МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

|  |  |
| --- | --- |
| КУРСОВОЙ ПРОЕКТ  ЗАЩИЩЕН С ОЦЕНКОЙ |  |

РУКОВОДИТЕЛЬ:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| доцент |  |  |  |  |  | С. В. Щекин |
| (должность, уч. степень, звание) |  | (дата защиты) |  | (подпись) |  | (инициалы, фамилия) |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| «РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ» |
| ПО КУРСУ: «ПРОГРАМММИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ» |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТ: | 4831 |  | 12.11.2021 |  |  |  | К. А. Корнющенков |
|  | (группа) |  | (выполнено) |  | (подпись) |  | (инициалы, фамилия) |

Санкт-Петербург 2021

**СОДЕРЖАНИЕ**

[ЗАДАНИЕ 3](#_Toc87611986)

[ХОД РАБОТЫ 3](#_Toc87611987)

[РЕЗУЛЬТАТ РАБОТЫ 4](#_Toc87611988)

[ЗАКЛЮЧЕНИЕ 7](#_Toc87611989)

[В ходе выполнения курсового проекта было разработано мобильное приложение для мобильной операционной системы IOS. 7](#_Toc87611990)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 7](#_Toc87611991)

[ПРИЛОЖЕНИЕ А 8](#_Toc87611992)

ЗАДАНИЕ

Разработать мобильное приложение, основываясь на опыте, полученном при выполнении лабораторных работ. Тип приложения, выполняемая им задача, используемые технологии, средства разработки и целевая мобильная платформа согласовываются с преподавателем.

ХОД РАБОТЫ

Для реализации данного приложения был выбран следующий стек технологий:

Язык программирования Swift, для создания UI- компонентов библиотека данного языка UIKit, которая позволяет создавать императивный UI, а для работы с данными UserDefaults, что является контейнером для хранения значений по ключу. Конечный продукт будет представлен в виде мобильного приложения под платформу IOS.

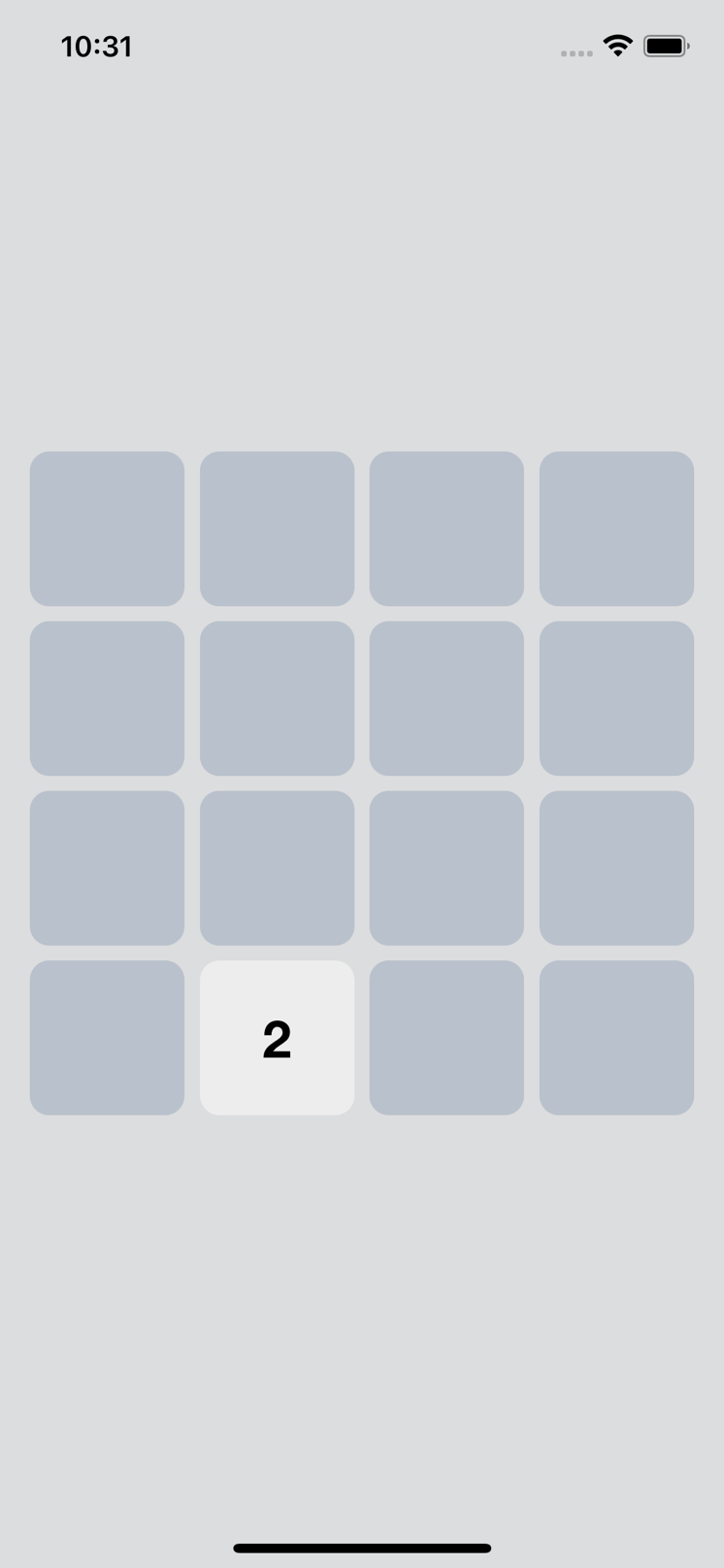
Swift — открытый мультипарадигмальный компилируемый язык программирования общего назначения. Создан компанией Apple в первую очередь для разработчиков iOS и macOS (в настоящий момент вышел за рамки этих ОС). ... Программы на Swiftкомпилируются при помощи LLVM, входящей в интегрированную среду разработки Xcode 6 и выше.

