山东理工大学计算机学院

课　程　设　计

（数据结构）

班　　级	信应1301
姓　　名	张睿提
学　　号	13310542008
指导教师	石少俭

二〇一五年一月二十日

课程设计任务书及成绩评定

课题名称 通讯录的制作

Ⅰ、题目的目的和要求：

1、设计目的

巩固和加深对数据结构的理解，通过上机实验、调试程序，加深对课本知识的理解，最终使学生能够熟练应用数据结构的知识写程序。

（1）通过本课程的学习，能熟练掌握几种基本数据结构的基本操作。

（2）能针对给定题目，选择相应的数据结构，分析并设计算法，进而给出问题的正确求解过程并编写代码实现。

2、设计题目要求：

设计目的:用〈〈数据结构〉〉中的双向链表作数据结构,结合 C/C++语言基 知识。编写一个通讯录管理系统。以把所学数据结构知识应用到实际软件开发中去。

设计内容： 系统应完成一下几方面的功能：输入信息——enter(); 显示信息———display( ); 查找以姓名作为关键字 ———search( ); 删除信息———delete( );

存盘———save ( );

装入———load( ) ; 设计要求:

1) 每条信息至包含 :姓名(NAME)街道(STREET)城市(CITY)邮编(EIP) 国家(STATE)几项

2) 作为一个完整的系统,应具有友好的界面和较强的容错能力

3) 上机能正常运行,并写出课程设计报告

Ⅱ、设计进度及完成情况

日 期	内 容
1.10-1.11	选取参考书，查阅有关文献资料，完成资料搜集和系统分析工作。
1.12~1.14	创建相关数据结构,录入源程序。
1.17~1.19	调试程序并记录调试中的问题，初步完成课程设计报告。
1.20~1.21	上交课程设计报告打印版并进行课程设计答辩，要求每个同学针对自己的设计回答指导教师 3-4 个问题。
	考核结束后将课程设计报告和源程序的电子版交班长统一刻光盘上交。

Ⅲ、主要参考文献及资料

[1] 严蔚敏 数据结构（C 语言版）清华大学出版社 1999

[2] 严蔚敏 数据结构题集（C 语言版）清华大学出版社 1999

[3] 谭浩强 C 语言程序设计 清华大学出版社

[4] 与所用编程环境相配套的 C 语言或 C++相关的资料

[5] Jack Nutting 等 精通 iOS 开发（第 6 版）Apress

Ⅳ、成绩评定：

设计成绩: _____

（教师填写）

指导老师: _____

（签字）

二〇一五 年 一 月 二 十一 日

目　录

第一章 概述

课程设计是实践性教学中的一个重要环节，它以某一课程为基础，可以涉及和课程相关的各个方面，是一门独立于课程之外的特殊课程。课程设计是让同学们对所学的课程更全面的学习和应用，理解和掌握课程的相关知识。《数据结构》是一门重要的专业基础课，是计算机理论和应用的核心基础课程。

数据结构课程设计，要求学生在数据结构的逻辑特性和物理表示、数据结构的选择和应用、算法的设计及其实现等方面，加深对课程基本内容的理解。同时，在程序设计方法以及上机操作等基本技能和科学作风方面受到比较系统和严格的训练。

在这次的课程设计中我选择的题目是通讯录的制作。传统的通讯录，填写费时费力，不易携带，易丢失，不方便查找。但是现在通讯录借助计算机系统完成后，填写效率可以得到提高，也可以减少出错的几率，查找更加方便，不易丢失，不需要维护，可添加照片。可以使通讯录的使用更加的方便，迅捷，减少很多劳动量。

第二章 系统分析

1. 通讯录的功能包括：对一个人的姓名、地址、电话等信息的采集。由于上述三项基本信息都是通过字符（即关键字）进行的，所以要对姓名建立索引，以获的高效率。故重点是要完成数据的建立、查找、插入、删除等基本操作。
2. 既为通讯录管理，就需要一个模块来完成对被记录信息的注册、登记和删除，本程序使用 viewDidLoad 完成上述操作。
3. 演示程序是以用户于 iOS 设备的对话方式执行，这需要一个模块来完成使用者与计算机语言是转化。
4. 程序执行时的命令：

   本程序为了使用时的方便，采用菜单式的方式来完成程序的演示，只需按提示输入即可。

5.测试数据。

第三章 概要设计

1、数据结构的设计

A,添加:

系统将提示用户输入新添加人员信息,输入到文件中,人员信息数据包括姓名（name）,街道（street）,城市（city），邮编（eip）,国家（state）.

