目录

[SSH 2](#_Toc10955_WPSOffice_Level1)

[VUE 8](#_Toc4322_WPSOffice_Level1)

[2. 不使用extend也可以 9](#_Toc10428_WPSOffice_Level1)

[3. template只能有一个根结点，下面这张会报错 10](#_Toc11393_WPSOffice_Level1)

[1. 创建组件 12](#_Toc551_WPSOffice_Level1)

[2. 配置路由（是一个数组） 12](#_Toc21897_WPSOffice_Level1)

[3. 实例化一个router对象 12](#_Toc14551_WPSOffice_Level1)

[4. 把router对象配置到vue对象中 12](#_Toc10047_WPSOffice_Level1)

[5. 在router-view里显示 12](#_Toc16638_WPSOffice_Level1)

[1. 文件末尾要回车到下一行 即在</template>后回车一下 13](#_Toc31499_WPSOffice_Level1)

[2. 路由键的冒号后要与值间隔个空格，如path: ‘sd’ 13](#_Toc21627_WPSOffice_Level1)

[3. import from ‘’也要加空格 13](#_Toc6424_WPSOffice_Level1)

[三、jquery中ajax获取 13](#_Toc24935_WPSOffice_Level1)

[Bootstrap栅格系统： 16](#_Toc27443_WPSOffice_Level1)

[移动设备适配 17](#_Toc30228_WPSOffice_Level1)

[Python 17](#_Toc11092_WPSOffice_Level1)

[spyder里面tab补全！！ 17](#_Toc24023_WPSOffice_Level1)

[a,b=b,a+b 看成等号左右两边b赋a，a+b赋b 17](#_Toc5045_WPSOffice_Level1)

[例1：将两个文件中的键值对，键相同的拼接起来插到另一个文件 18](#_Toc32088_WPSOffice_Level1)

[例2. 用matplotlib绘图 19](#_Toc1128_WPSOffice_Level1)

[例3 爬虫 20](#_Toc12866_WPSOffice_Level1)

[Pandas 21](#_Toc29379_WPSOffice_Level1)

[技巧 22](#_Toc22917_WPSOffice_Level1)

[a,b,c=students #列表也可以这样赋值 叫做列表解包 22](#_Toc22821_WPSOffice_Level1)

[py词云 22](#_Toc30851_WPSOffice_Level1)

[scrapy 23](#_Toc20955_WPSOffice_Level1)

[#Request使用 25](#_Toc10473_WPSOffice_Level1)

[BeautifulSoup 使用： 26](#_Toc7726_WPSOffice_Level1)

[数据库连接 26](#_Toc9214_WPSOffice_Level1)

[y1=df\_china\_daily[‘suspect’].values 28](#_Toc8931_WPSOffice_Level1)

[c.render\_notebook() 30](#_Toc29096_WPSOffice_Level1)

[visa 是信用卡或者签证 31](#_Toc28079_WPSOffice_Level1)

[文件夹遍历和统计 （OS模块） 31](#_Toc8324_WPSOffice_Level1)

[s = [ 32](#_Toc13175_WPSOffice_Level1)

[线性回归预测房价（机器学习） 33](#_Toc22458_WPSOffice_Level1)

[Xpath 34](#_Toc30028_WPSOffice_Level1)

[动态爬取全球肺炎（腾讯网）\* 35](#_Toc22647_WPSOffice_Level1)

[y1=data['累计确诊'].values 36](#_Toc10882_WPSOffice_Level1)

[y2=data['新增确诊'].values 36](#_Toc29747_WPSOffice_Level1)

[手写体识别 36](#_Toc88_WPSOffice_Level1)

[验证码生成 40](#_Toc1035_WPSOffice_Level1)

[42. 44](#_Toc12743_WPSOffice_Level1)

[43. 44](#_Toc29256_WPSOffice_Level1)

[44. 44](#_Toc29850_WPSOffice_Level1)

[45. 45](#_Toc32594_WPSOffice_Level1)

[46. 45](#_Toc21133_WPSOffice_Level1)

[47. 45](#_Toc17767_WPSOffice_Level1)

[48. 45](#_Toc4906_WPSOffice_Level1)

[49. 45](#_Toc13084_WPSOffice_Level1)

[50. 45](#_Toc771_WPSOffice_Level1)

[2.定义字符指针char \*str=”ABCDE” 51](#_Toc25035_WPSOffice_Level1)

[再来一遍 53](#_Toc237_WPSOffice_Level1)

[二维数组指针运算~！！！ 61](#_Toc15435_WPSOffice_Level1)

[浙大总复习题 62](#_Toc7763_WPSOffice_Level1)

当文档目录显示出正文内容时：

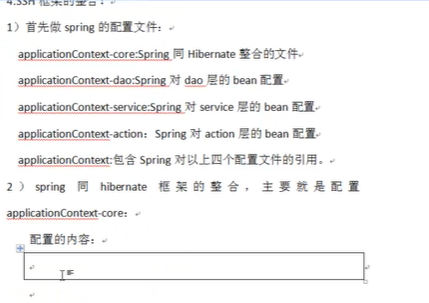
把文档切换成大纲视图，凡是有zd目录的地方都有个+号，找到错误位置，把光标在加号后面点一下，在WORD的上面会出现这条目录的级别专，改成所要的级别就行，如果不是目录就改成正文文本，改完属后回到页面视图，目录，右键更新全部目录就好了。

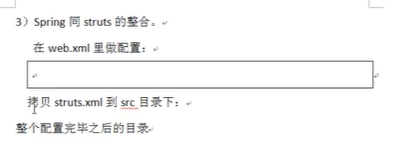
ctrl -R 是替换 -F是查找

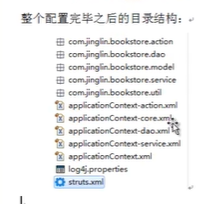
ctrl - alt -L是自动格式化

DNS是domain name system，就是域名系统，把域名和IP相对应，翻译官

SSH





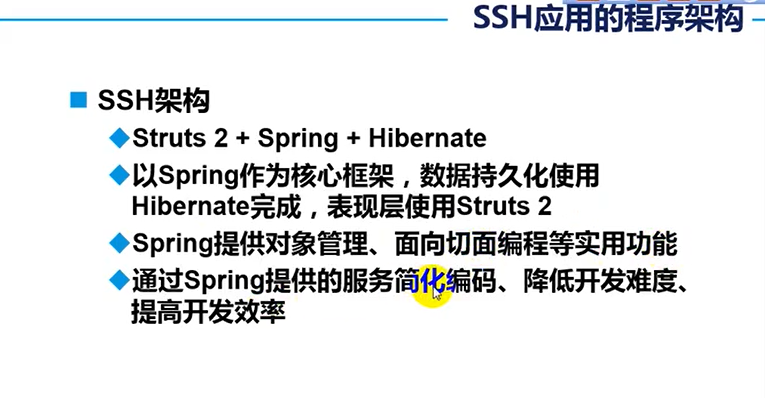


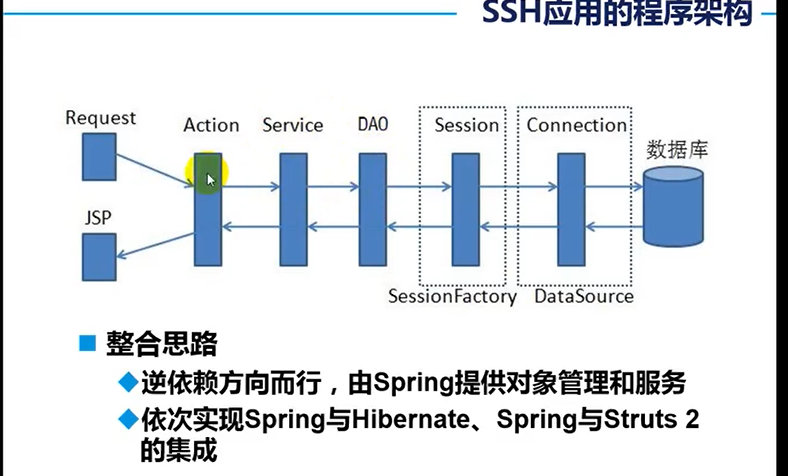
创建config,src,test(单元测试)三个文件夹

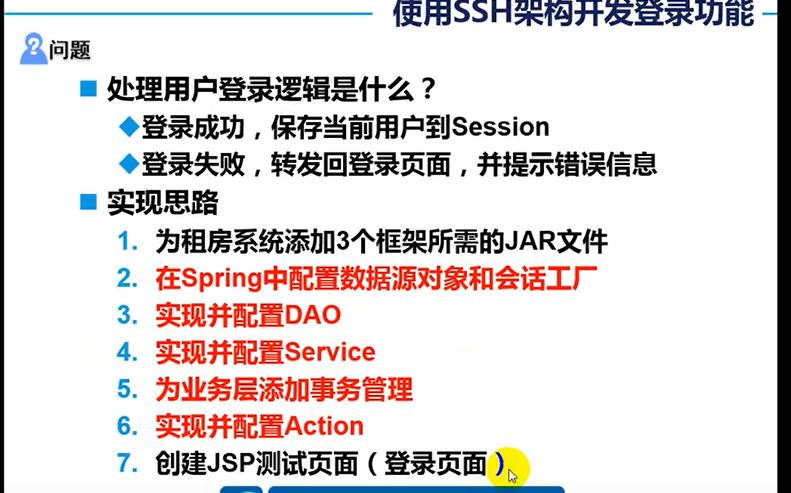
在src里创建一个包com.tx.dao,再在这个包创一个接口empDao.java，再创建个实现类的包。