UIKit — это фреймворк, который используется при разработке приложений для iOS. Он определяет основные компоненты приложения iOS, от ярлыков и кнопок до табличных представлений и контроллеров навигации.

UserDefaults средство для хранения небольших данных нашего приложения. UserDefaults способен поддерживать все основные типы данных:Bool, Dictionary, Int, String, Data, and Array. Но не поддерживает опциональные типы.

Код программы приведен в приложении А

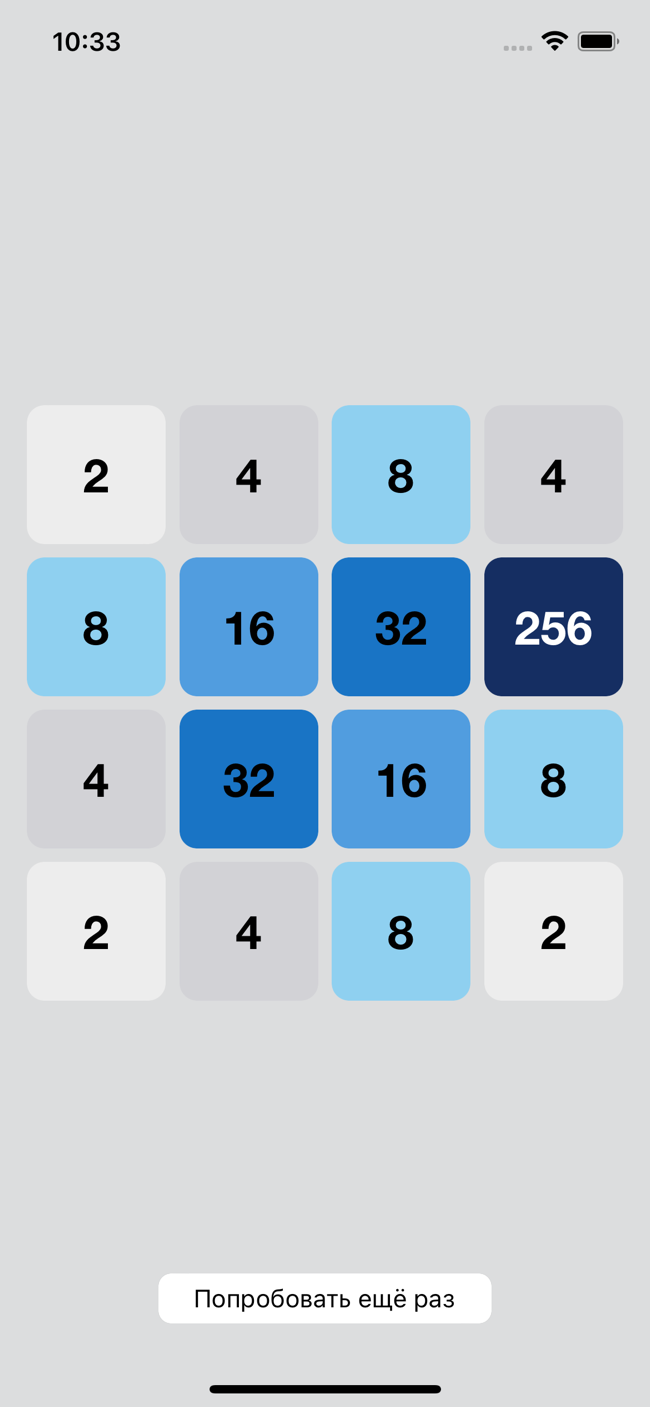
РЕЗУЛЬТАТ РАБОТЫ



Стартовый экран мобильного приложения новой игры



Экран во время игры



Экран при поражении в игре

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано мобильное приложение для мобильной операционной системы IOS. Применены практические навыки работы с базой данных, работа с жизненным циклом мобильного приложение, проектирование сложных UI компонентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ISO/IEC/IEEE 29148:2018 International Standard — Systems and software engineering — Life cycle processes — Requirements engineering
2. Виггерс, Карл. Разработка требований к программному обеспечению = Software Requirements: пер. с англ.; 3-е издание, дополненное / Карл Виггерс, Джой Битти — М.: Издательство «Русская редакция», 2014. — 736 с.: ил.
3. IEEE Recommended Practice for SRS / IEEE Std 830-1998 (Revision of IEEE Std 830-1993) [Электронный ресурс]. — University of Alaska Anchorage, 2020 — URL: <http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf>   
   (дата обращения: 10.04.2021)
4. Writing Software Requirements Specifications (SRS) [Электронный ресурс]. — INKtopia Limited, 2020 — URL: <https://techwhirl.com/writing-software-requirements-specifications/> (дата обращения: 10.04.2021)
5. Software Requirements Specification Helps to Protect IT Projects From Failure [Электронный ресурс]. — Belitsoft, 2004-2020 — URL: <https://belitsoft.com/php-development-services/software-requirements-specification-helps-protect-it-projects-failure> (дата обращения: 10.04.2021)

ПРИЛОЖЕНИЕ А

**class** Button: UIButton {

**override** **func** awakeFromNib() {

**super**.awakeFromNib()

view()

