import UIKit**

**import SwiftyJSON**

**class LJ\_UnitManger: NSObject {**

**class func getModelWith(fileName:String,baseValue:String)->([LJ\_BaseModel]){**

**let path = Bundle.main.path(forResource: fileName, ofType: "json")**

**if let jsonPath = path {**

**let data = NSData(contentsOfFile: jsonPath)**

**let countryJSON = try! JSON(data: data! as Data)**

**let modelArray = countryJSON.arrayValue**

**return modelArray.compactMap { json -> LJ\_BaseModel in**

**return LJ\_BaseModel(json: json, baseValue: baseValue)**

**}**

**}else{**

**return []**

**}**

**}**

**class func getModel(fileName:String,baseValue:String)->([LJ\_BaseModel]){**

**let path = Bundle.main.path(forResource: fileName, ofType: "json")**

**if let jsonPath = path {**

**let data = NSData(contentsOfFile: jsonPath)**

**let countryJSON = try! JSON(data: data! as Data)**

**let modelArray = countryJSON.arrayValue**

**return modelArray.compactMap { json -> LJ\_BaseModel in**

**return LJ\_BaseModel(json: json, binaryValue: baseValue)**

**}**

**}else{**

**return []**

**}**

**}**

**}**

**import UIKit**

**import SwiftyJSON**

**enum Op:String {**

**case plus = "+"**

**case minus = "-"**

**case multiiply = "\*"**

**case divide = "/"**

**}**

**enum DataType:String {**

**case Binary = "Binary"**

**case Octal = "Octal"**

**case Decimal = "Decimal"**

**case Hex = "Hex"**

**}**

**class LJ\_BaseModel: NSObject {**

**var index = 0**

**//单位**

**var unit = ""**

**//单位简写**

**var abbUnit = ""**

**//**

**var coefficient = 0.0**

**//计算公式**

**var formula = ""**

**//计算结果值**

**var realValue:String = "1.0"**

**//计算base公式**

**var formulaToBase = ""**

**//当前值对应的base值**

**var toBaseValue:String = "0.0"**

**var base\_Value = ""**

**var type:DataType = .Decimal**

**init(json:JSON,baseValue:String) {**

**index = json["index"].intValue**

**unit = json["nuit"].stringValue**

**abbUnit = json["abbUnit"].stringValue**

**coefficient = json["coefficient"].doubleValue**

**formula = json["formula"].stringValue.replacingOccurrences(of: "r", with: "\(baseValue)")**

**formulaToBase = json["formulaToBase"].stringValue**

**realValue = CalculatorBrain().calculation(str: formula, isRadina: true)**

**toBaseValue = CalculatorBrain().calculation(str: formulaToBase, isRadina: true)**

**base\_Value = json["baseValue"].stringValue**

**}**

**init(json:JSON,binaryValue:String) {**

**index = json["index"].intValue**

**unit = json["nuit"].stringValue**

**abbUnit = json["abbUnit"].stringValue**

**coefficient = json["coefficient"].doubleValue**

**type = DataType(rawValue: json["nuit"].stringValue) ?? .Decimal**

**switch type {**

**case .Binary:**

**realValue = binaryValue.decTobin()**

**toBaseValue = binaryValue.BinaryToDecimal()**

**case .Octal:**

**realValue = binaryValue.decToOct()**

**toBaseValue = binaryValue.OctalToDecimal()**

**case .Decimal:**

**realValue = binaryValue**

**toBaseValue = binaryValue**

**case .Hex:**

**realValue = binaryValue.decTohex()**

**toBaseValue = binaryValue.HexToDecimal()**

**}**

**}**

**}**

**extension String{**

**//    func BinaryToDecimal() -> String {**

**//        var sum:Int = 0**

**//        for c in self {**

**//            if let number = Int(String(c))**

**//            {**

**//                sum = sum \* 2 + number**

**//            }**

**//        }**

**//        return "\(sum)"**

**//    }**

**func BinaryToDecimal() -> String {**

**let str = self.removeLastZero()**

**if str == "" {**

**return "0"**

**}**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i]**

**let res = CalculatorBrain().calculation(str: "\(n)×(2^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**return "\(num)"**

**}else{**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i]**

**let res = CalculatorBrain().calculation(str: "\(n)×(2^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**for i in 0 ..< resArr[1].count {**

**let n = String(resArr[1])[i ..< i+1]**

**let res = CalculatorBrain().calculation(str: "\(n)×(2^p\(i+1))")**

**num1 = num1 + (Double(res) ?? 0.0)**

**}**

**}**

**return CalculatorBrain().calculation(str: "\(num)+\(num1)")**

**}**

**//    func decTobin() -> String {**

**//        var num = Int(self) ?? 0**

**//        var str = ""**

**//        if num == 0 {**

**//            return "0"**

**//        }**

**//        while num > 0 {**

**//            str = "\(num % 2)" + str**

**//            num /= 2**

**//        }**

**//        return str**

**//    }**

**func decTobin() -> String {**

**let str = self.removeLastZero()**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**str = "\(num % 2)" + str**

**num /= 2**

**}**

**return "\(str)"**

**}else{**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**str = "\(num % 2)" + str**

**num /= 2**

**}**

**let deStr = "0." + String(resArr[1])**

**var dic = ["deci":[],"num":deStr] as [String : Any]**

**while Double(dic["num"] as! String)! - Double(Int(Double(dic["num"] as! String)!)) > 0 {**

**dic = decimalToBin(dic: dic, rax: "2")**

**if (dic["deci"] as! Array<String>).count > 20{**

**break**

**}**

**}**

**print(dic)**

**let resDeStr = "0." + (dic["deci"] as! Array<String>).joined()**

**return NSString.floatOne("\(str)", calculationType: .forAdd, floatTwo: resDeStr)**

**}**

**}**

**func decimalToBin(dic:Dictionary<String, Any>,rax:String) -> Dictionary<String, Any> {**

**var deciArr = dic["deci"] as! Array<String>**

**let dec = dic["num"] as? String ?? "0"**

**var num = CalculatorBrain().calculation(str: "\(dec)×\(rax)")**

**let intValue = Int(Double(num) ?? 0.0)**

**var char:Character = Character(Unicode.Scalar(48))**

**if intValue > 9 {**

**char = Character(Unicode.Scalar(intValue + 65 - 10) ?? "0")**

**}else{**

**char = Character(Unicode.Scalar(intValue + 48) ?? "0")**

**}**

**deciArr.append("\(char)")**

**num = NSString.floatOne(num, calculationType: .forSubtract, floatTwo: "\(intValue)")**

**return ["deci":deciArr,"num":"\(num)"] as [String : Any]**

**}**

**func OctalToDecimal() -> String {**

**let str = self.removeLastZero()**

**if str == "" {**

**return "0"**

**}**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i]**

**let res = CalculatorBrain().calculation(str: "\(n)×(8^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**return "\(num)"**

**}else{**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i]**

**let res = CalculatorBrain().calculation(str: "\(n)×(8^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**for i in 0 ..< resArr[1].count {**

**let n = String(resArr[1])[i ..< i+1]**

**let res = CalculatorBrain().calculation(str: "\(n)×(8^p\(i+1))")**

**num1 = num1 + (Double(res) ?? 0.0)**

**}**

**}**

**return CalculatorBrain().calculation(str: "\(num)+\(num1)")**

**}**

**func decToOct() -> String {**

**let str = self.removeLastZero()**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**str = "\(num % 8)" + str**

**num /= 8**

**}**

**return "\(str)"**

**}else{**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**str = "\(num % 8)" + str**

**num /= 8**

**}**

**let deStr = "0." + String(resArr[1])**

**var dic = ["deci":[],"num":deStr] as [String : Any]**

**while Double(dic["num"] as! String)! - Double(Int(Double(dic["num"] as! String)!)) > 0 {**

**dic = decimalToBin(dic: dic, rax: "8")**

**if (dic["deci"] as! Array<String>).count > 20{**

**break**

**}**

**}**

**print(dic)**

**let resDeStr = "0." + (dic["deci"] as! Array<String>).joined()**

**return NSString.floatOne("\(str)", calculationType: .forAdd, floatTwo: resDeStr)**

**}**

**}**

**func HexToDecimal() -> String {**

**let str = self.uppercased().removeLastZero()**

**if str == "" {**

**return "0"**

**}**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i].asciiValue**

**var number = Int(n ?? 0)**

**if number >= 65 {**

**number = number - 55**

**}else{**

**number = number - 48**

**}**

**let res = CalculatorBrain().calculation(str: "\(number)×(16^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**return "\(num)"**