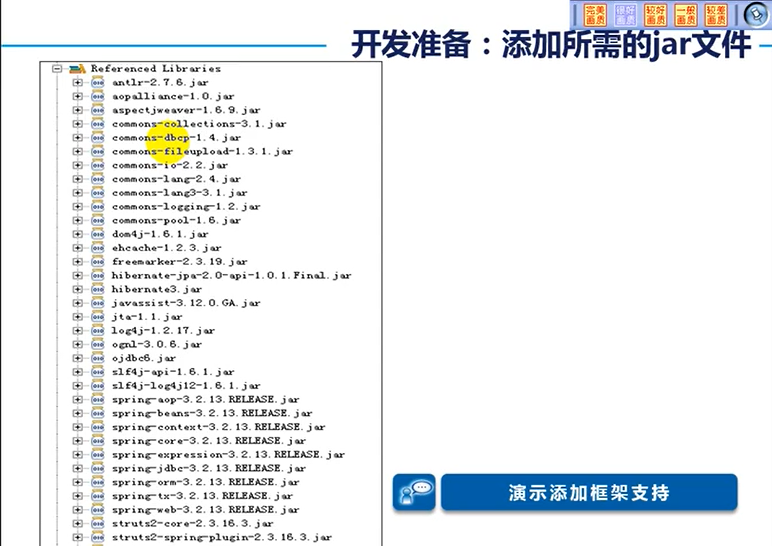
同理即在src下创建cn.tx.contoller,cn.tx.dao,cn.tx.model,cn.tx.service,cn.tx.utils，其中dao和service里还有个实现类

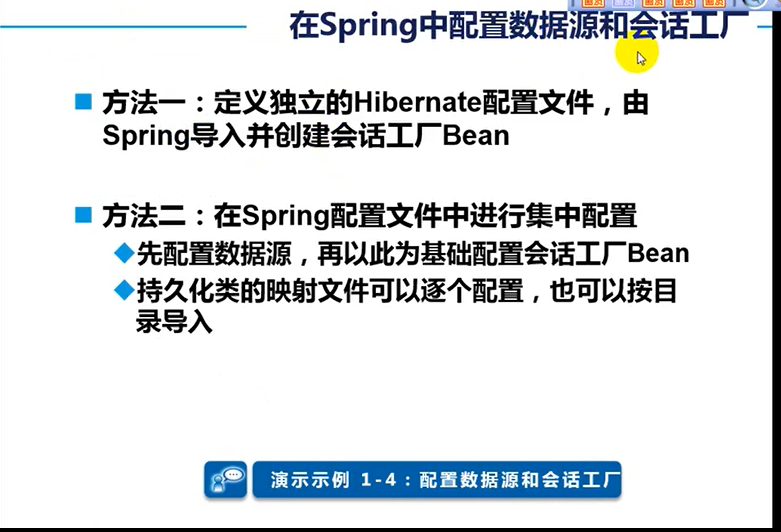
北大青鸟版





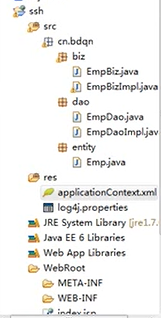




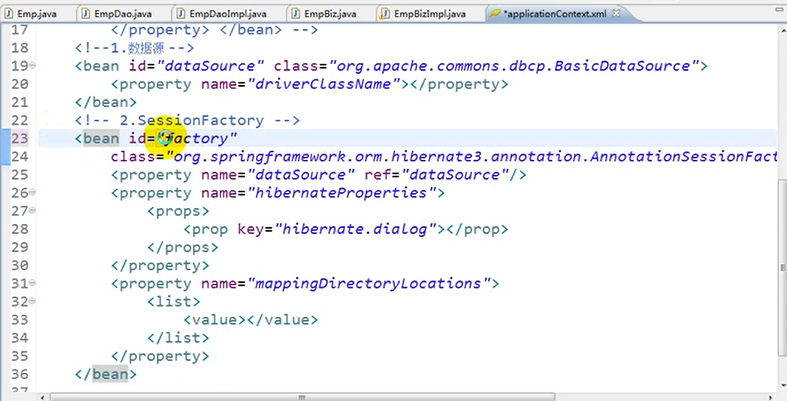


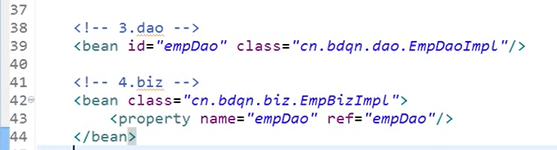
 Session接口是Hibernate向应用程序提供的操纵数据库的最主要的接口,它提供了基本的保存,更新,删除和查询的方法