}

**func** view(){

**self**.backgroundColor = colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)

**self**.layer.cornerRadius = **self**.frame.height / 4

**self**.clipsToBounds = **true**

**self**.layer.shadowRadius = 4.0

**self**.layer.shadowOpacity = 0.6

**self**.layer.shadowOffset = CGSize.zero

**self**.layer.shadowColor = UIColor.black.cgColor

**self**.isHidden = **true**

}

}

//проверка крайних ячеек

**struct** checkExtremeStruct {

**let** main:Int

**let** support1:Int

**let** support2:Int

}

//проверка боковых ячеек

**struct** checkSideStruct {

**let** main:Int

**let** support1:Int

**let** support2:Int

**let** support3:Int

}

//проверка центральных ячеек

**struct** checkCenterStruct {

**let** main:Int

**let** support1:Int

**let** support2:Int

**let** support3:Int

**let** support4:Int

}

**struct** colorStatic {

**let** a2 = colorLiteral(red: 0.9293108582, green: 0.9294700623, blue: 0.9293007255, alpha: 1)

**let** a4 = colorLiteral(red: 0.823451221, green: 0.8234113455, blue: 0.8406392932, alpha: 1)

**let** a8 = colorLiteral(red: 0.5626729131, green: 0.8172994256, blue: 0.9425179362, alpha: 1)

**let** a16 = colorLiteral(red: 0.3183631301, green: 0.6156502962, blue: 0.8762559295, alpha: 1)

**let** a32 = colorLiteral(red: 0.09935102612, green: 0.4532182813, blue: 0.772266984, alpha: 1)

**let** a64 = colorLiteral(red: 0.005977380555, green: 0.3324437141, blue: 0.7106510401, alpha: 1)

**let** a128 = colorLiteral(red: 0.01671738736, green: 0.2286086977, blue: 0.5820831656, alpha: 1)

**let** a256 = colorLiteral(red: 0.08390396088, green: 0.1794761419, blue: 0.3860501349, alpha: 1)

**let** a512 = colorLiteral(red: 0.04141493887, green: 0.1374525428, blue: 0.4242787957, alpha: 1)

**let** a1024 = colorLiteral(red: 0.006856347434, green: 0.06721398979, blue: 0.507160008, alpha: 1)

**let** a2048 = colorLiteral(red: 0.04177325219, green: 0.02215959318, blue: 0.2993516028, alpha: 1)

**func** color(digite:Int)->UIColor{

**switch** digite {

**case** 2:

**return** a2

**case** 4:

**return** a4

**case** 8:

**return** a8

**case** 16:

**return** a16

**case** 32:

**return** a32

**case** 64:

**return** a64

**case** 128:

**return** a128

**case** 256:

**return** a256

**case** 512:

**return** a512

**case** 1024:

**return** a1024

**case** 2048:

**return** a2048

**default**:

**return** .clear

}

}

}

**class** GameCV: UICollectionView {

**var** data:[cellData] = []

**func** returnData(data:**@escaping**([cellData])->Void){

data(**self**.data)

}

}

//**MARK: Проверка**

**extension** GameCV{

//проверка на наличие пустых ячеек

**func** checkFinishGame()->Bool{

**for** i **in** data{

**if** !i.data{

**return** **true**

}

}

**return** **false**

}

//win

**func** checkWin() ->Bool{

**for** i **in** data{

**if** i.digite == 2048{

**return** **true**

}

}

**return** **false**

}

//проверка на наличие одинаковых соседей