B,删除:

首先由用户输入要删除的人员的姓名,然后调用删除函数,删除该人员的所有相关资料.

C,显示所有人员信息:

该功能将显示已经保存的所有人员的姓名,街道,城市,邮编及国家.

2、算法的设计

创建数据类型

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
@interface Person : NSManagedObject
@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSString * tel;
@property (nonatomic, retain) NSData * imageData;
@property (nonatomic, retain) NSString * firstN;
@property (nonatomic, retain) NSString * street;
@property (nonatomic, retain) NSString * city;
@property (nonatomic, retain) NSString * eip;
@property (nonatomic, retain) NSString * state;

@end
```

调出数据

```
#import "EditViewController.h"
#import "Pinyin.h"
@interface EditViewController ()
//显示用户信息
@property (strong, nonatomic) IBOutlet UITextField *nameTextField;
@property (strong, nonatomic) IBOutlet UITextField *telTextField;

@end
```

第四章 详细设计

1. 存入数据

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
@interface Person : NSManagedObject
@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSString * tel;
@property (nonatomic, retain) NSData * imageData;
@property (nonatomic, retain) NSString * firstN;
@property (nonatomic, retain) NSString * street;
@property (nonatomic, retain) NSString * city;
@property (nonatomic, retain) NSString * eip;
@property (nonatomic, retain) NSString * state;

@end
```

2. 显示数据

```
#import "EditViewController.h"
#import "Pinyin.h"

@interface EditViewController ()
//显示用户信息
@property (strong, nonatomic) IBOutlet UITextField *nameTextField;
@property (strong, nonatomic) IBOutlet UITextField *telTextField;

//显示用户图片
@property (strong, nonatomic) IBOutlet UIButton *imageButton;

//声明 ImagePicker
@property (strong, nonatomic) UIImagePickerController *picker;

//声明上下文
@property (strong, nonatomic) NSManagedObjectContext * managedObjectContext;

@end
```

```
@implementation EditViewController

//点击图片按钮设置图片
- (IBAction)tapImageButton:(id)sender {

    //跳转到 ImagePickerView 来获取按钮
    [self presentViewController:self.picker animated:YES completion:^{}];

}

//回调图片选择取消
-(void)imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
    //在 ImagePickerView 中点击取消时回到原来的界面
    [self dismissViewControllerAnimated:YES completion:^{}];
}

//实现图片回调方法，从相册获取图片
-(void) imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{

    //获取到编辑好的图片
    UIImage * image = info[UIImagePickerControllerEditedImage];

    //把获取的图片设置成用户的头像
    [self.imageButton setImage:image forState:UIControlStateNormal];

    //返回到原来 View
    [self dismissViewControllerAnimated:YES completion:^{}];

}

- (IBAction)tapSave:(id)sender
{

    if (![self.nameTextField.text isEqualToString:@""] && ![self.telTextField.text
```

```
isEqualToString:@""])
    {

    //如果 person 为空则新建，如果已经存在则更新
    if (self.person == nil)
    {
        self.person = [NSEntityDescription
insertNewObjectForEntityForName:NSStringFromClass([Person class])
inManagedObjectContext:self.managedObjectContext];
    }
    //赋值
    self.person.name = self.nameTextField.text;
    self.person.tel = self.telTextField.text;
    self.person.firstN = [NSString stringWithFormat:@"%c",
pinyinFirstLetter([self.person.name characterAtIndex:0])-32];

    //把 button 上的图片存入对象
    UIImage *buttonImage = [self.imageButton imageView].image;
    self.person.imageData = UIImagePNGRepresentation(buttonImage);

        //保存
        NSError *error;
        if (![self.managedObjectContext save:&error]) {
            NSLog(@"%@", [error localizedDescription]);
        }

        //保存成功后 POP 到表视图
        [self.navigationController popToRootViewControllerAnimated:YES];


    }

}

- (void)viewDidLoad
{
    [super viewDidLoad];
```

```
        self.nameTextField.text = self.person.name;
        self.telTextField.text = self.person.tel;

        if (self.person.imageData != nil)
        {
            UIImage *image = [UIImage imageWithData:self.person.imageData];
            [self.imageButton setImage:image forState:UIControlStateNormal];

        }


    //获取上下文
    UIApplication *application = [UIApplication sharedApplication];
    id delegate = application.delegate;
    self.managedObjectContext = [delegate managedObjectContext];


    //处理键盘
    self.nameTextField.delegate = self;
    self.telTextField.delegate = self;



    //初始化并配置 ImagePicker
    self.picker = [[UIImagePickerController alloc] init];
    //picker 是否可以编辑
    self.picker.allowsEditing = YES;
    //注册回调
    self.picker.delegate = self;

}
-(BOOL) textFieldShouldReturn:(UITextField *)textField
{
    [self.nameTextField resignFirstResponder];
    [self.telTextField resignFirstResponder];
    return YES;
}

- (void)didReceiveMemoryWarning
```

```
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
}
*/
@end
```