**}else{**

**for i in 0 ..< resArr[0].count {**

**let n = resArr[0].reversed()[i].asciiValue**

**var number = Int(n ?? 0)**

**if number >= 65 {**

**number = number - 55**

**}else{**

**number = number - 48**

**}**

**let res = CalculatorBrain().calculation(str: "\(number)×(16^\(i))")**

**num = num + (Int(res) ?? 0)**

**}**

**for i in 0 ..< resArr[1].count {**

**let n = String(resArr[1])[i ..< i+1].uppercased().first?.asciiValue**

**var number = Int(n ?? 0)**

**if number >= 65 {**

**number = number - 55**

**}else{**

**number = number - 48**

**}**

**let res = CalculatorBrain().calculation(str: "\(number)×(16^p\(i+1))")**

**num1 = num1 + (Double(res) ?? 0.0)**

**}**

**}**

**return CalculatorBrain().calculation(str: "\(num)+\(num1)")**

**//        for i in str.utf8 {**

**//            //0-9：从48开始**

**//            sum = sum \* 16 + Int(i) - 48**

**//            //A-Z：从65开始**

**//            if i >= 65 {**

**//                sum -= 7**

**//            }**

**//        }**

**//        return "\(sum)"**

**}**

**func decTohex() -> String {**

**let str = self.removeLastZero()**

**let resArr = str.split(separator: ".")**

**var num = 0**

**var num1 = 0.0**

**if resArr.count == 1 {**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**var char:Character = Character(Unicode.Scalar(48))**

**if num % 16 > 9 {**

**char = Character(Unicode.Scalar(num % 16 + 65 - 10) ?? "0")**

**}else{**

**char = Character(Unicode.Scalar(num % 16 + 48) ?? "0")**

**}**

**str = "\(char)" + str**

**num /= 16**

**}**

**return "\(str)"**

**}else{**

**num = Int(Double(self) ?? 0.0)**

**var str = ""**

**if num == 0 {**

**return "0"**

**}**

**while num > 0 {**

**var char:Character = Character(Unicode.Scalar(48))**

**if num % 16 > 9 {**

**char = Character(Unicode.Scalar(num % 16 + 65 - 10) ?? "0")**

**}else{**

**char = Character(Unicode.Scalar(num % 16 + 48) ?? "0")**

**}**

**str = "\(char)" + str**

**num /= 16**

**}**

**let deStr = "0." + String(resArr[1])**

**var dic = ["deci":[],"num":deStr] as [String : Any]**

**while Double(dic["num"] as! String)! - Double(Int(Double(dic["num"] as! String)!)) > 0 {**

**dic = decimalToBin(dic: dic, rax: "16")**

**if (dic["deci"] as! Array<String>).count > 20{**

**break**

**}**

**}**

**print(dic)**

**return str + "." + (dic["deci"] as! Array<String>).joined()**

**}**

**}**

**}**

**import UIKit**

**class LJ\_UnitTableViewController: LJ\_BaseTableViewController,UIGestureRecognizerDelegate{**

**@objc class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let instance = LJ\_UnitTableViewController.init()**

**if let title = params?["title"] as? String{**

**instance.title = title**

**}**

**if let title = params?["fileName"] as? String{**

**instance.fileName = title**

**}**

**return instance**

**}**

**var dataArray:Array<LJ\_BaseModel> = []**

**var fileName:String = ""**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setNav()**

**titleLab?.text = NSLocalizedString(self.title ?? "", comment: "")**

**self.view.backgroundColor = UIColor.white**

**self.tableView.register(UINib(nibName: "BaseTableViewCell", bundle: nil), forCellReuseIdentifier: "baseCell")**

**dataArray = LJ\_UnitManger.getModelWith(fileName: fileName, baseValue: "1.0")**

**self.tableView.reloadData()**

**let tap = UITapGestureRecognizer(target: self, action: #selector(tableTap))**

**tap.delegate = self**

**self.tableView.addGestureRecognizer(tap)**

**}**

**override func viewWillDisappear(\_ animated: Bool) {**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "removeKeyboard")))**

**}**

**// MARK: - Table view data source**

**override func numberOfSections(in tableView: UITableView) -> Int {**

**// #warning Incomplete implementation, return the number of sections**

**return 1**

**}**

**override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**// #warning Incomplete implementation, return the number of rows**

**return dataArray.count**

**}**

**override func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {**

**return 56**

**}**

**override func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {**

**if let cell = tableView.dequeueReusableCell(withIdentifier: "baseCell", for: indexPath) as? BaseTableViewCell{**

**cell.delegate = self**

**cell.model = dataArray[indexPath.row]**

**return cell**

**}else{**

**return UITableViewCell()**

**}**

**}**

**override func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {**

**return UIView()**

**}**

**@objc func tableTap(){**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "removeKeyboard")))**

**}**

**}**

**extension LJ\_UnitTableViewController:BaseTableViewCellDelegete{**

**func moveView(y: CGFloat) {**

**UIView.animate(withDuration: 0.3) {**

**self.view.frame.origin.y = self.view.frame.origin.y - y**

**}**

**}**

**func changeValue(value: String, model: LJ\_BaseModel) {**

**let baseValue = CalculatorBrain().calculation(str: model.formulaToBase.replacingOccurrences(of: "r", with: "\(value)"))**

**dataArray = LJ\_UnitManger.getModelWith(fileName: fileName, baseValue: baseValue)**

**dataArray[model.index].base\_Value = "\(value)"**

**self.tableView.reloadData()**

**}**

**}**

**import UIKit**

**class LJ\_DataTableViewController: LJ\_BaseTableViewController,UIGestureRecognizerDelegate {**

**@objc class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let instance = LJ\_DataTableViewController.init()**

**if let title = params?["title"] as? String{**

**instance.title = title**

**}**

**if let isDataView = params?["isDataView"] as? String{**

**if isDataView == "1" {**

**instance.isDataView = true**

**}else{**

**instance.isDataView = false**

**}**

**}**

**return instance**

**}**

**var dataArray:Array<LJ\_BaseModel> = []**

**var fileName:String = ""**

**var isDataView = true{**

**didSet{**

**dataArray = []**

**if isDataView {**

**dataArray = LJ\_UnitManger.getModelWith(fileName: "Data", baseValue: "1.0")**

**}else{**

**dataArray = LJ\_UnitManger.getModel(fileName: "FeelTheBase", baseValue: "0")**

**}**

**tableView.reloadData()**

**}**

**}**

**lazy var headView: UIView = {**

**let view = UIView(frame: CGRect(x: 0, y: 0, width: CALScreenWidth, height: 56))**

**view.backgroundColor = UIColor.colorWithHexColorString("FAFAFA")**

**view.addSubview(segController)**

**segController.center = view.center**

**return view**

**}()**

**lazy var segController: UISegmentedControl = {**

**let items = ["Data Unit","Feel The Base"]**

**let seg = UISegmentedControl(items: items)**

**seg.frame = CGRect(x: 0, y: 0, width: CALScreenWidth - 32, height: 32)**

**seg.selectedSegmentIndex = 0**

**seg.addTarget(self, action: #selector(segmentedControlChanged), for: .valueChanged)**

**return seg**

**}()**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setNav()**

**titleLab?.text = NSLocalizedString(self.title ?? "", comment: "")**

**self.view.backgroundColor = UIColor.white**

**self.tableView.tableHeaderView = headView**

**self.tableView.register(UINib(nibName: "BaseTableViewCell", bundle: nil), forCellReuseIdentifier: "baseCell")**

**self.tableView.register(UINib(nibName: "DataTableViewCell", bundle: nil), forCellReuseIdentifier: "DataCell")**

**self.tableView.reloadData()**

**let tap = UITapGestureRecognizer(target: self, action: #selector(tableTap))**

**tap.delegate = self**

**self.tableView.addGestureRecognizer(tap)**

**// 监听键盘弹出事件，控制toolbar位置**

**NotificationCenter.default.addObserver(self, selector: #selector(self.keyboardWillChangeFrame(node:)), name: UIResponder.keyboardWillShowNotification, object: nil)**