**func** continion()->Bool{

**var** check = **false**

**let** checkExtremeArray:[checkExtremeStruct] = [checkExtremeStruct(main: 0, support1: 1, support2: 4),

checkExtremeStruct(main: 3, support1: 2, support2: 7),

checkExtremeStruct(main: 12, support1: 13, support2: 8),

checkExtremeStruct(main: 15, support1: 14, support2: 11)]

**for** i **in** 0...3{

**if** check{

**break**

}

check = checkExtreme(main: checkExtremeArray[i].main,

support1: checkExtremeArray[i].support1,

support2: checkExtremeArray[i].support2)

}

**let** checkSideArray:[checkSideStruct] = [checkSideStruct(main: 1, support1: 0, support2: 2, support3: 5),

checkSideStruct(main: 2, support1: 1, support2: 3, support3: 6),

checkSideStruct(main: 4, support1: 0, support2: 5, support3: 8),

checkSideStruct(main: 8, support1: 4, support2: 9, support3: 12),

checkSideStruct(main: 13, support1: 14, support2: 12, support3: 9),

checkSideStruct(main: 14, support1: 13, support2: 15, support3: 10),

checkSideStruct(main: 7, support1: 3, support2: 11, support3: 6),

checkSideStruct(main: 11, support1: 7, support2: 15, support3: 10)]

**for** i **in** 0...7{

**if** check{

**break**

}

check = checkSide(main: checkSideArray[i].main,

support1: checkSideArray[i].support1,

support2: checkSideArray[i].support2,

support3: checkSideArray[i].support3)

}

**let** checkCenterArray:[checkCenterStruct] = [checkCenterStruct(main: 5, support1: 1, support2: 4, support3: 6, support4: 9),

checkCenterStruct(main: 6, support1: 5, support2: 7, support3: 2, support4: 10),

checkCenterStruct(main: 9, support1: 5, support2: 8, support3: 10, support4: 13),

checkCenterStruct(main: 10, support1: 9, support2: 11, support3: 6, support4: 14)]

**for** i **in** 0...3{

**if** check{

**break**

}

check = checkCenter(main: checkCenterArray[i].main,

support1: checkCenterArray[i].support1,

support2: checkCenterArray[i].support2,

support3: checkCenterArray[i].support3,

support4: checkCenterArray[i].support4)

}

**return** check

}

//проверяем крайние числа на равенство с их соседями

**func** checkExtreme(main:Int,support1:Int,support2:Int) -> Bool{

**if** (data[main].digite == data[support1].digite || data[main].digite == data[support2].digite){

**return** **true**

}

**return** **false**

}

//проверяем боковые числа на равенство с их соседями

**func** checkSide(main:Int,support1:Int,support2:Int,support3:Int) -> Bool {

**if** data[main].digite == data[support1].digite || data[main].digite == data[support2].digite

|| data[main].digite == data[support3].digite {

**return** **true**

}

**return** **false**

}

//проверяем центральные числа на равенство с их соседями

**func** checkCenter(main:Int,support1:Int,support2:Int,support3:Int,support4:Int) -> Bool {

**if** data[main].digite == data[support1].digite || data[main].digite == data[support2].digite

|| data[main].digite == data[support3].digite || data[main].digite == data[support4].digite{