src下创包和类分层，注意来个实现类 biz貌似是service层



Spring整合Hibernate

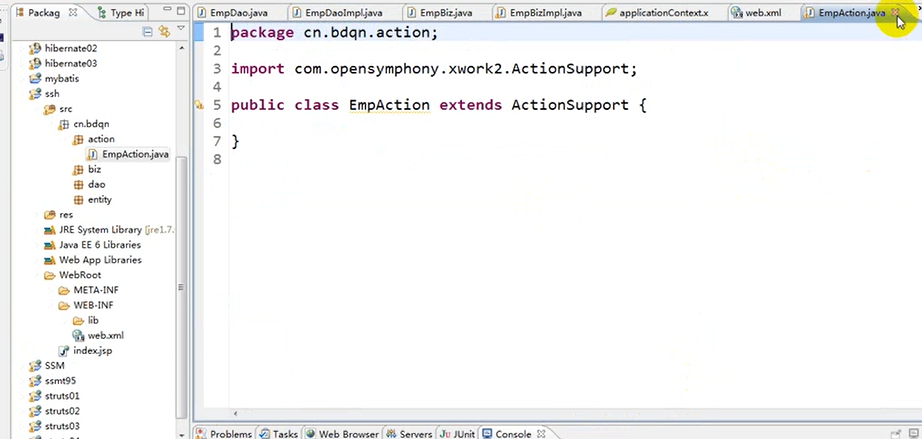




然后在web.xml设置启动读取上面的配置，从而spring和hibernate都读取了，后面就差整合struts2了

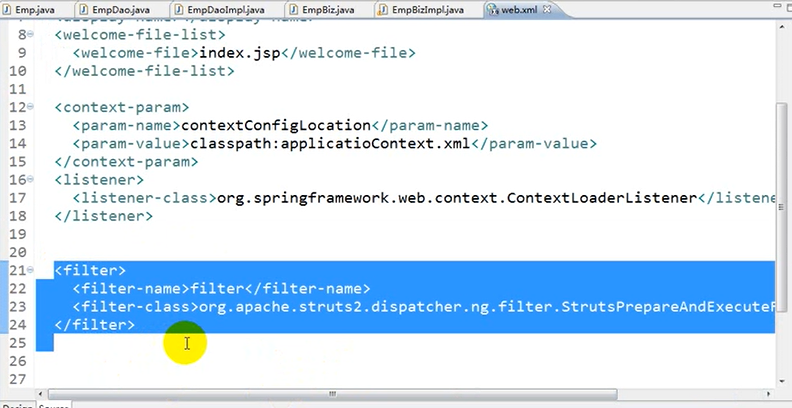


下面搞struts，添加action包，和类，然后在res中添加struts2.xm来配置

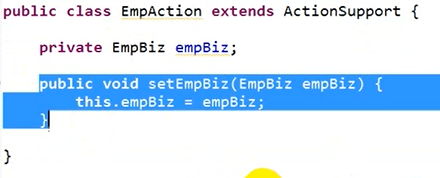




在web.xml添加struts的action，从而SSH三者可以互相找到了

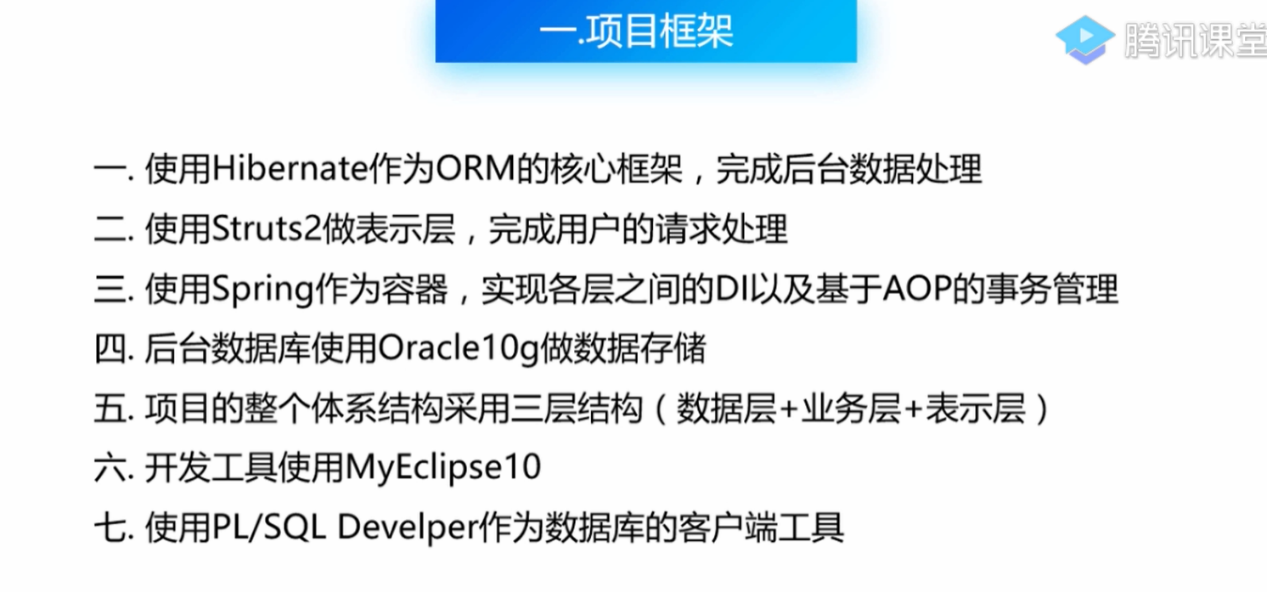


action类中添加set方法，service也这样，设值注入



后面就add hibernate，add struts

美食城



VUE

·MVVM是前端视图层的分层，为Model View ViewModel

MVC是后端的分层

·输出东西时，专业的前端是用console.log(),而不是alert()

因为console.log可以输出对象而alert不行

·1 ==’1’ 会转换类型，这样会返回true

而三等号会报错

·创建一个user对象

let user={

id: this.id; //上面data中有id和name字段

name:this.name;

}

·箭头函数使用，filter函数内就需要输入一个函数作为参数

this.userlist.filter(item=>{

if(item.name.include())

return true

}

·双花括号里可以输入表达式，即会将表达式解释为普通文本，而非HTML代码

{{name==1?男:女}}

·属性绑定v-bind:title=”msg” 缩写为：//此时的msg是vue对象中data中的属性

<!-- 完整语法 -->

<a v-bind:href="url">...</a>

<!-- 缩写 -->

<a :href="url">...</a>

事件绑定v-on:click=”cio()” 缩写为@ //无参方法可以省略() ,可缩写成@

<!-- 完整语法 -->

<a v-on:click="doSomething">...</a>

<!-- 缩写 -->

<a @click="doSomething">...</a>

双向绑定v-model

v-model 绑定的值是一个静态字符串 或 布尔值，但在业务中，有时需要绑定一个动态的数据，这时可以用 v-bind 来实现。

<input type="radio" v-model="picked" name="radio" :value="value1"> 单选按钮

<input type="radio" v-model="picked" name="radio" :value="value2"> 单选按钮

Vue定义组件时可以像实例一样定义属性，如data，method等，但是data必须用renturn返回。

·样式CSS方式

1. class定义：class=”red” 再定义一个.red{ color:red}
2. 行内样式，直接添加style=”color:red”
3. v-bind:class=”[‘red’,’yellow’]” 使用v-bind可以加class数组或者对象绑定（即其中写的是js代码）
4. v-bind:class=”flag ? ’red’ : ’size’ ”
5. v=bind:class={‘red’:true} //使用对象来表示，即括号里也可以定义为一个对象

· V-show和V-if都可以控制元素的隐藏显示，v-show是简单的css变换，而v-if是基于DOM

·@click=”flag=!flag” 可以实现点击逆转flag

·过渡动画

·vue定义组件时可以用驼峰，但是在使用时要转为小写并在前加个-

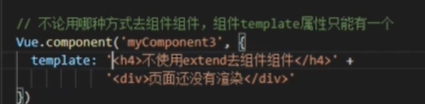
即myComponent --》my-component

·创建组件的方式：

1. vue.extend 
2. 不使用extend也可以



1. template只能有一个根结点，下面这张会报错



template也可以在外面定义（注意加个id）

然后在里面的话就是template:”#abc”

·ES6新语法，反引号(在1左边)可以添加多行字符串，省的字符串拼接了，像python中的三引号

而如果在反引号中需要使用变量，则可以使用模板语法${num}

·vue组件的data必须是一个方法而且返回一个对象

·vue提供了<component></component>来展示组件，用属性 ：is来指定组件

·props是用于父组件向子组件传递数据

·node是js的运行时环境，node给前端项目做配置

·npm是node提供的一个包管理工具，类似于maven，通过npm去安装依赖包，就不需要在页面上使用script来引入了

·-g全局安装，任何项目都可以用，-d表示开发环境安装，不会打包到生产，-s生产安装

·npm install可以简写成npm i

·li:even 是选择偶数位的，li：odd是选择奇数的，css（“backgroundcolor”,”yellow”）是设置某个元素的某个css属性的值

\*\*

运行vue 项目，可进入该文件夹，在地址栏输入cmd进入该文件夹的cmd，然后输入

cnpm run dev

\*\*

v-model 可应用于input标签，从而与之绑定

\*\*

输出{{ message}}

\*\*

子元素获取父元素数据用props

父元素获取子元素数据用emit



安装脚手架，创建项目

1. cnpm install vue-cli -g 安装vue-cli
2. vue init webpack-simple 项目名称 (simple稍微少点东西可去）

初始化项目，生成项目模板

1. cnpm install 先进入项目目录(cd vue-cli-demo)，安装模块包 (.html页面 怎么认识 .vue文件)
2. cnpm run dev 启动项目 启动后就可以在浏览器中运行了（localhost：8080）

使用路由方式：

1. 创建组件
2. 配置路由（是一个数组）

var routes=[

{path:”/home”,component:home}

]

1. 实例化一个router对象

var router=new VueRouter({

routes:routes //将上面定义的routes赋来

})

1. 把router对象配置到vue对象中
2. 在router-view里显示

定义路由：path:”/”，redirect：“/login” （这句话表示的是默认是跳转到/login(下面会定义的））

vue项目思路：

1. 在index.html里定义了一个div，id是app
2. 与index.html对应的js文件是main.js（在main.js中定义了一个vue对象，用于配置，也就是说main.js是全局配置文件，如在main.js中引入了一个elementui并Vue.use()，就可以在所有的.vue文件中使用这个，但是有些如axios（ajax使用，如jquery的$post）不能用，只能在单独的想用的页面中use）
3. 默认index.html调用的是APP.vue
4. 在router目录下的index.js可以配置路由并创建router对象（需要在首部先把所用的组件导入import进index.js）
5. 而在main.js中import进index.js所定义的router对象，配置到vue对象中（即在vue中加一个属性router:router）

从而配置路由结束，会在index.html中显示

6.在APP.vue中有个router-view,会显示路由配置里面调用的那个组件

1、main.js是我们的入口文件，主要作用是初始化vue实例，并引入所需要的插件

2、App.vue是我们的主组件，所有页面都是在App.vue下进行切换的。其实你也可以理解为所有的路由也是App.vue的子组件。所以我将router标示为App.vue的子组件