3. 搜索

```
#import "SearchCell.h"

@implementation SearchCell

- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)reuseIdentifier
{
    self = [super initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:reuseIdentifier];
    if (self) {
        // Initialization code

    }
    return self;
}

- (void)awakeFromNib
{
    // Initialization code
}
```

```
- (void)setSelected:(BOOL)selected animated:(BOOL)animated
{
    [super setSelected:selected animated:animated];

    // Configure the view for the selected state
}

@end
```

4. 通讯录索引

```
//
//  RootTableViewController.m
//  TellBook
//


#import "RootTableViewController.h"
#import "Person.h"
#import "SearchCell.h"

@interface RootTableViewController ()

//声明上下文和 fetchedResultController
@property (nonatomic, strong) NSManagedObjectContext *
managedObjectContext;
@property (nonatomic, strong) NSFetchedResultsController *
fetchedResultsController;

//添加 Search Display Controller 属性
@property (strong, nonatomic) IBOutlet UISearchDisplayController
*displayC;




@end

@implementation RootTableViewController


- (void)viewDidLoad
{
```

```
    [super viewDidLoad];

    //获取上下文
    UIApplication *application = [UIApplication sharedApplication];
    id delegate = application.delegate;
    self.managedObjectContext = [delegate managedObjectContext];

    //创建请求对象
    NSFetchRequest *request = [[NSFetchRequest alloc]
initWithEntityName:NSStringFromClass([Person class])];
    //创建排序规则
    NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"firstN" ascending:YES];
    [request setSortDescriptors:@[sortDescriptor]];

    //执行查询
    self.fetchedResultsController = [[NSFetchedResultsController alloc]
initWithFetchRequest:request managedObjectContext:self.managedObjectContext
sectionNameKeyPath:@"firstN" cacheName:nil];

    self.fetchedResultsController.delegate = self;

    NSError *error;
    if (![self.fetchedResultsController performFetch:&error]) {
        NSLog(@"%@", [error localizedDescription]);
    }


    //注册我们自定义的 Cell
    [self.displayC.searchResultsTableView registerClass:[SearchCell
class] forCellReuseIdentifier:@"SearchCell"];

    self.tableView.delegate = self;
    self.tableView.dataSource = self;

}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
//当 search 中的文本变化时就执行下面的方法
- (void)searchBar:(UISearchBar *)searchBar textDidChange:(NSString
*)searchText
{
    //新建查询语句
    NSFetchRequest * request = [[NSFetchRequest
alloc]initWithEntityName:NSStringFromClass([Person class])];

    //排序规则
    NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"firstN" ascending:YES];
    [request setSortDescriptors:@[sortDescriptor]];

    //添加谓词
    NSPredicate * predicate = [NSPredicate
predicateWithFormat:@"name contains %@",searchText];
    [request setPredicate:predicate];

    //把查询结果存入 fetchedResultsController 中
    self.fetchedResultsController = [[NSFetchedResultsController alloc]
initWithFetchRequest:request managedObjectContext:self.managedObjectContext
sectionNameKeyPath:@"firstN" cacheName:nil];

    NSError *error;
    if (![self.fetchedResultsController performFetch:&error]) {
        NSLog(@"%@", [error localizedDescription]);
    }

    self.tableView.delegate = nil;
    self.tableView.dataSource = nil;
}

//当在 searchView 中点击取消按钮时我们重新刷新一下通讯录
-(void)searchBarCancelButtonClicked:(UISearchBar *)searchBar
{
   [self viewDidLoad];
}

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{

    NSArray *sections = [self.fetchedResultsController sections];
    return sections.count;
```

```
}

- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{

    NSArray *sections = [self.fetchedResultsController sections];
    id<NSFetchedResultsSectionInfo> sectionInfo = sections[section];
    return [sectionInfo numberOfObjects];


}

-(NSString *)tableView:(UITableView *)tableView
titleForHeaderInSection:(NSInteger)section
{
    NSArray *sections = [self.fetchedResultsController sections];