**}**

**override func viewWillDisappear(\_ animated: Bool) {**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "removeKeyboard")))**

**}**

**@objc func segmentedControlChanged(){**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "removeKeyboard")))**

**if segController.selectedSegmentIndex == 0 {**

**isDataView = true**

**}else{**

**isDataView = false**

**}**

**}**

**// MARK: - Table view data source**

**override func numberOfSections(in tableView: UITableView) -> Int {**

**// #warning Incomplete implementation, return the number of sections**

**return 1**

**}**

**override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**// #warning Incomplete implementation, return the number of rows**

**return dataArray.count**

**}**

**override func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {**

**return 56**

**}**

**override func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {**

**if isDataView {**

**if let cell = tableView.dequeueReusableCell(withIdentifier: "baseCell", for: indexPath) as? BaseTableViewCell{**

**cell.delegate = self**

**cell.model = dataArray[indexPath.row]**

**return cell**

**}else{**

**return UITableViewCell()**

**}**

**}else{**

**if let cell = tableView.dequeueReusableCell(withIdentifier: "DataCell", for: indexPath) as? DataTableViewCell{**

**cell.delegate = self**

**cell.model = dataArray[indexPath.row]**

**return cell**

**}else{**

**return UITableViewCell()**

**}**

**}**

**}**

**override func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {**

**return UIView()**

**}**

**@objc func tableTap(){**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "removeKeyboard")))**

**}**

**@objc func keyboardWillChangeFrame(node : Notification){**

**//        changeTextEffectsWindowsHostViewFrame()**

**}**

**func changeTextEffectsWindowsHostViewFrame() {**

**var keyboardWindow:UIWindow?**

**var inputSetHostView:UIView?**

**for testWindow in UIApplication.shared.windows {**

**if testWindow.description.hasPrefix("<UIRemoteKeyboardWindow") {**

**keyboardWindow = testWindow**

**break**

**}**

**}**

**if (keyboardWindow == nil) {return}**

**for possibleKeyboard in keyboardWindow!.subviews {**

**if possibleKeyboard.description.hasPrefix("<UIInputSetContainerView") {**

**possibleKeyboard.subviews.first?.frame.size = CGSize(width: CALScreenWidth, height: 500)**

**}**

**}**

**}**

**}**

**extension LJ\_DataTableViewController:BaseTableViewCellDelegete{**

**func changeValue(value: String, model: LJ\_BaseModel) {**

**if isDataView {**

**let baseValue = CalculatorBrain().calculation(str: model.formulaToBase.replacingOccurrences(of: "r", with: "\(value)"))**

**dataArray = LJ\_UnitManger.getModelWith(fileName: "Data", baseValue: baseValue)**

**dataArray[model.index].base\_Value = "\(value)"**

**}else{**

**var baseValue = "0"**

**switch model.type {**

**case .Binary:**

**baseValue = value.BinaryToDecimal()**

**case .Octal:**

**baseValue = value.OctalToDecimal()**

**case .Hex:**

**baseValue = value.HexToDecimal()**

**default:**

**baseValue = value**

**}**

**dataArray = LJ\_UnitManger.getModel(fileName: "FeelTheBase", baseValue: baseValue)**

**}**

**self.tableView.reloadData()**

**}**

**func moveView(y: CGFloat) {**

**UIView.animate(withDuration: 0.3) {**

**self.view.frame.origin.y = self.view.frame.origin.y - y**

**}**

**}**

**}**

**import UIKit**

**class LJ\_DateTableViewController: LJ\_BaseTableViewController,StoryboardLoadable {**

**@objc class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let instance = LJ\_DateTableViewController.loadStoryboard(name: "Main")**

**if let title = params?["title"] as? String{**

**instance.title = title**

**}**

**return instance**

**}**

**@IBOutlet weak var startTimeLab: UILabel!**

**@IBOutlet weak var endTimeLab: UILabel!**

**@IBOutlet weak var datePick: UIDatePicker!**

**@IBOutlet var datePickView: UIView!**

**var isStart = true**

**@IBOutlet weak var dayLab: UILabel!**

**@IBOutlet weak var weekLab: UILabel!**

**@IBOutlet weak var monthLab: UILabel!**

**@IBOutlet weak var yearLab: UILabel!**

**@IBOutlet weak var hourLab: UILabel!**

**@IBOutlet weak var minuteLab: UILabel!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setNav()**

**titleLab?.text = NSLocalizedString(self.title ?? "", comment: "")**

**datePickView.frame = UIScreen.main.bounds**

**setCurrentValue()**

**}**

**func setCurrentValue() {**

**let date = Date()**

**let dateStr = formateDate(date: date)**

**startTimeLab.text = dateStr**

**endTimeLab.text = dateStr**

**setValueforView(startDate: date, endDate: date)**

**}**

**// MARK: - Table view data source**

**override func numberOfSections(in tableView: UITableView) -> Int {**

**// #warning Incomplete implementation, return the number of sections**

**return 2**

**}**

**override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**if section == 0 {**

**return 2**

**}else{**

**return 6**

**}**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor.colorWithHexColorString("FAFAFA")**

**return view**

**}**

**override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 10**

**}**

**override func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {**

**if indexPath.section == 0 {**

**if indexPath.row == 0 {**

**datePick.setDate(stringDate(startTimeLab.text ?? ""), animated: true)**

**UIApplication.shared.keyWindow?.addSubview(datePickView)**

**isStart = true**

**}**

**if indexPath.row == 1 {**

**datePick.setDate(stringDate(endTimeLab.text ?? ""), animated: true)**

**UIApplication.shared.keyWindow?.addSubview(datePickView)**

**isStart = false**

**}**

**}**

**}**

**@IBAction func pickCancel(\_ sender: UIButton) {**

**datePickView.removeFromSuperview()**

**}**

**@IBAction func pickDown(\_ sender: UIButton) {**

**datePickView.removeFromSuperview()**

**let date = stringDate(startTimeLab.text ?? "")**

**let date2 = stringDate(endTimeLab.text ?? "")**

**if isStart {**

**if datePick.date >  date2{**

**return**

**}**

**startTimeLab.text = formateDate(date: datePick.date)**

**}else{**

**if datePick.date <  date{**

**return**

**}**

**endTimeLab.text = formateDate(date: datePick.date)**

**}**

**let stdate = stringDate(startTimeLab.text ?? "")**

**let endate = stringDate(endTimeLab.text ?? "")**

**setValueforView(startDate: stdate, endDate: endate)**

**}**

**func stringDate(\_ string:String, dateFormat:String = "yyyy.MM.dd EEEE") -> Date {**

**let formatter = DateFormatter()**

**formatter.locale = Locale.current**

**formatter.dateFormat = dateFormat**

**let date = formatter.date(from: string)**

**return date!**

**}**

**func formateDate(date:Date) -> String {**

**let timeFormatter = DateFormatter()**

**timeFormatter.dateFormat = "yyyy.MM.dd EEEE"**

**return timeFormatter.string(from: date) as String**

**}**

**func setValueforView(startDate:Date,endDate:Date){**

**let components = NSCalendar.current.dateComponents([.day], from: startDate, to: endDate)**

**let components2 = NSCalendar.current.dateComponents([.month,.day], from: startDate, to: endDate)**

**let components3 = NSCalendar.current.dateComponents([.year,.month,.day], from: startDate, to: endDate)**

**dayLab.text = "\(components.day ?? 0)" + NSLocalizedString("day", comment: "")**

**weekLab.text = "\((components.day ?? 0)/7)" + NSLocalizedString("week", comment: "") + "\((components.day ?? 0)%7)" + NSLocalizedString("day", comment: "")**