所以可以在APP.vue里引入其他组件

import Test from './components/Test'

export default {

  name: 'App',

  components: { Test }

}

在一个组件中使用另一个组件，先import,再添加到components中，就可以在上面使用<Test>标签了

UI可以使用elementui，在Vue.use后，在各自的.vue文件中可以进入elementui的官网找模板贴到.vue文件中即可

Vue.nextTick()作用：在下次dom更新循环结束之后，执行延迟回调。在修改数据之后立即使用这个方法，获得更新后的dom

在函数setTimeout中直接使用this不会得到data的变量，而是window对象，要解决这个

可以定义一个\_this暂存this，或者使用箭头函数setTimeout(() => {  
//你要执行的事件  
}, 1600)

箭头函数中访问的this实际上是其父级作用域中的this，箭头函数本身的this是不存在的，这样就相当于箭头函数的this是在声明的时候就确定了（因为相当于作用域嘛），这个特性是很有用的

箭头函数如果是一个参数的话就不用加括号了

每个Vue实例都实现了事件接口

vm.$on(event,calback):监听当前实例上的自定义事件，事件可由vm.$emit触发。

vm.$emit(event,[...args])：触发当前实例上的事件，附加参数会给监听器回调

@input 可以实时监听input标签输入，而v-model是双向数据绑定，不能实时

render函数是为了用js创建HTML，渲染

router-link to属性既可以用路径，也可以用name

fetch为Promise对象

async用于加在定义函数为异步函数

而await用于在调用函数前加，也会异步执行它，一般加在fetch前

常见错误：

1. 文件末尾要回车到下一行 即在</template>后回车一下
2. 路由键的冒号后要与值间隔个空格，如path: ‘sd’
3. import from ‘’也要加空格

过滤器可应用在v-bind 或者{{ }}中

过滤器可以串联：

{{ message | filterA | filterB }}

在这个例子中，filterA 被定义为接收单个参数的过滤器函数，表达式 message 的值将作为参数传入到函数中。然后继续调用同样被定义为接收单个参数的过滤器函数 filterB，将 filterA 的结果传递到 filterB 中。

## 三、jquery中ajax获取

用ajax。

引入jquery的话，

|  |  |
| --- | --- |
| 1  2  3 | $("button").click(function(){     $("div").load('test.html');  }); |

如果你要获取这个页面某个id或者类的html还可以这么写

|  |  |
| --- | --- |
| 1  2  3 | $("button").click(function(){     $("div").load('test.html #container');  }); |

 这个可以直接把新页面的[html元素](https://www.baidu.com/s?wd=html%E5%85%83%E7%B4%A0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)加载到指定的div或者别的元素里面。

JS访问网页并获取其中HTML内容

//用于创建XMLHttpRequest对象

function createXmlHttp() {

//根据window.XMLHttpRequest对象是否存在使用不同的创建方式

if (window.XMLHttpRequest) {

xmlHttp = new XMLHttpRequest(); //FireFox、Opera等浏览器支持的创建方式

} else {

xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");//IE浏览器支持的创建方式

}

}

//直接通过XMLHttpRequest对象获取远程网页源代码

function getSource() {

var url = document.getElementById("url").value; //获取目标地址信息

//地址为空时提示用户输入

if (url == "") {

alert("请输入网页地址。");

return;

}

document.getElementById("source").value = "正在加载……"; //提示正在加载

createXmlHttp(); //创建XMLHttpRequest对象

xmlHttp.onreadystatechange = writeSource; //设置回调函数

xmlHttp.open("GET", url, true);

xmlHttp.send(null);

}

//将远程网页源代码写入页面文字区域

function writeSource() {

if (xmlHttp.readyState == 4) {

document.getElementById("source").value = xmlHttp.responseText;

}

}

//JSON.parse解析json字符串为json对象

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myObj = JSON.parse(this.responseText);

document.getElementById("demo").innerHTML = myObj.name;

}

};

xmlhttp.open("GET", "/try/ajax/json\_demo.txt", true);

xmlhttp.send();

//默写

var xmlhttp=new XMLHttpRequest();

xmlhttp.onreadystatechange=function(){

if(this.readyState==4){

myobj=JSON.parse(this.responseText);

document.getElementById(“text”).innerHTML=myobj.name;

}

xmlhttp.open(“GET”,”http://baidu.com,true);//第三个参数true表示异步，false为同步（即等到数据获取到再返回）

xmlhttp.send():

*在vvscode里快速初始化一个html模板可用!+tab*

*在Navicat里面连接数据库首先要打开数据库服务，即wampserver*

*mounted{*

*this(1)*

*axios.get(“gee.json”).ge(){*

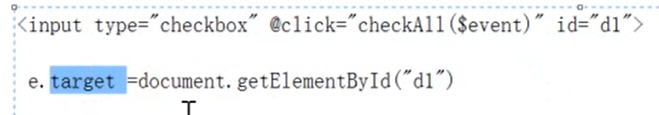
*this(2)*

*}*

*}*

*里面写的是页面加载后挂载后执行的*

1. *处的this是指vue的对象*
2. *处的this是指axios的对象*



*Bootstrap栅格系统：*

*“行”（row）必需包含在.container或.container-fluid中*

*"行"（row）的直接子元素必需是“列”（column）*

*"行"（row）的负值margin抵消了.container的padding.使得“列”（column）的两端距离container边界是15px.*

*如果column大于12，多余的column所在元素整体的另起一行排列*

**添加meta**

<meta name="viewport" content="width=device-width, initial-scale=1, maximun-scale=1, user-scalable=no" />

initial-scale:初始缩放比例

maximun-scale:最大缩放比例。

user-scalable:禁止用户缩放

在我们写布局的时候，我们不喜欢或者不想让两个容器挨在一起，我们可以用[Bootstrap](https://blog.csdn.net/weixin_42973273/article/category/9315029)的：列偏移来实现！

我们可以使用：col-md-offset-\*来实现列偏移效果！相当于：CSS  margin效果

data-toggle指以什么事件触发，常用的如modal,popover,tooltips等，data-target指事件的目标,你这段代码的意思就是指将#signin-signup-tab这个Dom元素的内容以模态框的形式展示。

两个输入框分别放在两个 form-group中包裹起来。 然后在两个form-group最外层再包裹一层 form-inline即可。

<div class="form-inline">

    <div class="form-group">controls</div>

    <div class="form-group">controls</div>

</div>

移动设备适配

*视口 view-point：*移动设备上的viewpoint就是屏幕上能用来显示我们网页的一片区域。即：就是浏览器上用来显示网页的去区域。

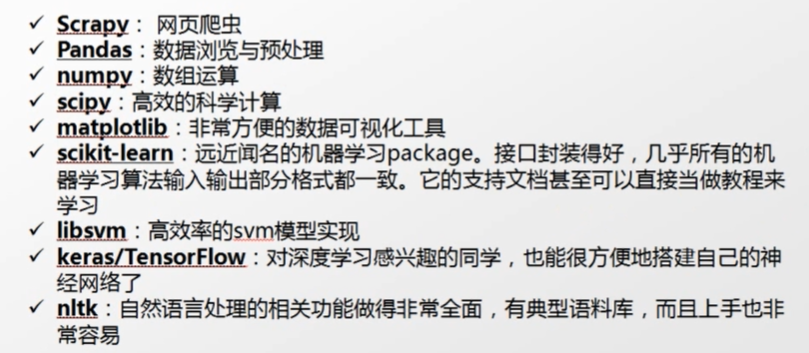
用meta标签来得到理想的视口布局；  
<meta name="viewport" content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=1,user-scalable=no" />

1.width=device-width---当前的viewpoint宽高等于设备的宽高（可适配）

1. user-scalable=no---是否允许用户缩放；默认值为yes；若设置为no

*Python*

*spyder里面tab补全！！*



*a,b=b,a+b 看成等号左右两边b赋a，a+b赋b*

*打开文件输入路径可以前面加个r表示不转义，即可不用输入双斜杠f1=open(r’F:\temp\user.txt’)*

*爬虫requests+BeautifulSoup+re*

*BeautifulSoup学习*

一、find()用法 find(name,attrs,recursive,text,\*\*wargs) 这些参数相当于过滤器一样可以进行筛选处理，不同的参数过滤可以应用到以下情况： 查找标签，基于name参数 查找文本，基于text参数 基于正则表达式的查找 查找标签的属性，以及基于attrs参数 基于函数的查找

用法p=soup.find('ul') ，那么返回结果是第一个ul标签以及<xx>...</xx>的所有内容，即上面的代码；注意若用p=soup.find('ul').get\_text()那么结果不是...的所有内 容，而应该是plants 10000 algae 10000，即...中的标签不算text文本。