    id<NSFetchedResultsSectionInfo> sectionInfo = sections[section];
    return [sectionInfo name];

}


- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    Person *person = [self.fetchedResultsController
objectAtIndexPath:indexPath];
    UITableViewCell *cell;


    //根据不同的 tableView 来设置不同的 cell 模板
    if ([tableView isEqual:self.tableView])
    {

        cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"
forIndexPath:indexPath];

    }
    else
    {
        cell = [tableView
dequeueReusableCellWithIdentifier:@"SearchCell" forIndexPath:indexPath];
```

```
            }

            if (person.imageData != nil) {
                UIImage *image = [UIImage
imageWithData:person.imageData];
                cell.imageView.image = image;
            }

            cell.textLabel.text = person.name;
            cell.detailTextLabel.text = person.tel;

            return cell;


        }



        // Override to support conditional editing of the table view.
        - (BOOL)tableView:(UITableView *)tableView
canEditRowAtIndexPath:(NSIndexPath *)indexPath
        {
            if ([tableView isEqual:self.displayC.searchResultsTableView])
            {
                return NO;
            }
            // Return NO if you do not want the specified item to be editable.
            return YES;
        }


        -(NSString *) tableView:(UITableView *)tableView
titleForDeleteConfirmationButtonForRowAtIndexPath:(NSIndexPath *)indexPath
        {
            return @"删除";
        }

        // Override to support editing the table view.
        - (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath
        {
            if (editingStyle == UITableViewCellEditingStyleDelete) {
                Person *person = [self.fetchedResultsController
objectAtIndexPath:indexPath];
```

```
                [self.managedObjectContext deleteObject:person];
                NSError *error;
                if (![self.managedObjectContext save:&error]) {
                    NSLog(@"%@", [error localizedDescription]);
                }
            }
        }


        //给我们的通讯录加上索引，下面的方法返回的时一个数组
        -(NSArray *) sectionIndexTitlesForTableView:(UITableView *)tableView
        {
            //通过 fetchedResultsController 来获取 section 数组
            NSArray *sectionArray = [self.fetchedResultsController sections];

            //新建可变数组来返回索引数组，大小为 sectionArray 中元素的多
少
            NSMutableArray *index = [NSMutableArray
arrayWithCapacity:sectionArray.count];

            //通过循环获取每个 section 的 header,存入 addObject 中
            for (int i = 0; i < sectionArray.count; i ++)
            {
                id <NSFetchedResultsSectionInfo> info = sectionArray[i];
                [index addObject:[info name]];
            }

            //返回索引数组
            return index;


        }

        -(void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
        {
            if ([tableView isEqual:self.displayC.searchResultsTableView])
            {
                Person *person = [self.fetchedResultsController
objectAtIndexPath:indexPath];
                UIStoryboard * s = [UIStoryboard
storyboardWithName:@"Main" bundle:[NSBundle mainBundle]];

                //获取要目标视图
                UIViewController *destination = [s
```

```
instantiateViewControllerWithIdentifier:@"EditViewController"];

                    //键值编码传值
                    [destination setValue:person forKeyPath:@"person"];

                    [self.navigationController pushViewController:destination
animated:YES];
                }
            }

            /*
            // Override to support rearranging the table view.
            - (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:(NSIndexPath
*)toIndexPath
            {
            }
            */

            /*
            // Override to support conditional rearranging of the table view.
            - (BOOL)tableView:(UITableView *)tableView
canMoveRowAtIndexPath:(NSIndexPath *)indexPath
            {