**monthLab.text = "\(components2.month ?? 0)" + NSLocalizedString("month", comment: "") + "\(components2.day ?? 0)"  + NSLocalizedString("day", comment: "")**

**yearLab.text = "\(components3.year ?? 0)" + NSLocalizedString("year", comment: "") + "\(components3.month ?? 0)" + NSLocalizedString("month", comment: "") + "\(components3.day ?? 0)" + NSLocalizedString("day", comment: "")**

**hourLab.text = "\((components.day ?? 0)\*24)"  + NSLocalizedString("hour", comment: "")**

**minuteLab.text = "\((components.day ?? 0)\*24\*60)"  + NSLocalizedString("minute", comment: "")**

**}**

**}**

**import UIKit**

**protocol BaseTableViewCellDelegete {**

**func changeValue(value:String,model:LJ\_BaseModel)**

**func moveView(y:CGFloat)**

**}**

**class BaseTableViewCell: UITableViewCell,UITextFieldDelegate,NormalKeyBoardDelgate {**

**@IBOutlet weak var abbUnit: UILabel!**

**@IBOutlet weak var unit: UILabel!**

**@IBOutlet weak var valueTF: UITextField!**

**var moveValue:CGFloat = 0.0**

**var model:LJ\_BaseModel?{**

**didSet{**

**if let unit = model?.unit,let abbUnit = model?.abbUnit,let realValue = model?.realValue {**

**self.unit.text = NSLocalizedString(unit, comment: "")**

**self.abbUnit.text = abbUnit**

**print(realValue)**

**if model?.base\_Value != "" {**

**self.valueTF.text = model?.base\_Value.removeLastZero()**

**}else{**

**self.valueTF.text = "\(realValue.formatterNumToSCI().removeLastZero())"**

**}**

**}**

**}**

**}**

**lazy var keyBoard: NormalKeyBoard = {**

**let kb = NormalKeyBoard.loadFromNib()**

**kb.frame = CGRect(x: 0, y: CALScreenHeight - 340 - CAL\_TabbarSafeBottomMargin, width: CALScreenWidth, height: 340 + CAL\_TabbarSafeBottomMargin)**

**kb.delegate = self**

**return kb**

**}()**

**var delegate:BaseTableViewCellDelegete?**

**override func awakeFromNib() {**

**super.awakeFromNib()**

**valueTF.delegate = self**

**valueTF.inputView = UIView()**

**NotificationCenter.default.addObserver(self, selector: #selector(removeKeyBoard), name: NSNotification.Name("removeKeyboard"), object: nil)**

**// Initialization code**

**}**

**func textFieldDidBeginEditing(\_ textField: UITextField) {**

**textField.text = ""**

**UIApplication.shared.keyWindow?.addSubview(keyBoard)**

**print(self.frame.origin.y + self.frame.size.height)**

**print(keyBoard.frame.height)**

**if keyBoard.frame.size.height > CALScreenHeight - self.frame.origin.y - self.frame.size.height {**

**moveValue =**

**self.frame.origin.y + self.frame.size.height - keyBoard.frame.origin.y**

**moveValue = keyBoard.frame.size.height - (CALScreenHeight - self.frame.origin.y - self.frame.size.height - CAL\_Navi\_Height)**

**print(moveValue)**

**delegate?.moveView(y: moveValue)**

**}**

**}**

**func textFieldDidEndEditing(\_ textField: UITextField, reason: UITextField.DidEndEditingReason) {**

**delegate?.changeValue(value: "\(textField.text ?? "1.0")" ,model: model ?? LJ\_BaseModel(json: "", baseValue: "1.0"))**

**delegate?.moveView(y: -moveValue)**

**moveValue = 0.0**

**self.valueTF.resignFirstResponder()**

**keyBoard.removeFromSuperview()**

**}**

**@objc func removeKeyBoard() {**

**self.valueTF.resignFirstResponder()**

**keyBoard.removeFromSuperview()**

**}**

**override func setSelected(\_ selected: Bool, animated: Bool) {**

**super.setSelected(selected, animated: animated)**

**// Configure the view for the selected state**

**}**

**func keyClick(type: KeyType) {**

**switch type {**

**case .digit(let num):**

**self.valueTF.text?.append("\(num)")**

**case .delete:**

**if self.valueTF.text?.count ?? 0 > 0 {**

**self.valueTF.text?.removeLast()**

**}**

**case .clear:**

**self.valueTF.text = ""**

**case.hex(let str):**

**self.valueTF.text?.append(str)**

**case .negative:**

**self.valueTF.text?.append("-")**

**default:**

**return**

**}**

**}**

**}**

**import UIKit**

**class DataTableViewCell: UITableViewCell,UITextFieldDelegate,NormalKeyBoardDelgate {**

**@IBOutlet weak var titleLab: UILabel!**

**@IBOutlet weak var valueTF: UITextField!**

**var delegate:BaseTableViewCellDelegete?**

**var model:LJ\_BaseModel?{**

**didSet{**

**if let unit = model?.unit,let realValue = model?.realValue,let type = model?.type {**

**self.titleLab.text = NSLocalizedString(unit, comment: "")**

**print(realValue)**

**self.valueTF.text = model?.realValue**

**keyBoard.dataType = type**

**}**

**}**

**}**

**lazy var keyBoard: SystemKeyBoard = {**

**let kb = SystemKeyBoard.loadFromNib()**

**kb.frame = CGRect(x: 0, y: CALScreenHeight - 375 - CAL\_TabbarSafeBottomMargin, width: CALScreenWidth, height: 375 + CAL\_TabbarSafeBottomMargin)**

**kb.delegate = self**

**return kb**

**}()**

**override func awakeFromNib() {**

**super.awakeFromNib()**

**valueTF.delegate = self**

**valueTF.inputView = UIView()**

**NotificationCenter.default.addObserver(self, selector: #selector(removeKeyBoard), name: NSNotification.Name("removeKeyboard"), object: nil)**

**}**

**func textFieldDidBeginEditing(\_ textField: UITextField) {**

**textField.text = ""**

**UIApplication.shared.keyWindow?.addSubview(keyBoard)**

**}**

**func textFieldDidEndEditing(\_ textField: UITextField, reason: UITextField.DidEndEditingReason) {**

**delegate?.changeValue(value: "\(textField.text ?? "1.0")" ,model: model ?? LJ\_BaseModel(json: "", binaryValue: "0"))**