相比find()，find\_all()有个额外的参数limit，如下所示： p=soup.find\_all(text='algae',limit=2) 实际上find()也就是当limit=1时的find\_all()

*即*

findAll(tag, attributes, recursive, text, limit, keywords)

find(tag, attributes, recursive, text, keywords)

*如*

*.findAll("span", {"class": "green"})*

*也可是个列表多个*

*.findAll("span", {"class":{"green", "red"}})*

*.findAll({"h1","h2","h3","h4","h5","h6"})*

class**在 python 中是关键保留字，不能再将这些字作为变量名或过程名使用，所以**class\_**应运而生。**

*定义字典 al={}即可，可用于保存从文本文件中读取的信息*

*pass可以在函数里占位*

*{*

*如果py文件作为模块被导入(import)，那么\_\_name\_\_就是该py文件的文件名(也称 模块名)；*

*如果py文件直接运行时(Ctrl+Shift+F10)，那么\_\_name\_\_默认等于字符串”\_\_main\_\_”;*

*所以 if \_name\_==’\_main\_’*

*r()*

*意思是在直接运行py文件时调用r函数*

*}*

*例1：将两个文件中的键值对，键相同的拼接起来插到另一个文件*

*从文件中取出以逗号分隔的信息 文件里是以逗号分隔的(姓名，电话)*

*def extract():*

*add={} #初始化字典*

*file=open(“add.txt”,’r’)*

*lines=file.readlines();*

*for line in lines:*

*line=line.strip() #去空白*

*content=line.split(‘,’) #取得数组*

*add[content[0]]=content[1] #插入字典*

*file.close() #open就要close*

*lines=[] #初始化数组*

*head=”姓名\t 电话\t 地址\t”*

*lines.append(head)*

*for key in add:*

*line=’ ‘*

*if key in add2.keys(): #如果add的key也在add2中出现，则拼接它们三个*

*line=line+‘\t’.join([key,add[key],add2[key]])+’\n’ #用\t连接在它们三个之间，且join参数必须是可迭代，故用列表[]*

*else:*

*line=line+‘\t’.join([key,add[key],’\*\*\*\*\*\*\*’])+’\n’#当第二个没有时用\*代替*

*lines.append(line)*

*tips: with语句后面的对象必须要有\_\_enter\_\_和\_\_exit\_\_方法*

*with as可以巧妙地处理异常(相当于try finally)，也可以自动帮助分配和释放资源(文件操作)，这样就不用close了*

*使用:with open(‘rr.txt’,’w’,encoding=’utf-8’) as newfile: #encoding参数可省略*

*newfile.writelines(lines)*

1. *用matplotlib绘图*

*import matplotlib as plt*

*import numpy as np*

*x=np.linespace(-3,3,100) #在-3,3内平均生成100个点，（数组）*

*y=np.sin(x) #用x列表生成y*

*plt.plot(x,y) #用x，y坐标绘制图形*

*tips:*

*plt.plot(x, y, format\_string, \*\*kwargs): x为x轴数据，可为列表或数组*

*pyplot并不默认支持中文显示，需要rcParams修改字体来实现*

*import matplotlib*

*matplotlib.rcParams[‘font.family’] = ‘STSong’*

*matplotlib.rcParams[‘font.size’] = 20*

*设定绘制区域的全部字体变成 华文仿宋，字体大小为20*

*plt.xlabel()：对x轴增加文本标签*

*只希望在某地方绘制中文字符，不改变别的地方的字体*

*在有中文输出的地方，增加一个属性： fontproperties*

*eg:*

*plt.xlabel(‘横轴：时间’, fontproperties = ‘simHei’, fontsize = 20)*

*plt.ylabel()：同理*

*plt.title(): 对图形整体增加文本标签*

*plt.text(): 在任意位置增加文本*

*plt. annotate(s, xy = arrow\_crd, xytext = text\_crd, arrowprops = dict)*

*: 在图形中增加带箭头的注解。s表示要注解的字符串是什么，xy对应箭头所在的位置，xytext对应文本所在位置，arrowprops定义显示的属性*

*plt.xlim(-10,10) 设定范围*

*plt.axis([-1, 10, 0, 6]) : x轴起始于-1，终止于10 ，y轴起始于0，终止于6*

*plt.subplot(3,2,4) : 分成3行2列，共6个绘图区域，在第4个区域绘图。排序为行优先。也可 plt.subplot(324)，将逗号省略。*

*绘各种类型图*

*plt.plot(x,y , fmt) ：绘制坐标图*

*plt.boxplot(data, notch, position): 绘制箱形图*

*plt.bar(left, height, width, bottom) : 绘制条形图*

*plt.barh(width, bottom, left, height) : 绘制横向条形图*

*plt.polar(theta, r) : 绘制极坐标图*

*plt.pie(data, explode) : 绘制饼图*

*plt.scatter(x, y) :绘制散点图*

*plt.hist(x, bings, normed) : 绘制直方图*

*例3 爬虫*

*控制台里network里显示了所有所发出的请求*

*header的user-agent里会显示浏览器名字，如果用爬虫的可能会显示python，反爬虫手段之一*

*解析方式：1.正则表达式2.BeautifulSoup（常用）3.PyQuery（类似jquery）4.Xpath（树）*

*import urllib*

*requests包是常用的发出请求的模块*

*response=requests.get(url) #以get形式发送，返回response*

*if response.status\_code==200 #判断状态码*

*return response.text*

*定义headers为（从浏览器里面copy一下即可）*

*tips:get函数里还可以加个header参数即get(url，header=headers)从而使用自定义的header可以修改user-agent为浏览器形式，从而躲过反爬虫。*

*beautifulsoup 使用*

*soup=BeautifulSoup(html,’lxml’) #html是response.text返回到的，lxml用来解table=soup.find(‘table’,attr={‘class’:’list’}) 析html*

*trs=table.find\_all(‘tr’)*

*for item in trs[1:-1]:*

*tds=item.find\_all(‘td’) #找到里面所有的td*

*name=tds[0].get\_text() #get\_text()获取html（td）标签里的文本*

*job=tds[1].get\_text()*

*不想访问第一个和最后一个可以用切片list[1:-1] #-1表示最后一个，且为左闭右开*

*Pandas*

*Pandas模块的数据结构主要有两种：1.Series 2.DataFrame*

*Series 是一维数组，基于Numpy的ndarray 结构*

*Pandas 会默认然用0到n-1来作为series的index, 但也可以自己指定index( 可以把index理解为dict里面的key )*

*1.创建series(用一维列表) ： pd.Series([9, 'zheng', 'beijing', 128]) 得到索引为0,1,2.*

*2.创建dataframe(用嵌套列表)：pd.DataFrame([[‘T’,1],[‘E’,2],[‘F’,3]])*

*他们都可以自定义索引，*

*如dataframe：pd.DataFrame([[‘T’,1],[‘E’,2],[‘F’,3]],columns=[‘name’,’age’])*

*也可以用随机二维数组来创建DataFrame(np.random.randn(6,4),columns=[‘name’,’age’])*

*访问dataframe： 行：df.iloc[0] 列df[‘name’]*

*访问某个元素，横纵均为数字可用df.iloc[0,0],有一个不为数字用df.loc[0,’name’]*

*添加一列(直接赋):df[‘gender’]=[‘F’,’M’,’M’]*

*例，从文件中读取*

*df=pd.read\_csv(r‘book.csv’) #如果是写的话就是to\_csv,前面加r是不转义*

*输出前十行df.head(10),输出后五行df.tail(5)*

*df.describe()*

*其物理意义在于观察这一系列数据的范围。大小、波动趋势等等*

*技巧*

*py的列表元素可以不同类型*

*1.a,b,c=’name’,10,230*

*2. student=[‘name’,’asa’,’ad’]*

*a,b,c=students #列表也可以这样赋值 叫做列表解包*

*3.students=[10,20,30,40,50]*

*a,b,\*c=students 得到a=10,b=20,c=[30,40,50] #个数不对称时，可以用\**

*接收，此时c为列表*

1. *a,b=b,a #可以实现变量交换*
2. *函数也可以多返回值，接收时也这样接收 return 3,4,5*
3. *if else 一行写：b=a if a>0 else -a*
4. *判断某个变量是否在一堆值之中 if a in (‘A’,’B’,’C’)*
5. *python中可以a<x<b了，if 30<a<50:*
6. *判断空只需要把变量名作为判断条件即可(字典，列表等），a=[],if a:*
7. *判断多个条件中至少一个成立，用any ,任何一个成立就成立*