                // Return NO if you do not want the item to be re-orderable.
                return YES;
            }
            */


            #pragma mark - Navigation

            // In a storyboard-based application, you will often want to do a little
preparation before navigation
            - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
            {
               //Navigation 切换时调用该方法，把值传递给下一个 Push 的页面，


                //在本 viewController 中 sender 会有两种情况，因为我们点击 cell
和 itemAdd 都可以跳转到下一个界面，所以得判断一下
                if ([sender isKindOfClass:([UITableViewCell class])])
                {
                    UITableViewCell *cell = sender;
                    //通过 cell 获取索引
```

```
                NSIndexPath *indexPath = [self.tableView
indexPathForCell:cell];

                //获取实体对象
                Person *person = [self.fetchedResultsController
objectAtIndexPath:indexPath];

                //获取要目标视图
                UIViewController *destination = [segue
destinationViewController];

                //键值编码传值
                [destination setValue:person forKeyPath:@"person"];
            }
        }
```

```
        /*
         Assume self has a property 'tableView' -- as is the case for an instance of
a UITableViewController
         subclass -- and a method configureCell:atIndexPath: which updates the
contents of a given cell
         with information from a managed object at the given index path in the
fetched results controller.
         */

        - (void)controllerWillChangeContent:(NSFetchedResultsController
*)controller {
            [self.tableView beginUpdates];
        }


        - (void)controller:(NSFetchedResultsController *)controller
didChangeSection:(id <NSFetchedResultsSectionInfo>)sectionInfo
                   atIndex:(NSUInteger)sectionIndex
forChangeType:(NSFetchedResultsChangeType)type {
```

```
                switch(type) {
                    case NSFetchedResultsChangeInsert:
                        [self.tableView insertSections:[NSIndexSet
indexSetWithIndex:sectionIndex]

withRowAnimation:UITableViewRowAnimationFade];
                        break;

                    case NSFetchedResultsChangeDelete:
                        [self.tableView deleteSections:[NSIndexSet
indexSetWithIndex:sectionIndex]

withRowAnimation:UITableViewRowAnimationFade];
                        break;
                }
            }


            - (void)controller:(NSFetchedResultsController *)controller
didChangeObject:(id)anObject
                atIndexPath:(NSIndexPath *)indexPath
forChangeType:(NSFetchedResultsChangeType)type
                newIndexPath:(NSIndexPath *)newIndexPath {

                UITableView *tableView = self.tableView;

                switch(type) {

                    case NSFetchedResultsChangeInsert:
                        [tableView insertRowsAtIndexPaths:[NSArray
arrayWithObject:newIndexPath]

withRowAnimation:UITableViewRowAnimationFade];
                        break;

                    case NSFetchedResultsChangeDelete:
                        [tableView deleteRowsAtIndexPaths:[NSArray
arrayWithObject:indexPath]

withRowAnimation:UITableViewRowAnimationFade];
                        break;

                    case NSFetchedResultsChangeUpdate:
                        [tableView reloadRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationFade];
```

```
                    break;

                case NSFetchedResultsChangeMove:
                    [tableView deleteRowsAtIndexPaths:[NSArray
arrayWithObject:indexPath]

withRowAnimation:UITableViewRowAnimationFade];
                    [tableView insertRowsAtIndexPaths:[NSArray
arrayWithObject:newIndexPath]

withRowAnimation:UITableViewRowAnimationFade];
                    break;
            }
        }


        - (void)controllerDidChangeContent:(NSFetchedResultsController
*)controller {
            [self.tableView endUpdates];
        }

        @end
```