**self.valueTF.resignFirstResponder()**

**keyBoard.removeFromSuperview()**

**}**

**@objc func removeKeyBoard() {**

**self.valueTF.resignFirstResponder()**

**keyBoard.removeFromSuperview()**

**}**

**func keyClick(type: KeyType) {**

**switch type {**

**case .digit(let num):**

**self.valueTF.text?.append("\(num)")**

**case .delete:**

**if self.valueTF.text?.count ?? 0 > 0 {**

**self.valueTF.text?.removeLast()**

**}**

**case .dot:**

**self.valueTF.text?.append(".")**

**case .clear:**

**self.valueTF.text = ""**

**case.hex(let str):**

**self.valueTF.text?.append(str)**

**case .negative:**

**self.valueTF.text?.append("-")**

**default:**

**return**

**}**

**}**

**override func setSelected(\_ selected: Bool, animated: Bool) {**

**super.setSelected(selected, animated: animated)**

**// Configure the view for the selected state**

**}**

**}**

**import UIKit**

**class FormulaView: UIView ,UIScrollViewDelegate,UITextFieldDelegate{**

**var scrollView: UIScrollView?**

**var formulaTF:UITextField?**

**var inputStr:String = ""{**

**didSet{**

**formulaTF?.text = inputStr**

**}**

**}**

**override init(frame: CGRect) {**

**super.init(frame: frame)**

**scrollView = UIScrollView(frame: CGRect(x: 0, y: 0, width: frame.size.width - 60, height: frame.size.height))**

**scrollView?.delegate = self**

**scrollView?.contentSize = CGSize(width: frame.size.width - 60, height: frame.size.height)**

**self.addSubview(scrollView!)**

**formulaTF = UITextField(frame: CGRect(x: 20, y: 0, width: frame.size.width - 80, height: frame.size.height))**

**formulaTF?.delegate = self**

**formulaTF?.inputView = UIView(frame: CGRect.zero)**

**formulaTF?.textAlignment = .right**

**scrollView?.addSubview(formulaTF!)**

**}**

**required init?(coder: NSCoder) {**

**fatalError("init(coder:) has not been implemented")**

**}**

**}**

**import UIKit**

**protocol Stack {**

**/// 持有的元素类型**

**associatedtype Element**

**/// 是否为空**

**var isEmpty: Bool { get }**

**/// 栈的大小**

**var size: Int { get }**

**/// 栈顶元素**

**var peek: Element? { get }**

**/// 栈顶元素**

**var top: Int { get }**

**/// 进栈**

**mutating func push(\_ newElement: Element)**

**/// 出栈**

**mutating func pop()**

**}**

**struct StackSymbol: Stack {**

**var top: Int = -1**

**typealias Element = String**

**var isEmpty: Bool { return stack.isEmpty }**

**var size: Int { return stack.count }**

**var peek: Element? { return stack.last }**

**private var stack = [Element]()**

**mutating func push(\_ newElement: Element) {**

**stack.append(newElement)**

**top = top  + 1**

**//        print("StackSymbol push ------stack:\(address(o: &self)) top: \(top)")**

**}**

**mutating func pop() {**

**top = top  - 1**

**\_ = stack.popLast()**

**//        print("StackSymbol pop ------stack:\(address(o: &self)) top: \(top)")**

**}**

**///退到栈顶**

**mutating func popToTop()->Bool{**

**if self.top == -1 {**

**return false**

**}else{**

**self.top = self.top - 1**

**\_ = stack.popLast()**

**return true**

**}**

**}**

**}**

**struct StackNum: Stack {**

**typealias Element = String**

**var isEmpty: Bool { return stack.isEmpty }**

**var size: Int { return stack.count }**

**var peek: Element? { return stack.last }**

**var top: Int = -1**

**private var stack = [Element]()**

**mutating func push(\_ newElement: Element) {**

**stack.append(newElement)**

**top = top + 1**

**//        print("StackNum push ------stack:\(address(o: &self)) top: \(top)")**

**}**

**mutating func pop() {**

**top = top - 1**

**\_ = stack.popLast()**

**//        print("StackNum pop ------stack:\(address(o: &self)) top: \(top)")**

**}**

**///退到栈顶**

**mutating func popToTop()->Bool{**

**if self.top == -1 {**

**return false**

**}else{**

**while self.top != 0 {**

**self.pop()**

**}**

**return true**

**}**

**}**

**}**

**class CalculatorBrain: NSObject {**

**var inputMut:Array<String> = []**

**var flag = 0**

**var priority = [[1,1,-1,-1,-1,1,1],[1,1,-1,-1,-1,1,1],[1,1,1,1,-1,1,1],[1,1,1,1,-1,1,1],[-1,-1,-1,-1,-1,0,-2],[ 1,1,1,1,-2,1,1],[-1,-1,-1,-1,-1,-2,0]]**

**override init() {**

**super.init()**

**}**

**func calculationPercent(){**

**}**

**func calculationCheck(str:String) -> Bool {**

**let str1 = "\(str)#"**

**var stackSymbol = StackSymbol()**

**//        stackSymbol.top = -1**

**var x:String = ""**

**var i = 0**

**//        var flag = 0**

**x = String(str1.subCharacter(i: i))**

**while x != "#" {**

**if isBracket(x) {**

**flag = 1**

**if x == "(" {**

**stackSymbol.push(x)**

**}else{**

**if stackSymbol.popToTop() {**

**return false**

**}**

**}**

**}**

**i+=1**

**x = String(str1.subCharacter(i: i))**

**}**

**return true**

**}**

**func calculation(str:String,isRadina:Bool? = true) -> String {**

**//        print("op:-----(\(str))")**

**var newStr = str**

**//        newStr = self.addMissBrackets(string: newStr)**

**newStr = newStr + "#"**

**var stackSymbol = StackSymbol()**

**stackSymbol.top = -1**

**stackSymbol.push("#")**

**var stackNum = StackNum()**

**stackNum.top = -1**

**var x:Character = " "**

**var i = 0**

**x = newStr.subCharacter(i: i)**

**//        print("000000000000000----\(newStr)")**

**while x != "#" || stackSymbol.peek != "#" {**

**if x.isNumber {**

**var data:String = "0.0"**

**var v:String = "10"**

**while x.isNumber || x == "." {**

**if x == "." {**

**i += 1**

**x = newStr.subCharacter(i: i)**

**while x.isNumber {**

**data = NSString.floatOne(data, calculationType: .forAdd, floatTwo: NSString.floatOne("\(x)", calculationType: .forDivide, floatTwo: v))**

**//                            data = data + Float80(Float80("\(x)") ?? 0 - 0)/Float80(v)**

**v = NSString.floatOne(v, calculationType: .forMultiply, floatTwo: "10")**

**//                            v = v \* 10**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}**

**}else{**

**data = NSString.floatOne(NSString.floatOne(data, calculationType: .forMultiply, floatTwo: "10"), calculationType: .forAdd, floatTwo: "\(x)")**

**//                        data = data \* 10 + Float80(Float80("\(x)") ?? 0) - 0**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}**

**}**

**stackNum.push(StackNum.Element(data))**

**}else{**

**if x == "f" || x == "g" || x == "h" || x == "j" || x == "k" || x == "l" || x == "m" || x == "q" || x == "㏑" || x == "F" || x == "G" || x == "H" || x == "J" || x == "K"{**

**var ms = ""**

**var j = i + 1**

**var k = 0**

**repeat{**

**j += 1**

**if newStr.subCharacter(i: j) == "(" {**

**k += 1**

**}**

**if newStr.subCharacter(i: j) == ")" {**

**k -= 1**

**}**

**ms = ms + "\(newStr.subCharacter(i: j))"**

**}while !(newStr.subCharacter(i: j) == ")" && k == -1)**

**var mixRex:String = "0.0"**

**if ms.regexCharCount(pattern: "[()]") > 0 {**

**ms = self.addMissBrackets(string: ms)**

**mixRex = calculation(str: ms)**

**}else{**

**mixRex = calculation(str: ms)**

**}**

**var num = Double(mixRex) ?? 0.0**

**//是角度，并且是三角函数运算转弧度**

**if isRadina! && (x == "f" || x == "g" || x == "h" || x == "j" || x == "k" || x == "l" || x == "m" || x == "F" || x == "G" || x == "H" || x == "J" || x == "K") {**

**num = Double.pi / 180.0 \* (num)**

**}**

**var SR:Double = 0.0**

**if x == "f" {        //正弦**

**SR = sin(num )**

**}else if x == "g"{**

**SR = cos(num)    //余弦**

**}else if x == "h"{**

**SR = tan(num)    //正切**

**}else if x == "F"{**

**SR = asin(num)   //反正弦**

**}else if x == "G"{**

**SR = acos(num)   //反余弦**

**}else if x == "H"{**

**SR = atan(num)   //反正切**

**}else if x == "j"{**

**SR = 1.0/tan(num) //余切**

**}else if x == "J"{**

**SR = 1.0/cos(num) //正割**

**}else if x == "K"{**

**SR = 1.0/sin(num) //余割**

**}else if x == "k"{**

**SR = sinh(num)**

**}else if x == "l"{**

**SR = cosh(num)**

**}else if x == "m"{**

**SR = tanh(num)**

**}else if x == "q"{**

**SR = log10(num)**

**}else if x == "㏑"{**

**SR = log(num)**

**}**

**stackNum.push(StackNum.Element(SR))**

**//                    let count = ms.regexCharCount(pattern: "[fghjklmq㏑]")**

**i = j + 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "!"{**

**let z = stackNum.peek ?? "0"**

**stackNum.pop()**

**let fac:String = "\(factorial(Int(z) ?? 0))"**

**stackNum.push(fac)**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "v"{**

**let z = stackNum.peek ?? "0"**

**stackNum.pop()**

**let fac:String = NSString.floatOne("1", calculationType: .forDivide, floatTwo: z)**

**stackNum.push(fac)**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "%"{**

**let z = stackNum.peek ?? "0"**

**stackNum.pop()**

**let fac:String = NSString.floatOne(z, calculationType: .forDivide, floatTwo: "100.0")**

**stackNum.push(fac)**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "e"{**

**stackNum.push(StackNum.Element(Double(2.718281828)))**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "π"{**

**stackNum.push(StackNum.Element(Double.pi))**

**i += 1**

**x = newStr.subCharacter(i: i)**

**}else if x == "p"{**

**var j = i + 1**

**var data:String = "0.0"**

**var v:String = "10"**

**x = newStr.subCharacter(i: j)**

**while x.isNumber {**

**while x.isNumber || x == "." {**

**if x == "." {**

**j += 1**

**x = newStr.subCharacter(i: j)**

**while x.isNumber {**

**data = NSString.floatOne(data, calculationType: .forAdd, floatTwo: NSString.floatOne("\(x)", calculationType: .forDivide, floatTwo: v))**