*if any([x>40,y>50]):*

1. *判断多个条件全部成立，用all*
2. *同时遍历列表元素和下标 用enumerate*

*for i,item in enumerate(list):*

*print(i,’:’,item)*

1. *从两个列表中生成字典，用zip,再用dict强转*

*name=[‘a’,’b’,’c’]*

*age=[2,3,4]*

*x=dict(zip(name,age)) #不一致的话就把多的部分删了*

1. *匿名函数 lambda x:express ，用作函数嵌套里如filter函数需要的模板函数*
2. *列表推导式 li=[i for i in range(1,101) if i%2==0]*
3. *生成器（小括号g=（i for i in range(1,101)））不像列表推导式一次就把值全输出，而是需要一个用一个，相当于给你一个方法，要用自己算（next（g）来算）*

*py词云*

*使用wordcloud库中的WordCloud类*

*from wordcloud import WordCloud*

*%matplotlib inline#这是 IPython 中定义的魔法函数（Magic Function）,其意义是将那些用于matplotlib绘制的图显示在页面里而不是弹出一个窗口，因此就不需要plt.show()这一语句来显示图片*

*import matplotlib.pyplot as plt*

*with open(‘text.txt’,’r’) as f:*

*text=f.read()*

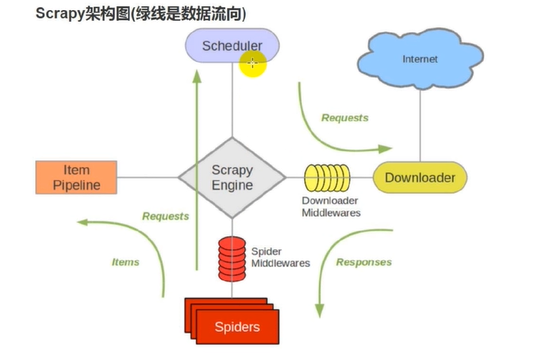
*wc=WordCloud()*

*wc.generate(text)*

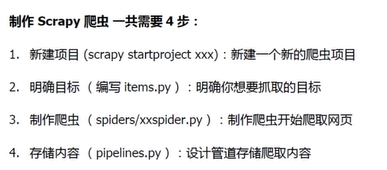
*wc.to\_file(a.png) #可以输出为图片*

*plt.imshow(wc) #也可以借助plt输出在本页面*

*scrapy*











import urllib.request

import pymysql #将搜出来的结果存数据库

movielist=[] #定义存储列表

url=”http://movie.douban.com/top250”

#Request使用

def get\_html():

res=urllib.request.urlopen(url) #url的选择很重要，如果当前url获取不到页面数据，看获取数据的url到底是哪个！去network里看size大点的看response是哪个，再看header里的requesturl，callback可省略即保留callback=即可，一串数字是时间戳可保留

html=res.read().decode()

return html

‘’’如果是用requests库的话 import requests

就可以response=requests.get(url) #post需要带数据

if request.status\_code==200:

request.text

BeautifulSoup 使用：

from bs4 import BeautifulSoup

def parse\_html()：

mysoup = BeautifulSoup(htmlfile,’html.parser’)

movie\_zone=mysoup.find(‘ol’)

movie\_list=movie\_zone.find\_all(‘li’)

for movie in movie\_list:

movie\_name=movie.find(‘span’,attrs={‘class’:’title’}).gettext() #获取文本

movielist.append(movie\_name) #添加到数据库

nextpage=mysoup.find(‘span’,attrs={‘class’:’next’}).find(‘a’)#找下一页链接

if nextpage: #如果有下一页

new\_url=url+nextpage[‘href’]

parse\_html(get\_html(new\_url)) #递归调用继续用新url解析

数据库连接

def save\_data(movelist):

conn=pymysql.connect(host=’localhost’,user=’root’,password=’mysql’

,db=’test’)

mycursor=conn.cursor()

sql=’CREATE TABLE movie250(ID VARCHAR(10),name VARCHAR(20))DEFAULT CHARSET=utf8’

mycursor.execute(sql)

for id,movie in enumerate(movielist): #enumerate函数可以生成id

sql=”INSERT INTO movie250 VALUES(%s,%s)”

mycursor.execute(sql,(id,movie)) #把id，movie动态插入占位符%s

conn.commit()

mycursor.close()

conn.close()

爬取疫情的数据

先在network中找到真正的url，一般是动态加载的，是js，而且要大点的文件，在preview里看

response=requests.get(url)

if response.status\_code==200:

data=response.json()['data'] #可以将json数据转换为python字典

import json

data\_dict=json.loads(data) #json.loads用于将已编码的 JSON 字符串解码为 Python 对象

#因此response.json()和json.loads(response.content)都可以将json编

码的数据转化为python类型字典

然后用这个这个字典来绘图（需要先转换为DataFrame）

import pandas as pd

df\_china\_daily=pd.DataFrame(data\_dict['chinaDayList'])

#转化为DataFrame后就可以将哪列作为x，哪列作为y了

import matplotlib.pyplot as plt

x=df\_china\_daily[‘data’].values #需要使用values转换为数组才能传入绘图函数，原本是series类型

y1=df\_china\_daily[‘confirm’].values

y1=df\_china\_daily[‘suspect’].values

plt.figure(figsize=(15,8)) #画板

plt.plot(x,y1,marker='^',label='Confirm')#画两条线

plt.plot(x,y2,marker='\*',label='Suspect')

plt.legend()#图例

plt.show() #显示

#可以通过xticks和xlable 调整x轴刻度密度

#然后要画疫情地图的话，可以引入pyecharts

pip install pyecharts

pip install echarts-china-provinces-pypkg

live\_data\_dict=json.loads(live\_data)

province\_list=[] #准备把各省信息存进列表

for province in live\_data\_dict['areaTree'][0]['children']:#先看下字典的内容，再通过key访问字典，找到各个省份的信息所在

name=province['name'] #把name单独取出

data=province['total'] #把total字典赋给data

data['name']=name #然后把name加到total字典中

province\_list.append(data)

df\_province\_list=pd.DataFrame(province\_list)#转化为dataframe

provinces=list(df\_province\_list['name'].values) #然后再把它们取出来作为列表

confirm=list(df\_province\_list['confirm'].astype(int)) #注意numpy的数 字类型需要转换为int才认识

from pyecharts.charts import Map #使用pyecharts显示地图

from pyecharts import options as opts #选项用于传入map参数

pieces=[ #设定显示各种颜色的显示范围

{'min':0,'max':9},

{'min':10,'max':99},

{'min':100,'max':499},

{'min':500,'max':999},

{'min':1000,'max':9999},

{'min':10000}

]

c=( #add 中把两个列表打包后作为列表传给map

#

Map()

.add('疫情地图',[list(z) for z in zip(provinces,confirm)],'china')

.set\_global\_opts(visualmap\_opts=opts.VisualMapOpts(is\_piecewise=True,pieces=pieces))

)

c.render\_notebook()

实在找不到可以返回数据的目标url时，还可以使用selenium 自动化测试工具，他可以驱动浏览器，像人一样操作，从而可以直接获取那些爬虫爬不动的url返回数据，就不用自己找url了。

pip install selenium

安装webdriver(chromedriver)

from selenium import webdriver

driver=webdriver.Chrome() #打开一个chrome浏览器

driver.get(url)

html=driver.page\_source #存放在这个属性中

driver.close()

#使用beautifulsoup解析

from bs4 import BeautifulSoup

mysoup=BeautifulSoup(html,’lxml’)

china\_list=mysoup\_find(‘div’,class\_=’chianList’)#注意是class\_，不是class，因为class是python的关键字，所以后面要加个尾巴，防止冲突

province\_list=china\_list.find\_all(‘div’,class\_=’placeItemWrap ’)

data\_list=[]

for province in province\_list:

name=province.find(‘h2’,class\_=’blue’).gettext()

added=province.find(‘div’,class\_=’add ac\_add ‘).gettext()

confirm=province.find(‘div’,class\_=’confirm‘).gettext()

dead=province.find(‘div’,class\_=’dead‘).gettext()

heal=province.find(‘div’,class\_=’heal‘).gettext()

data\_list.append({‘name’:name,’added’:added,’confirm’:confirm,’dea` d’:dead,’heal’:heal})

