作业1:（重点：闭包）

1. 找出从1-10000的所有质数，将找出的质数存入可变数组，对数组进行正反排序并输出结果(采用5种以上的方法实现)；
2. 掌握版本控制git的基本使用
   1. git init 初始化版本库
   2. git add . 添加当前目录下的所有文件到暂存区
   3. git commit -m 提交暂存区的文件到版本库

var primes = [Int]()

for i in 2...100 {

var isPrime = true

for j in 2..<i{

if i % j == 0{

isPrime = false

}

}

if isPrime {

primes.append(i)

}

}

//再建一个保留正序

var primes1 = [Int]()

primes1 = primes

/\*\*for item in primes1 {

print(item)

}\*/

//使顺序变成倒序的

//方法一

/\*\*

func compare(n1:Int,n2:Int)->Bool{

return n1>n2

}

primes1.sort(by:compare(n1:n2:))\*/

//方法二

/\*\*

func compare(n1:Int,n2:Int)->Bool{

return n1>n2

}

primes1.sort(by:compare)\*/

//方法三

/\*\*

primes1.sort(by:{

(one:Int,two:Int) -> Bool in

return one>two

})\*/

//方法四

/\*\*

primes1.sort(by:{

(one,two) in return one>two

})\*/

//方法五

/\*\*

primes1.sort(by:{

(one,two) in one>two

})\*/

//方法六

/\*\*

primes1.sort(by:{$0>$1})\*/

//方法七

primes1.sort(by:>)

print(primes1)

作业2:（枚举、类、派生）

1. 实现Person类：
   1. 要求具有firstName, lastName，age，gender等存储属性,fullName计算属性；其中gender是枚举类型（male，female）；
   2. 具有指定构造函数和便利构造函数；
   3. 两个Person实例对象可以用==和!=进行比较；
   4. Person实例可以直接用print输出；
2. 从Person分别派生Teacher类和Student类：
   1. Teacher类增加属性title，实例可以直接用print输出；
   2. Student类增加属性stuNo，实例可以直接用print输出；
3. 分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；
4. 对数组执行以下要求：
   1. 分别统计Person、Teacher和Student对象的个数并放入一字典中，统计完后输出字典内容；
   2. 对数组按以下要求排序并输出：age、fullName、gender+age；

enum Gender: Int{

case male

case female

static func >(p1: Gender, p2: Gender) -> Bool {

return p1.rawValue < p2.rawValue

}

}

class Person{

var firstName:String

var lastName:String

var age:Int

var gender:Gender

var fullName:String{

get{

return firstName+" "+lastName

}

}

//具有指定构造函数和便利构造函数

init(firstName:String,lastName:String,age:Int,gender:Gender){

self.firstName = firstName

self.lastName = lastName

self.age = age

self.gender = gender

}

convenience init(name:String) {

self.init(firstName:name,lastName:"",age:18,gender:Gender.male)

}

var description:String {

return "Name:\(fullName) Age:\(age) Gender:\(gender)"

}

static func ==(p1:Person,p2:Person) -> Bool {

return p1.description == p2.description

}

static func !=(p1:Person,p2:Person) -> Bool {

return p1.description != p2.description

}

}

class Teacher:Person{

var title:String

init(firstName:String,lastName:String,age:Int,gender:Gender,title:String){

self.title = title

super.init(firstName:firstName,lastName:lastName,age:age,gender:gender)

}

convenience init(name:String) {

self.init(firstName:name,lastName:"",age:18,gender:Gender.female,title:"cs")

}

override var description:String {

return super.description + " title:\(title)"

}

}

class Student: Person{

var stuNo:Int

init(firstName:String,lastName:String,age:Int,gender:Gender,stuNo:Int){

self.stuNo = stuNo

super.init(firstName:firstName,lastName:lastName,age:age,gender:gender)

}

convenience init(name:String) {

self.init(firstName:name,lastName:"",age:18,gender:Gender.female,stuNo:60)

}

override var description:String {

return super.description + " grade:\(stuNo)"

}

}

/\*\*方法一

extension Person: Equatable {

static func ==(lhs: Person, rhs: Person) -> Bool{

return lhs.description == rhs.description

}

}\*/

let p1 = Person(firstName:"li",lastName:"guiyang",age:43,gender:Gender.male)

let p2 = Person(name:"gaoyuexiang")

//Person实例可以直接用print输出

print(p1.description)

print(p2.description)

let s1 = Student(name:"fan")

print(s1.description)

let t1 = Teacher(name:"tan")

print(t1.description)

//两个Person实例对象可以用==

print(p1==p2)

//两个Person实例对象可以用!=(好像有问题)

//print(p1!=p2)

//分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；

var manyPerson = [Person]()

manyPerson.append(p1)

manyPerson.append(p2)

manyPerson.append(s1)

manyPerson.append(t1)

for item in manyPerson{

print(item.description)

}

//分别统计Person、Teacher和Student对象的个数并放入一字典中，统计完后输出字典内容

var PersonNum = ["Person":0,"Teacher":0,"Student":0]

for item in manyPerson{

if item is Student {

PersonNum["Student"]! += 1

}else if item is Teacher {

PersonNum["Teacher"]! += 1

}else{

PersonNum["Person"]! += 1

}

}

for (key, value) in PersonNum.enumerated() {

print("字典 key \(key) - 字典 (key, value) 对 \(value)")

}

//原始数组

print("---------------原始数组---------------")

for item in manyPerson {

print(item)

}

//根据age从大到小排序

print("-------------age从大到小-----------------")

manyPerson.sort { return $0.age > $1.age}

for item in manyPerson {

print(item)

}

//根据全名从前往后排序

print("-------------全名从前往后-----------------")

manyPerson.sort { return $0.fullName < $1.fullName}

for item in manyPerson {

print(item)

}

//根据gender和age从大往小排序

print("----------------gender和age从大往小--------------")

manyPerson.sort { return ($0.gender > $1.gender) && ($0.age > $1.age) }

for item in manyPerson {

print(item)

}