**return** **true**

}

**return** **false**

}

}

//**MARK: Ход**

**extension** GameCV{

**func** up()->Int{

**var** swipe = 0

swipe += UpOrLeftSwipe(minIndex: 0, maxIndex: 12, kef: 4)

swipe += leftAndUpSum(one: 0, two: 4, tree: 8, four: 12)

swipe += UpOrLeftSwipe(minIndex: 1, maxIndex: 13, kef: 4)

swipe += leftAndUpSum(one: 1, two: 5, tree: 9, four: 13)

swipe += UpOrLeftSwipe(minIndex: 2, maxIndex: 14, kef: 4)

swipe += leftAndUpSum(one: 2, two: 6, tree: 10, four: 14)

swipe += UpOrLeftSwipe(minIndex: 3, maxIndex: 15, kef: 4)

swipe += leftAndUpSum(one: 3, two: 7, tree: 11, four: 15)

**return** swipe

}

**func** down()->Int{

**var** swipe = 0

swipe += downOrRigthSwipe(minIndex: 0, maxIndex: 12, kef: 4)

swipe += rigthAndDownSum(one: 0, two: 4, tree: 8, four: 12)

swipe += downOrRigthSwipe(minIndex: 1, maxIndex: 13, kef: 4)

swipe += rigthAndDownSum(one: 1, two: 5, tree: 9, four: 13)

swipe += downOrRigthSwipe(minIndex: 2, maxIndex: 14, kef: 4)

swipe += rigthAndDownSum(one: 2, two: 6, tree: 10, four: 14)

swipe += downOrRigthSwipe(minIndex: 3, maxIndex: 15, kef:4)

swipe += rigthAndDownSum(one: 3, two: 7, tree: 11, four: 15)

**return** swipe

}

**func** left()->Int{

**var** swipe = 0

swipe += UpOrLeftSwipe(minIndex: 0, maxIndex: 3, kef: 1)

swipe += leftAndUpSum(one: 0, two: 1, tree: 2, four: 3)

swipe += UpOrLeftSwipe(minIndex: 4, maxIndex: 7, kef: 1)

swipe += leftAndUpSum(one: 4, two: 5, tree: 6, four: 7)

swipe += UpOrLeftSwipe(minIndex: 8, maxIndex: 11, kef: 1)

swipe += leftAndUpSum(one: 8, two: 9, tree: 10, four: 11)

swipe += UpOrLeftSwipe(minIndex: 12, maxIndex: 15, kef: 1)

swipe += leftAndUpSum(one: 12, two: 13, tree: 14, four: 15)

**return** swipe

}

**func** right()->Int{

**var** swipe = 0

swipe += downOrRigthSwipe(minIndex: 0, maxIndex: 3, kef: 1)

swipe += rigthAndDownSum(one: 0, two: 1, tree: 2, four: 3)

swipe += downOrRigthSwipe(minIndex: 4, maxIndex: 7, kef: 1)

swipe += rigthAndDownSum(one: 4, two: 5, tree: 6, four: 7)

swipe += downOrRigthSwipe(minIndex: 8, maxIndex: 11, kef: 1)

swipe += rigthAndDownSum(one: 8, two: 9, tree: 10, four: 11)

swipe += downOrRigthSwipe(minIndex: 12, maxIndex: 15, kef: 1)

swipe += rigthAndDownSum(one: 12, two: 13, tree: 14, four: 15)

**return** swipe

}

**func** leftAndUpSum(one:Int,two:Int,tree:Int,four:Int)->Int{

**var** swipe = 0

**if** data[one].digite == data[two].digite && data[one].digite != **nil** {

data[one] = cellData(digite: data[one].digite! \* 2)

data[two] = data[tree]

data[tree] = data[four]

data[four] = cellData()

swipe += 1

}

**if** data[tree].digite == data[two].digite && data[two].digite != **nil**{

data[two] = cellData(digite: data[two].digite! \* 2)

data[tree] = data[four]

data[four] = cellData()

swipe += 1

}

**if** data[tree].digite == data[four].digite && data[tree].digite != **nil**{

data[tree] = cellData(digite: data[tree].digite! \* 2)