Robot协议：每个网站后面都有个robots.txt指示什么可以爬

如http://www.taobao.com/robots.txt

下面user-agent:baiduspider或者googlespider 就是百度或者goole搜索引擎，搜索引擎其实就是一个爬虫

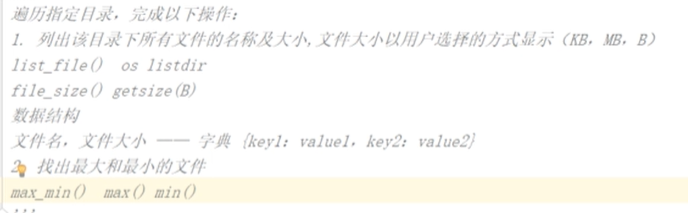
下面disallow的可能是网站的关键信息，但是也可以看（浏览器go）

还有sitemap 给你提供站点地图(求你别爬我正网了，有压力的，给你地图自己看更简单。)

所以我们可以用request.add\_header(‘User-agent’,”...”)伪造浏览器

visa 是信用卡或者签证

文件夹遍历和统计 （OS模块）



os.path模块主要用于文件的属性获取

注意os.listdir显示的不是全部路径，只是文件名，所以需要join父路径

还有walk（）函数更好，自动递归了，返回三元组

import os

files={} #用字典存文件：大小

root\_dir=r”F:\temp” #一定要加个r才能不然它转义

def list\_file(roo\_dir):

if os.path.isfile(root\_dir,KB=True):#如果是文件就存进去

size=file\_size(root\_dir,KB=True) #调用自定义函数获取大小，选择KB为单位

files[root\_dir]=size #存入字典

else: #如果为目录就对内部循环

for f in os.listdir(root\_dir):

file\_path=os.path.join(root\_dir,f) #因为isile函数需要绝对路径，所以需要拼 接为完整路径

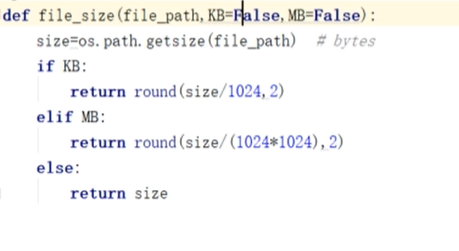
if os.path.isfile(file\_path):

size=file\_size(file\_path,KB=True)

files[file\_path]=size

else:

list\_file(file\_path) #如果还是为目录的话就递归



默认的max(),min()函数是按照字典的键来比较，但是我想比较value

传入命名参数key，其为一个函数，用来指定取最大值的方法

s = [

{'name': 'sumcet', 'age': 18},

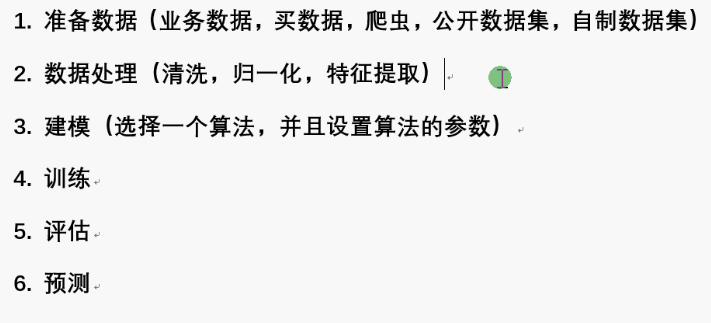
{'name': 'bbu', 'age': 11}

]

a = max(s, key=lambda x: x['age'])



线性回归预测房价（机器学习）



scikit-learn 是强大的机器学习库

在jupyter中当前页面画图需要用%matplotlib inline就不用plt.show()

%matplotlob inline

import matplotlib pyplot as plt

import numpy as np

from sklearn.datasets import make\_regression #Sklearn提供一些标准数据，我们不必再从其他网站寻找数据进行训练。例如我们上面用来训练的load\_iris数据，可以很方便的返回数据特征变量和目标值。除了引入数据之外，我们还可以通过load\_sample\_images()来引入图片。

from sklearn.linear\_model import LinearRegression #引入线性回归模型（model=LinearRegression()）

#先准备数据

x,y=make\_regression(n\_samples=100,n\_features=1,noise=10)

plt.scatter(x,y) #先看看生成的数据，自己生成的数据不用处理直接建模即可

#建模

model=LinearRegression()

#训练

x=x.reshape(-1,1) #先对数据进行格式变换才能适应

y=y.reshape(-1,1)

model.fit(x,y)

#评价

train\_score=model.score(x,y)

#开始预测新数据

new\_house=[125,130]

new\_house=np.array(new\_house).reshape(-1,1)

pred=model.predict(new\_house)

print(pred)

#保存模型

from sklearn.externals import joblib

joblib.dump(model,’houses.pkl’)

#保存后下次load就可以使用这个模型

mymodel=joblib.load(‘houses.pkl’)

深度学习介绍：

深度学习，别名是无监督学习，不需要手工选取特征，自动选取特征

神经网络

感知器：多种因素输入返回一个值，各个因素有权重，相加再与threshold阈值比较得出结果，即对后面神经元的抑制和激活

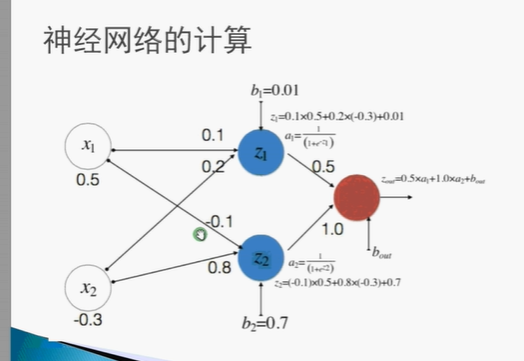
S型神经元：就是在感知器计算结果后再应用sigmoid函数，让原来的非0即1的输出变为0到1之间，即从阶跃函数变为sigmoid函数

从神经元到神经网络

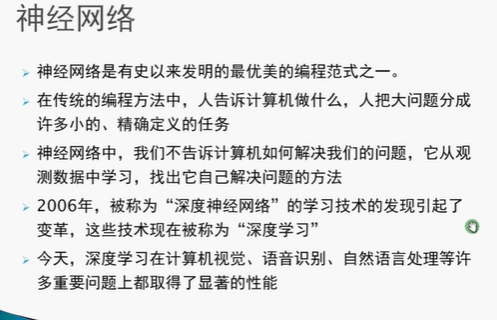
每一个隐含层都是向量输入，向量输出

同层的单元输出构成该层的输出向量

输入层-->隐含层-->输出层



多层，从输入到输出之间的层数越多，越复杂

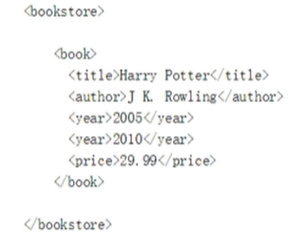




Xpath

对于绝对路径是以/开头的，而相对路径不用直接来

在xpath中是以1开始的，而不是0



1.取上面的jkrowling，两种方式：可以逐级查找/bookstore/book/author 或者直接查找 //author（这样只有当author唯一才行）

2.取上面的2010,由于是第二个year，可以/bookstore/book/year[2],或者//book/year[2]

提取数据

以a标签为例：

取其中中文本数据，a/text(),

取某个属性，a/@href ,a/@title

Chrome插件Xpathhelper

可以确认自己写的对不对

通过xpath解析DOM树的时候会使用lxml的etree，可以很方便的从html源码中得到自己想要的内容

代码：

from lxml import etree

import requests

url=’http://www.planning.org.cn/news/newslist?cid=12&page=1’

html=requests.get(url).text

txt=etree.HTML(html) #实例化构造一个Xpath解析对象，如为本地文件，可以把HTML-->parse

txt.xpath(‘//h4/a/text()’)

动态构造url需要用for循环和format

dt=[]

for i in range(1,300):

url=’http://www.planning.org.cn/news/newslist?cid=12&page={}’.format(str(i))#填充i

html=requests.get(url).text

txt=etree.HTML(html)

txt2=txt.xpath(‘//div[@class=”fsdsd”] ’) #使用//标签[@属性=”xx”]可以直接定位

for t in txt2[1:]: #从第二个开始

row=[]

title=t.xpath(‘h4/a/text()’)[0] #此时下面已经为相对路径,[0]为第一个数据

url=t.xpath(‘h4’/a/@href)[0]

row.append(title)

row.append(url)

dt.append(row) #添加完row再把row添加到dt，相当于二维数组了

pd.DataFrame(dt).to\_excel(‘./data/xx.xlsx’) #将这个二维数组（列表嵌套列表）转化为DataFrame，就可以存了

爬取城市道路信息

动态爬取全球肺炎（腾讯网）\*

import requests

from bs4 import BeautifulSoup

import time

timestamp=int(time.time()\*1000)

url=’https://news.qq.com/zt2020/page/feiyan.htm#/global’ #是动态页面，异步加载，不能访 问到所需数据，是通过其他的js