第五章 运行与测试

1、引用了第三方开源库

```
/*
 *    pinyin.h
 *    Chinese Pinyin First Letter
 *
 *    Created by George on 4/21/10.
 *    Copyright 2010 RED/SAFI. All rights reserved.
 *
 */

/*
 * // Example
 *
 * #import "pinyin.h"
 *
 * NSString *hanyu = @"中国共产党万岁！ ";
 * for (int i = 0; i < [hanyu length]; i++)
 * {
 *          printf("%c", pinyinFirstLetter([hanyu characterAtIndex:i]));
 * }
 *
 */

char pinyinFirstLetter(unsigned short hanzi);

//
//    AppDelegate.m
//    TellBook
//



#import "AppDelegate.h"
```

```
@implementation AppDelegate

@synthesize managedObjectContext = _managedObjectContext;
@synthesize managedObjectModel = _managedObjectModel;
@synthesize persistentStoreCoordinator = _persistentStoreCoordinator;

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    // Called as part of the transition from the background to the inactive state; here you can
```

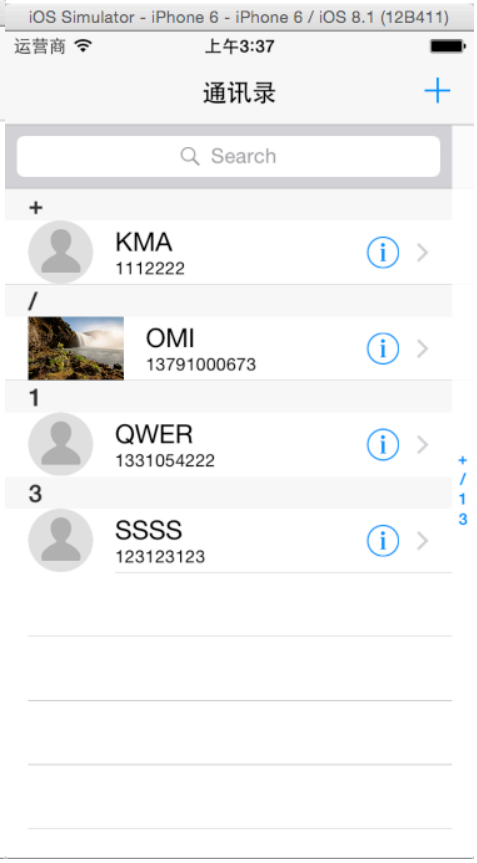
```
undo many of the changes made on entering the background.
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    // Restart any tasks that were paused (or not yet started) while the application was inactive.
If the application was previously in the background, optionally refresh the user interface.
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    // Saves changes in the application's managed object context before the application
terminates.
    [self saveContext];
}

- (void)saveContext
{
    NSError *error = nil;
    NSManagedObjectContext *managedObjectContext = self.managedObjectContext;
    if (managedObjectContext != nil) {
        if ([managedObjectContext hasChanges] && ![managedObjectContext save:&error])
{
            // Replace this implementation with code to handle the error appropriately.
            // abort() causes the application to generate a crash log and terminate. You
should not use this function in a shipping application, although it may be useful during
development.
            NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
            abort();
        }
    }
}

#pragma mark - Core Data stack
```