data[four] = cellData()

swipe += 1

}

**return** swipe

}

**func** rigthAndDownSum(one:Int,two:Int,tree:Int,four:Int)->Int{

**var** swipe = 0

**if** data[four].digite == data[tree].digite && data[four].digite != **nil** {

data[four] = cellData(digite: data[four].digite! \* 2)

data[tree] = data[two]

data[two] = data[one]

data[one] = cellData()

swipe += 1

}

**if** data[tree].digite == data[two].digite && data[tree].digite != **nil**{

data[tree] = cellData(digite: data[tree].digite! \* 2)

data[two] = data[one]

data[one] = cellData()

swipe += 1

}

**if** data[two].digite == data[one].digite && data[two].digite != **nil**{

data[two] = cellData(digite: data[two].digite! \* 2)

data[one] = cellData()

swipe += 1

}

**return** swipe

}

**func** UpOrLeftSwipe(minIndex:Int,maxIndex:Int,kef:Int)->Int{

**var** swipeCount = 0

**var** value = minIndex

**var** count = 0

**var** hado = 0

**while** value < maxIndex + 1 {

**if** **self**.data[value].data{

**if** hado==1{

**self**.data[value-count] = **self**.data[value]

**self**.data[value] = cellData()

swipeCount += 1

}

}**else**{

hado = 1

count += kef

}

value += kef

}

**return** swipeCount

}

**func** downOrRigthSwipe(minIndex:Int,maxIndex:Int,kef:Int)->Int{

**var** swipeCount = 0

**var** value = maxIndex

**var** count = 0

**var** hado = 0

**while** value > minIndex - 1 {

**if** **self**.data[value].data{

**if** hado==1{

**self**.data[value+count] = **self**.data[value]

**self**.data[value] = cellData()

swipeCount += 1

}

}**else**{

hado = 1

count += kef

}

value -= kef

}

**return** swipeCount

}

}

//MARK:Вспомогающие функции

**extension** GameCV{

**func** startArray(){

**let** savedArray = UserDefaults.standard.object(forKey: "data") **as**? [Int?]

**if** savedArray?.count == 16 {

**for** i **in** savedArray!{

**if** i == -1{

data.append(cellData())

}**else**{

data.append(cellData(digite: i!))

}

}

}**else**{

**for** \_ **in** 1...16{

data.append(cellData())

}

genNewElement { \_ **in** }

}

}

**func** genNewElement(newIndex:**@escaping**((Int)->Void)){

**let** newElement = cellData(digite: 2)

**let** check = **true**

**var** index = arc4random() % 16

**while** check {

**if** !**self**.data[Int(index)].data{

data[Int(index)] = newElement

newIndex(Int(index))

**break**

}

index = arc4random() % 16

}

}

}

**class** Cell: UICollectionViewCell {

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

**override** **func** awakeFromNib() {

**super**.awakeFromNib()

label.textAlignment = .center

label.sizeToFit()

}

**override** **func** layoutSubviews() {

**super**.layoutSubviews()

label.frame = CGRect(x: 0, y: 0, width: frame.width, height: frame.height)

}

**func** cellSetting(cell:UICollectionViewCell){

cell.layer.cornerRadius = cell.frame.height / 8

cell.clipsToBounds = **true**

cell.backgroundColor = .white

}

**func** cellContent(cell:Cell,data:cellData){

**if** data.digite != **nil**{

cell.backgroundColor = data.color

cell.label.text = String(data.digite!)

cell.label.textColor = data.digiteColor

}**else**{

cell.backgroundColor = data.color

cell.label.text = ""

}

}

**func** animateNewElement(){

**let** bounds = **self**.bounds

**self**.bounds = CGRect(x: **self**.center.x, y: **self**.center.y, width: 0, height: 0)

**let** text = label.text

label.text = ""

UICollectionViewCell.animate(withDuration: 0.25, delay: 0, options: .curveEaseInOut, animations: {

**self**.bounds = bounds

}, completion: { [**weak** **self**] \_ **in**

**self**?.label.text = text

})

}

}

**struct** cellData {

**let** color: UIColor

**let** digite: Int?

**let** digiteColor: UIColor?

**let** data:Bool

**init**(digite:Int){

**self**.color = colorStatic().color(digite: digite)

**self**.digite = digite

digite > 32 ? (**self**.digiteColor = **#colorLiteral**(red: 1.0, green: 1.0, blue: 1.0, alpha: 1.0)) : (**self**.digiteColor = **#colorLiteral**(red: 0, green: 0, blue: 0, alpha: 1))