**//                                    data = data + Float80(Float80("\(x)") ?? 0 - 0)/Float80(v)**

**v = NSString.floatOne(v, calculationType: .forMultiply, floatTwo: "10")**

**//                                    v = v \* 10**

**j += 1**

**x = newStr.subCharacter(i: j)**

**}**

**}else{**

**data = NSString.floatOne(NSString.floatOne(data, calculationType: .forMultiply, floatTwo: "10"), calculationType: .forAdd, floatTwo: "\(x)")**

**//                                data = data \* 10 + Float80(Float80("\(x)") ?? 0) - 0**

**j += 1**

**x = newStr.subCharacter(i: j)**

**}**

**}**

**}**

**stackNum.push(StackNum.Element("-" + data))**

**i = j**

**x = newStr.subCharacter(i: i)**

**}else{**

**guard let \_ = stackSymbol.peek else {**

**return stackNum.peek!**

**}**

**switch compare(a: Character(stackSymbol.peek ?? ""), b: x) {**

**case -1:**

**stackSymbol.push("\(x)")**

**i += 1**

**x = newStr.subCharacter(i: i)**

**break**

**case 1:**

**guard let left = stackNum.peek else { return "0" }**

**stackNum.pop()**

**guard let right = stackNum.peek else { return "0" }**

**stackNum.pop()**

**stackNum.push(calculationValue(left: "\(left)", symbol: stackSymbol.peek ?? "", right: "\(right)"))**

**\_ = stackSymbol.popToTop()**

**break**

**case 0:**

**\_ = stackSymbol.popToTop()**

**i += 1**

**x = newStr.subCharacter(i: i)**

**break**

**case -2:**

**stackNum.pop()**

**stackNum.push(StackNum.Element("\(x)") )**

**\_ = stackSymbol.popToTop()**

**x = newStr.subCharacter(i: i)**

**default:**

**break**

**}**

**}**

**}**

**}**

**return stackNum.peek ?? "Error"**

**}**

**func addMissBrackets(string:String) -> String {**

**let count = string.filter({$0 == "("}).count - string.filter({$0 == ")"}).count**

**var newStr = string**

**for \_ in 0 ..< abs(count) {**

**if count > 0 {**

**newStr.append(")")**

**}else if count < 0{**

**if newStr.last == ")" {**

**newStr.removeLast()**

**}**

**}**

**}**

**return newStr**

**}**

**func calculaSub(string:String) -> String {**

**return ""**

**}**

**func calculationValue(left:String,symbol:String,right:String) -> String{**

**let x = CLongDouble(left) ?? 0**

**let y = CLongDouble(right) ?? 0**

**var res:String = ""**

**var decimalPlaces = 0**

**decimalPlaces = y.getDecimalplaces() > x.getDecimalplaces() ? y.getDecimalplaces() : x.getDecimalplaces()**

**switch symbol {**

**case "+":**

**res = NSString.floatOne(left, calculationType: .forAdd, floatTwo: right) as String**

**//            res = (CLongDouble(resStr))?.roundTo(places: resStr.getStringDecimalplaces()) ?? 0**

**//            res = (y + x).roundTo(places: decimalPlaces);**

**case "-":**

**res = NSString.floatOne(right, calculationType: .forSubtract, floatTwo: left) as String**

**//            res =  (y - x).roundTo(places: decimalPlaces);**

**case "×":**

**res = NSString.floatOne(left, calculationType: .forMultiply, floatTwo: right) as String**

**//            res = y \* Float80(x)**

**//            var resDecimal = 0**

**//            if x.getDecimalplaces() == 0 || y.getDecimalplaces() == 0 {**

**//                resDecimal = x.getDecimalplaces() + y.getDecimalplaces() + 1**

**//            }else{**

**//                if x.getDecimalplaces() == 1 || y.getDecimalplaces() == 1 {**

**//                    resDecimal = x.getDecimalplaces() + y.getDecimalplaces() + 1**

**//                }else{**

**//                    resDecimal = x.getDecimalplaces() \* y.getDecimalplaces()**

**//                }**

**//            }**

**//            if resDecimal > decimalPlaces {**

**//                decimalPlaces = resDecimal > 19 ? 19 : resDecimal**

**//            }else{**

**//                decimalPlaces = resDecimal**

**//            }**

**//            res = (y \* Float80(x)).roundTo(places: decimalPlaces)**

**case "x":**

**res = NSString.floatOne(right, calculationType: .forPowerof10, floatTwo: left) as String**

**//            res = y \* Float80(x)**

**//            var resDecimal = 0**

**//            if x.getDecimalplaces() == 0 || y.getDecimalplaces() == 0 {**

**//                resDecimal = x.getDecimalplaces() + y.getDecimalplaces() + 1**

**//            }else{**

**//                if x.getDecimalplaces() == 1 || y.getDecimalplaces() == 1 {**

**//                    resDecimal = x.getDecimalplaces() + y.getDecimalplaces() + 1**

**//                }else{**

**//                    resDecimal = x.getDecimalplaces() \* y.getDecimalplaces()**

**//                }**

**//            }**

**//            if resDecimal > decimalPlaces {**

**//                decimalPlaces = resDecimal > 19 ? 19 : resDecimal**

**//            }else{**

**//                decimalPlaces = resDecimal**

**//            }**

**//            let data = self.multiWithExponentRecursion(base: 10.0, exponet: Int(x))**

**//            res = y \* Float80(data)**

**case "^":**

**//        res = y \* Float80(x)**

**//        let resDecimal = x.getDecimalplaces() \* y.getDecimalplaces()**

**//        if resDecimal > decimalPlaces {**

**//            decimalPlaces = resDecimal > 19 ? 19 : resDecimal**

**//        }**

**//        let data = powl(y, x)**

**//        res = CLongDouble(data)**

**if NSString.floatOne(left, calculationType: .forSquare, floatTwo: right) != nil {**

**res = NSString.floatOne(left, calculationType: .forSquare, floatTwo: right) as String**

**}else{**

**res = "Error"**

**}**

**case "÷":**

**res = NSString.floatOne(right, calculationType: .forDivide, floatTwo: left) as String**

**//            if x == 0 {**

**//                res = -0**

**//            }else{**

**//                let resDecimal = (y / x).getDecimalplaces()**

**//                if resDecimal > decimalPlaces {**

**//                    decimalPlaces = resDecimal > 19 ? 19 : resDecimal**

**//                }**

**//                res = (y / Float80(x)).roundTo(places: decimalPlaces - 1)**

**//**

**//            }**

**case "√":**

**let resDecimal = (y / CLongDouble(x)).getDecimalplaces()**

**if resDecimal > decimalPlaces {**

**decimalPlaces = resDecimal > 19 ? 