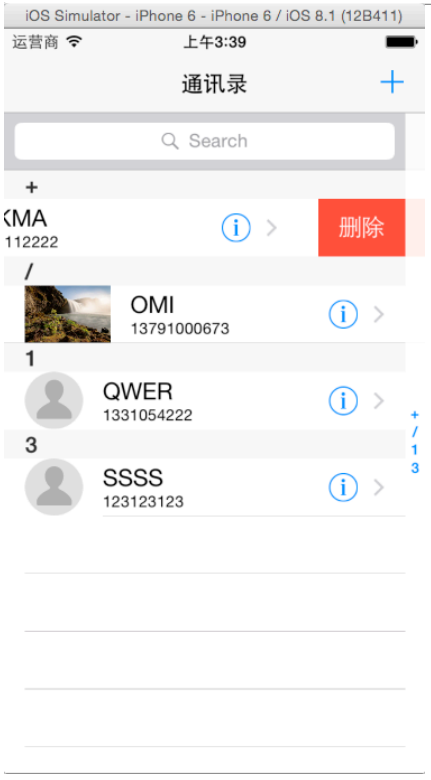
```
// Returns the managed object context for the application.
// If the context doesn't already exist, it is created and bound to the persistent store coordinator for the application.
- (NSManagedObjectContext *)managedObjectContext
{
    if (_managedObjectContext != nil) {
        return _managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
    if (coordinator != nil) {
        _managedObjectContext = [[NSManagedObjectContext alloc] init];
        [_managedObjectContext setPersistentStoreCoordinator:coordinator];
    }
    return _managedObjectContext;
}

// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectModel *)managedObjectModel
{
    if (_managedObjectModel != nil) {
        return _managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"Test090101" withExtension:@"momd"];
    _managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
    return _managedObjectModel;
}

// Returns the persistent store coordinator for the application.
// If the coordinator doesn't already exist, it is created and the application's store added to it.
```

```
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
    if (_persistentStoreCoordinator != nil) {
        return _persistentStoreCoordinator;
    }

    NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@"Test090101.sqlite"];

    NSError *error = nil;
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
initWithManagedObjectModel:[self managedObjectModel]];
    if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
configuration:nil URL:storeURL options:nil error:&error]) {
        /*
         Replace this implementation with code to handle the error appropriately.

         abort() causes the application to generate a crash log and terminate. You should not
use this function in a shipping application, although it may be useful during development.

         Typical reasons for an error here include:
         * The persistent store is not accessible;
         * The schema for the persistent store is incompatible with current managed object
model.
         Check the error message to determine what the actual problem was.


         If the persistent store is not accessible, there is typically something wrong with the
file path. Often, a file URL is pointing into the application's resources directory instead of a
writeable directory.

         If you encounter schema incompatibility errors during development, you can reduce
their frequency by:
         * Simply deleting the existing store:
```

```
            [[NSFileManager defaultManager] removeItemAtURL:storeURL error:nil]

            * Performing automatic lightweight migration by passing the following dictionary as the options parameter:
            @{NSMigratePersistentStoresAutomaticallyOption:@YES, NSInferMappingModelAutomaticallyOption:@YES}

            Lightweight migration will only work for a limited set of schema changes; consult "Core Data Model Versioning and Data Migration Programming Guide" for details.

            */
            NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
            abort();
        }

        return _persistentStoreCoordinator;
}

#pragma mark - Application's Documents directory

// Returns the URL to the application's Documents directory.
- (NSURL *)applicationDocumentsDirectory
{
        return [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory inDomains:NSUserDomainMask] lastObject];
}

@end
```

2、添加了许多不同人的数据验证，并删除





3、bug　修正，偶见输入问题，借用第三方库修复

第六章 总结与心得

此次课程设计是设计 iOS 通讯录。通过这次设计我总结了以下几点：

(1) 做课程设计有一定的规划步骤。确定课题->查阅参考资料->创建数据编写程序->调试程序->正确运行->总结报告。

(2) 任何一个程序设计都有一个算法思想，都应依附于某个核心算法进行设计，所以要牵涉到数据结构的设计，程序算法的设计，抽象数据类型的设计。

(3) 设计程序，尤其是比较大型的程序设计，都要确定模块的数量。明确每个模块的应该完成的具体任务，并要设计实现各个模块的接口。

(4) 程序设计先要确定主题，然后有一个大概的设计思路，再逐渐明确。是一个从模糊到明晰的过程。所设计的程序要逐渐完善，尽量将瑕疵降到最低，所以再不断的修改代码的时候要有恒心，有毅力。

参考文献:

[1] 严蔚敏、吴伟民主编　《数据结构》（C 语言版）　清华大学出版社　2002

[2] 殷人昆等著　《数据结构》（C++版）　清华大学出版社　2001

[3] 金远平著　《数据结构》（C++描述）　清华大学出版社　2005

[4] 许卓群等著　《数据结构与算法》　高等教育出版社　2004

[5] Frank M.Carrano 等著　《数据结构与C++高级教程》清华大学出版社　2004

[6] 严蔚敏、吴伟民　《数据结构习题集》（C 语言版）清华大学出版社

[7] Jack Nutting 等著　《精通 iOS 开发（第 6 版）》 Apress　2012