**self**.data = **true**

}

**init**() {

**self**.color = colorLiteral(red: 0.7271783352, green: 0.7574660778, blue: 0.800142169, alpha: 1)

**self**.digite = **nil**

**self**.digiteColor = **nil**

**self**.data = **false**

}

}

**class** ViewController: UIViewController {

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

**@IBOutlet** **weak** **var** newGameButton: Button!

**var** animateIndex:Int = -1

//**MARK: viewDidLoad**

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

collectionViewFuncForViewDidLoad()

NotificationCenter.default.addObserver(**self**,selector: **#selector**(sceneWillResignActiveNotification(\_:)),name: UIApplication.willResignActiveNotification,object: **nil**)

}

//**MARK: newGame**

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

newGame()

}

}

//**MARK: CollectionView**

**extension** ViewController: UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout{

**func** collectionViewFuncForViewDidLoad(){

collectionView.startArray()

collectionViewProtocol(collectionView: collectionView)

swipeAdd()

}

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

**return** **self**.collectionView.data.count

}

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

**let** cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cell", for: indexPath) **as**! Cell

cell.cellSetting(cell: cell)

cell.cellContent(cell: cell, data: **self**.collectionView.data[indexPath.row])

**if** indexPath.row == animateIndex{

cell.animateNewElement()

}

**return** cell

}

**func** collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize{

**let** size = (collectionView.frame.width - 25) / 4

**return** CGSize(width: size, height: size)

}

**func** swipeAdd(){

**let** left = UISwipeGestureRecognizer(target: **self**, action: **#selector**(handleSwipes(\_:)))

left.direction = .left

**let** right = UISwipeGestureRecognizer(target: **self**, action: **#selector**(handleSwipes(\_:)))

right.direction = .right

**let** up = UISwipeGestureRecognizer(target: **self**, action: **#selector**(handleSwipes(\_:)))

up.direction = .up

**let** down = UISwipeGestureRecognizer(target: **self**, action: **#selector**(handleSwipes(\_:)))

down.direction = .down

collectionView.addGestureRecognizer(left)

collectionView.addGestureRecognizer(right)

collectionView.addGestureRecognizer(up)

collectionView.addGestureRecognizer(down)

}

//все действия

**@objc** **func** handleSwipes(\_ sender:UISwipeGestureRecognizer){

**var** swipeCount = 0

**switch** sender.direction {

**case** .right:

swipeCount = collectionView.right()

**case** .left:

swipeCount = collectionView.left()

**case** .down:

swipeCount = collectionView.down()

**case** .up:

swipeCount = collectionView.up()

**default**:

print("error swipe")

}

**if** collectionView.checkWin(){

alertWin()

}**else**{

**if** collectionView.checkFinishGame() && swipeCount>0{

collectionView.genNewElement { (index) **in**

**self**.animateIndex = index

}

}

**if** !collectionView.checkFinishGame() && !collectionView.continion(){

newGameButton.isHidden = **false**

}

}

**if** swipeCount>0{

collectionView.reloadData()

}

}

//CollectionView portocol

**func** collectionViewProtocol(collectionView:UICollectionView){

collectionView.delegate = **self**

collectionView.dataSource = **self**

}

}

//**MARK: element satting**

**extension** ViewController{

//win alert

**func** alertWin(){

**let** alert = UIAlertController(title: "You WIN", message: **nil**, preferredStyle: .alert)

**let** action = UIAlertAction(title: "OK", style: .default) { [**weak** **self**](action) **in**

**self**?.newGame()

}

alert.addAction(action)

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

}

//new game

**func** newGame(){

collectionView.data.removeAll()

newGameButton.isHidden = **true**

**for** \_ **in** 1...16{

collectionView.data.append(cellData())

}

collectionView.genNewElement { \_ **in** }

collectionView.reloadData()

}

**@objc** **func** sceneWillResignActiveNotification(\_ notification: NSNotification) {

collectionView.returnData { (data) **in**

**var** dataDigite:[Int] = []

**for** i **in** data{

**if** i.digite==**nil**{

dataDigite.append(-1)

}**else**{

dataDigite.append(i.digite!)

}

}

**let** defaults = UserDefaults.standard

defaults.set(dataDigite, forKey: "data")

}

}

}