19 : resDecimal**

**}**

**res = "\(CLongDouble(self.squar(left: Double(y), right: Double(x))))"**

**default:**

**res = "0";**

**}**

**return res**

**}**

**func compare(a:Character,b:Character) -> Int {**

**var i = -1**

**var j = -1**

**switch a {**

**case "#":**

**i += 1**

**fallthrough**

**case ")":**

**i += 1**

**fallthrough**

**case "(":**

**i += 1**

**fallthrough**

**case "÷","√":**

**i += 1**

**fallthrough**

**case "×","x","^":**

**i += 1**

**fallthrough**

**case "-":**

**i += 1**

**fallthrough**

**case "+":**

**i += 1**

**fallthrough**

**default:**

**break;**

**}**

**switch b {**

**case "#":**

**j += 1**

**fallthrough**

**case ")":**

**j += 1**

**fallthrough**

**case "(":**

**j += 1**

**fallthrough**

**case "÷","√":**

**j += 1**

**fallthrough**

**case "×","x","^":**

**j += 1**

**fallthrough**

**case "-":**

**j += 1**

**fallthrough**

**case "+":**

**j += 1**

**fallthrough**

**default:**

**break;**

**}**

**if i >= 0 && j >= 0 {**

**return priority[i][j]**

**}else{**

**return -2**

**}**

**}**

**func isBracket(\_ x:String) -> Bool{**

**if x == "(" || x == ")" || x == "f"{**

**return true;**

**} else {**

**return false;**

**}**

**}**

**func factorial(\_ n:Int) -> CLongDouble {**

**if n == 0 {**

**return 1**

**}else{**

**return factorial(n-1) \* CLongDouble(Int64(n))**

**}**

**}**

**///left:开方次数 right：开方数**

**func squar(left:Double,right:Double) -> Double{**

**var res:Double = 0.0**

**var guess:Double = 0.0**

**if right < 0 {**

**return Double.nan**

**}else{**

**res = right / 2.0**

**guess = res - res \* (1 - right \* pow(res, -left))/left**

**repeat{**

**res = guess**

**guess = res - res \* (1 - right \* pow(res, -left))/left**

**}while (fabs(res - guess) >= 0.00000001)**

**}**

**return res**

**}**

**func multiWithExponentRecursion(base:Double,exponet:Int) -> Double {**

**if exponet == 0 {**

**return 1**

**}**

**if exponet == 1 {**

**return base**

**}**

**var result:Double = self.multiWithExponent(base: base, exponet: exponet>>1)**

**result \*= result**

**if  (exponet & 1) == 1 {**

**result \*= base**

**}**

**return result**

**}**

**func multiWithExponent(base:Double,exponet:Int) -> Double {**

**var result:Double = 1**

**for \_ in 0..<exponet {**

**result \*= base**

**}**

**return result**

**}**

**}**

**extension String {**

**var length: Int {**

**return count**

**}**

**subscript (i: Int) -> String {**

**return self[i ..< i + 1]**

**}**

**func substring(fromIndex: Int) -> String {**

**return self[min(fromIndex, length) ..< length]**

**}**

**func substring(toIndex: Int) -> String {**

**return self[0 ..< max(0, toIndex)]**

**}**

**func subCharacter(i: Int) -> Character {**

**return Character(self[i ..< i + 1])**

**}**

**subscript (r: Range<Int>) -> String {**

**let range = Range(uncheckedBounds: (lower: max(0, min(length, r.lowerBound)),**

**upper: min(length, max(0, r.upperBound))))**

**let start = index(startIndex, offsetBy: range.lowerBound)**

**let end = index(start, offsetBy: range.upperBound - range.lowerBound)**

**return String(self[start ..< end])**

**}**

**}**

**extension Character{**

**func toInt() -> Int {**

**return  Int((String(self) as NSString).character(at: 0))**

**}**

**}**

**extension CLongDouble {**

**public func roundTo(places: Int) -> CLongDouble {**

**if self.getDecimalplaces() >  places{**

**let divisor = pow(10.0, CLongDouble(places))**

**return (self \* divisor).rounded(.down) / divisor**

**}else{**

**return self**

**}**

**}**

**public func getDecimalplaces() -> Int{**

**let str = "\(self)"**

**var count = 0**

**var sen = 0**

**if str.contains("e") {**

**let array = str.split(separator: "e")**

**sen = Int("\(array[1])") ?? 0**

**var deci = 0**

**if array[1].contains(".") {**

**let deciCount = "\(array[1])".split(whereSeparator: {$0 == "."})[1].count**

**deci = array[0].count - deciCount - 1**

**}else{**

**deci = array[0].count**

**}**

**return abs(sen) + deci**

**}**

**if str.contains(".") {**

**let sub = str.suffix(from: str.firstIndex(of: ".") ?? str.endIndex)**

**for c in sub.reversed() {**

**if c == "0" {**

**count += 1**

**}else{**

**break**

**}**

**}**

**return sub.count - 1 - count**

**}else{**

**return 0**

**}**

**}**

**}**

**import UIKit**

**class BaseCalViewController: BaseViewController,StoryboardLoadable {**

**@objc  class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let vc = BaseCalViewController.loadStoryboard(identifier: "BasicCalculator")**

**vc.hidesBottomBarWhenPushed = true**

**return vc**

**}**

**@IBOutlet var calculatorBtns: [CalculatorBtn]!**

**@IBOutlet weak var calTF: UITextField!**

**@IBOutlet weak var historyView: UIView!**

**@IBOutlet weak var historyLab: UILabel!**

**@IBOutlet weak var calResultLab: UILabel!**

**var calculatorBrain:CalculatorBrain?**

**var inputStr:String = ""{**

**didSet{**

**calTF.attributedText = formatInputStr(string: inputStr)**

**}**

**}**

**var result:String = ""**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setCalculatorBtn()**

**calculatorBrain = CalculatorBrain()**

**setNav()**

**self.titleLab?.text = NSLocalizedString("Basic calculator", comment: "")**

**calResultLab.adjustsFontSizeToFitWidth = true**

**calTF.inputView = UIView()**

**}**

**func setCalculatorBtn(){**

**calculatorBtns.forEach { button in**

**if button.tag > 99 && button.tag<110{**

**button.calType = .digit(button.tag - 100)**

**}**

**if button.tag == 110{**

**button.calType = .dot**

**}**

**if button.tag == 201{**

**button.calType = .command(.clear)**

**}**

**if button.tag == 202{**

**button.calType = .command(.flip)**

**}**

**if button.tag == 203{**

**button.calType = .command(.percent)**

**}**

**if button.tag == 301{**

**button.calType = .op(.divide)**

**}**

**if button.tag == 302{**

**button.calType = .op(.multiply)**

**}**

**if button.tag == 303{**

**button.calType = .op(.minus)**

**}**

**if button.tag == 304{**

**button.calType = .op(.plus)**

**}**

**if button.tag == 305{**

**button.calType = .op(.equal)**

**}**

**if button.calType?.description == "p"{**

**button.setBackgroundImage(UIImage(named: "p"), for: .normal)**

**}else if button.calType?.description == "."{**

**button.setBackgroundImage(UIImage(named: "dot"), for: .normal)**

**}**

**else{**

**button.setBackgroundImage(UIImage(named: button.calType?.description ?? ""), for: .normal)**

**}**

**button.addTarget(self, action: #selector(calBtnClick(sender:)), for: .touchUpInside)**

**}**

**}**

**@objc func calBtnClick(sender:CalculatorBtn){**

**if self.calResultLab.text == "Error!" {**

**//            calculatorBrain?.inputMut.removeAll()**

**//            self.calResultLab.text = ""**

**}**

**guard (sender.calType?.description) != nil else {**

**return**

**}**

**guard let type = sender.calType else {**

**return**

**}**

**print(type.description)**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**if !inputStr.formulaCanAppend(Character(type.description)){**

**print("不可拼接")**

**return**

**}**

**switch type {**

**case .command(.clear):**

**calculatorBrain?.inputMut.removeAll()**

**self.calResultLab.text = ""**

**inputStr = ""**

**break**

**case .command(.flip):///正负号处理**

**calculatorBrain?.inputMut.append("p")**

**break**

**case .op(.equal):**

**if calculatorBrain?.inputMut.count == 0 {**

**return**

**}**

**if !inputStr.isRightFormula(){**

**self.calResultLab.text = "Error!"**

**return**

**}**

**let temp = calculatorBrain?.calculation(str: inputStr)**

**print(temp as Any)**

**result = "\(temp!)".removeLastZero()**

**self.calResultLab.text = result.formateToMoneyFount()**

**self.historyLab.text = result.formateToMoneyFount()**

**CalHistoryDBManager.shareManger().insertResult(tableName: "CalHis", result: result.formateToMoneyFount(), formula: inputStr)**

**break**

**case.op(let synmbol):**

**calculatorBrain?.inputMut.append(synmbol.rawValue)**

**break;**

**case.digit(let i):**

**var str = ""**

**for string in calculatorBrain!.inputMut.reversed() {**

**let bool = (Int(string) ?? -1 >= 0 && Int(string) ?? -1 <= 9) || string == "."**

**if bool {**

**str.append(string)**

**}else{**

**break**

**}**

**}**

**if !str.contains(".") && str.length > 14 {**

**return**

**}**

**calculatorBrain?.inputMut.append("\(i)")**

**break**

**case .command(.percent):**

**calculatorBrain?.inputMut.append("%")**

**case .dot:**

**if calculatorBrain?.flag == 0 {**

**calculatorBrain?.inputMut.append(".")**

**calculatorBrain?.flag = 1**

**}else{**

**calculatorBrain?.inputMut.append(".")**

**}**

**default:**

**break**

**}**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**}**

**func saveHistory(success:(Bool)->()) {**

**success(true)**

**}**

**@IBAction func deleteBtn(\_ sender: UIButton) {**

**if calculatorBrain?.inputMut.count ?? 0 > 0 {**

**calculatorBrain?.inputMut.removeLast()**

**inputStr = calculatorBrain?.inputMut.joined() ?? ""**

**}**

**}**

**func formatInputStr(string:String) -> NSMutableAttributedString {**

**var forStr = string**

**forStr = forStr.replacingOccurrences(of: "p", with: "-")**

**forStr = forStr.formateToMoneyFount()**

**let ranges = forStr.regexWithSymbol()**

**print("--------\(forStr)")**

**let att = NSMutableAttributedString(string: forStr)**

**att.addAttributes([NSAttributedString.Key.foregroundColor : UIColor.colorWithHexColorString("505050"),NSAttributedString.Key.font:UIFont.systemFont(ofSize: 18)], range: NSRange(location: 0, length: forStr.length))**

**for range in ranges {**

**att.addAttributes([NSAttributedString.Key.foregroundColor : UIColor.colorWithHexColorString("FF9734")], range: range)**

**if range.location > 0 {**

**att.addAttributes([NSAttributedString.Key.kern : NSNumber(value: 5.0)], range: NSRange(location: range.location - 1, length: range.length + 1))**

**}**

**}**

**return att**

**}**

**@IBAction func historyClick(\_ sender: UITapGestureRecognizer) {**

**let vc = CalculatorHistoryViewController.loadStoryboard(name: "Main")**

**vc.tableName = "CalHis"**

**self.navigationController?.present(vc, animated: true, completion: nil)**

**}**

**}**

**extension BaseCalViewController:UITextFieldDelegate{**

**func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {**

**if string.count == 0 {**

**self.calculatorBrain?.inputMut.removeSubrange(Range(range)!)**

**}else{**

**var i = 0**

**for c in string {**

**self.calculatorBrain?.inputMut.insert("\(c)", at: range.location + i)**

**i += 1**

**}**

**}**

**return true**

**}**

**}**

**import UIKit**

**class CalculatorHistoryViewController: UIViewController,UITableViewDelegate,UITableViewDataSource,StoryboardLoadable {**

**@objc  class func alloc(withRouterParams params: [AnyHashable : Any]?) -> Any? {**

**let vc = CalculatorHistoryViewController.loadStoryboard(identifier: "CalculatorHistoryViewController")**

**vc.hidesBottomBarWhenPushed = true**

**return vc**

**}**

**var tableName:String = ""**

**var resArr:Array<Dictionary<String,String>> = []**

**@IBOutlet weak var tableView: UITableView!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**requestData()**

**// Do any additional setup after loading the view.**

**}**

**func requestData() {**

**resArr = CalHistoryDBManager.shareManger().selectAllResult(tableName: tableName)**

**tableView.reloadData()**

**}**

**@IBAction func closeClick(\_ sender: UIButton) {**

**self.dismiss(animated: true, completion: nil)**

**}**

**@IBAction func clearClick(\_ sender: UIButton) {**

**CalHistoryDBManager.shareManger().deleteAllResult(tableName: tableName)**

**requestData()**

**}**

**func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {**

**return 64**

**}**

**func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**return resArr.count**

**}**

**func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {**

**let cell = tableView.dequeueReusableCell(withIdentifier: "hisCell", for: indexPath)**

**(cell.viewWithTag(101) as? UILabel)?.text = resArr[indexPath.row]["result"]**

**(cell.viewWithTag(102) as? UILabel)?.text = (resArr[indexPath.row]["formula"] ?? "") + "="**

**return cell**

**}**

**func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor.colorWithHexColorString("FAFAFA")**

**return view**

**}**

**func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 10**

**}**

**func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {**

**let actionSheet = UIAlertController(title: "", message: "Copy", preferredStyle: .actionSheet)**

**let actionR = UIAlertAction(title: "Copy Results", style: .default) { \_ in**

**let pas = UIPasteboard.general**

**pas.string = self.resArr[indexPath.row]["result"]**

**}**

**let actionRE = UIAlertAction(title: "Copy Equations and Results", style: .default) { \_ in**

**let pas = UIPasteboard.general**

**pas.string = (self.resArr[indexPath.row]["formula"] ?? "0") + "=" + (self.resArr[indexPath.row]["result"] ?? "0")**

**}**

**let calcel = UIAlertAction(title: "Cancel", style: .cancel) { \_ in**

**}**

**actionSheet.addAction(actionR)**

**actionSheet.addAction(actionRE)**

**actionSheet.addAction(calcel)**

**self.present(actionSheet, animated: true, completion: nil)**

**}**

**}**

**import UIKit**

**let digitBgColor = UIColor(red: 0.31, green: 0.31, blue: 0.31, alpha: 1)**

**let operatorBgColor = UIColor(red: 1, green: 0.59, blue: 0.2, alpha: 1)**

**let commandBgColor = UIColor(red: 0.9, green: 0.9, blue: 0.9, alpha: 1)**

**let scBgColor = UIColor(red: 0.96, green: 0.96, blue: 0.96, alpha: 1)**

**@IBDesignable class CalculatorBtn: UIButton {**

**@IBInspectable @objc var calculatorType:String?**

**var calType:CalculatorButton?{**

**didSet{**

**switch calType {**

**case .digit,.dot:**

**self.backgroundColor = digitBgColor**

**self.setTitleColor(UIColor.white, for: .normal)**

**self.setTitleColor(UIColor.white, for: .selected)**

**break**

**case .op:**

**self.backgroundColor = operatorBgColor**

**self.setTitleColor(UIColor.white, for: .normal)**

**self.setTitleColor(UIColor.white, for: .selected)**

**break**

**case .command:**

**self.backgroundColor = commandBgColor**

**self.setTitleColor(UIColor(red: 0.31, green: 0.31, blue: 0.31, alpha: 1), for: .normal)**

**self.setTitleColor(UIColor(red: 0.31, green: 0.31, blue: 0.31, alpha: 1), for: .selected)**

**break**

**case .sc:**

**self.backgroundColor = scBgColor**

**self.setTitleColor(UIColor(red: 0.31, green: 0.31, blue: 0.31, alpha: 1), for: .normal)**

**self.setTitleColor(UIColor(red: 0.31, green: 0.31, blue: 0.31, alpha: 1), for: .selected)**

**case .none:**

**break**

**}**

**}**

**}**

**override func draw(\_ rect: CGRect) {**

**self.layer.cornerRadius = 10**

**self.layer.masksToBounds = true**

**self.titleLabel?.font = UIFont.systemFont(ofSize: 30)**

**}**

**}**