#所以先进入network，刷新页面，就可以看见所有请求，然后选择JS文件看较大的，再把目标数据的Headers中的requestURL复制即可

#url=’https://view.inews.qq.com/g2/getOnsInfo?name=disease\_foreign&callback=jQuery34106049878571372072\_1586004757559&\_=1586004757560’

#删掉callback后面的，再把\_后面的时间戳用占位符实时更换，所以最终url

real\_url=’https://view.inews.qq.com/g2/getOnsInfo?name=disease\_foreign&callback=&\_=%d’%timestamp

response=requests.get(real\_url) #response.status\_code ,response.text

if response.status\_code==200:

print(‘succeed’)

else:

print(‘Failed’)

data=response.json() #注意异步加载的数据是json格式的从而要用这个函数把json转为python对象

#换selenium+pandas(因为数据在表格)来

from selenium import webdriver

import pandas as pd

driver=webdriver.Chrome() #模仿浏览器

driver.get(url)

html=driver.page\_source

driver.close()

soup=BeautifulSoup(html,’lxml’)

table=soup.find(‘div’,class\_=’listWraper’).find(‘table’)

#通过pandas库的read\_html直接获取表格数据，得到的将是目标网页所有table表格的list集合，prettify是规范代码

data=pd.read\_html(table.prettify())[0] #可以直接to\_csv了，已经是dataframe了

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode\_minus']=False #用来正常

data=data.head(10)#取前十个，不然太多了

x=data['地区'].values #需要使用values转换为数组才能传入绘图函数，原本是series类型

y1=data['累计确诊'].values

y2=data['新增确诊'].values

plt.figure(figsize=(15,8)) #画板

plt.plot(x,y1,marker='^',label='累计确诊')#画两条线

plt.plot(x,y2,marker='\*',label='新增确诊')

plt.legend()#图例

plt.show() #显示

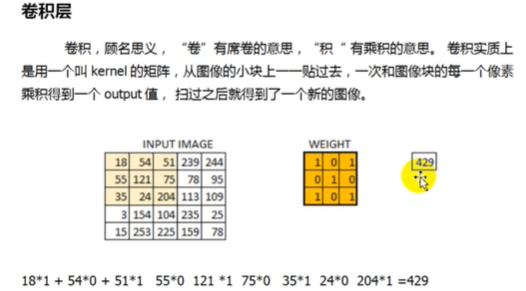
手写体识别

训练集用Mnist数据集

使用Keras深度学习框架基于theano/tensorflow/CNTK，主要使用GPU

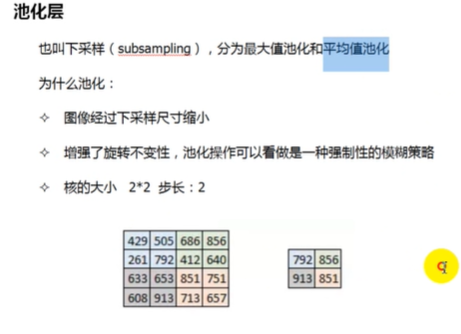
构建神经网络步骤：选择模型 构建网络层 编译 训练 预测

卷积神经网络包括卷积层和池化层



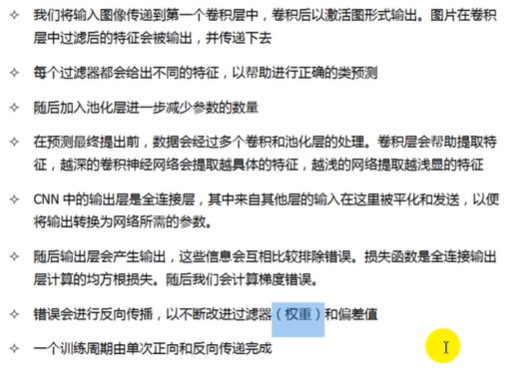
步长，一次滑动多少格

池化层



池化层夹在连续的卷积层中间， 用于压缩数据和参数的量，减小过拟合。  
简而言之，**如果输入是图像的话，那么池化层的最主要作用就是压缩图像**。







使用卷积神经网络CNN实现

分层识别

代码：

import keras

from keras.datasets import mnist #标准数据集

from keras.models import Sequential #序惯模型

from keras.layers import Dense,Dropout,Flatten #全连接层，防止过拟合，多维一维化

from keras.layers import Cov2D,MaxPooling2D #二维卷积层，最大值池化层

from keras import backend as K #引入后台，tensorflow一类的

#设置初始参数

batch\_size = 128 #一次给模型多少张图片

num\_classes = 10 #分多少类，一共十个数字

epochs = 12 #迭代多少次

img\_rows,img\_cols = 28,28 #mnist的图片都是28\*28的

(x\_train,y\_train),(x\_test,y\_test) =mnist.load\_data() #加载数据

#判断后台backend

if K.image\_data\_format()==’channels\_first’:

x\_train=x\_train.reshape(x\_train.shape[0],1,img\_rows,img\_cols)

x\_test=x.test.reshape(x\_test.shape[0],img\_rows,img\_cols)

input\_shape=(1,img\_rows,img\_cols)

else: #如果是tensorflow就需要把参数通道换一下

x\_train=x\_train.reshape(x\_train.shape[0],img\_rows,img\_cols,1)

x\_test=x\_test.reshape(x\_test.shape[0],img\_rows,img\_cols,1)

input\_shape=(img\_rows,img\_cols,1)

#数据处理 用浮点数表示灰度更精确，除255归一化

x\_train=x\_train.astype(‘float32’)

x\_test=x\_test.astype(‘float32’)

x\_train/=255

x\_train/=255

print(‘x\_train shape:’,x\_train.shape)

print(x\_train.shape[0],’train samples’)

print(x\_test.shape[0],’test samples’)

#下面y是标签，也需要做处理 如把5变成[000010000]

y\_train=keras.utils.to\_categorical(y\_train,num\_classes)#

y\_test=keras.utils.to\_categorical(y\_test,num\_classes)

#开始构建网络

model=Sequential() #初始化序列模型

#添加一个卷积层，有32个3\*3的卷积核

model.add(Conv2D(32,kernel\_size=(3,3),activation=’relu’,input\_shape=input\_shape))#卷积层1

model.add(Conv2D(64,(3,3),activation=’relu’) #卷积层2

model.add(MaxPooling2D(pool\_size=(2,2))) #池化

model.add(Dropout(0.25)) #防止过拟合 扔掉25%的数据

model.add(Flatten()) #压平，把多维压成一维的

model.add(Dense(128,activation=’relu’))#Dense全连接 relu为线性整流函数

model.add(Dropout(0.5)) #再丢

model.add(Dense(num\_classes,activation=’softmax’)) #全连接

#CNN参数：权重（卷积核的构成），卷积核的大小，数量，池化的大小，步长，dropout的rate

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=[‘accuracy’])

model.fit(x\_train,y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test,y\_test))#开始训练

score=model.evaluate(x\_test,y\_test,verbose=0) #测试集上测试

print(‘Test loss:’,score[0])

print(‘Test accuracy:’,score[1])

model.save(‘Model.model’)

验证码生成

PIL库与现在的pillow库不兼容，所以要先卸PIL，再装pillow

调用时还是调用PIL

from PIL import Image,ImageDraw,ImageFont

import numpy as np

im=Image.open(‘jjw.png’,’r’)

#Image库可支持裁剪，旋转，缩放，扩大，加水印，加滤镜

#流程：1.画一张200\*50的图片，白色

#2.逐个填充像素点，每个像素点随机

#3.往图片上写字符，字符随机生成，颜色随机，大写，小写，数字

#把这几个步骤分成几个函数1.填充背景色2.生成随机字符3.生成字符颜色

#4.生成验证码

def bgcolor():

＇＇＇

#填充背景色

#return： 返回一个随机颜色

＇＇＇

return (np.random.randint(64,255),np.random.randint(64,255),np.random.randint(64,255))

#返回三个随机值即RGB

def randomtxt():

‘’’

生成随机字符

ASCII 大写65-90 小写97-122, 数字48-57

return

‘’’

text\_list=[] #先把所有字符的ASCII码放在这个列表中，再随机取，最后返回把其转

text\_list.extend([i for i in range(65,91)]) #把一个列表接在另一个后面

text\_list.extend([i for i in range(97,123)])

text\_list.extend([i for i in range(48,58)])

return chr(text\_list[np.random.randint(0,len(text\_list)-1)]) #chr()即为转数字为对应ASCII字符

def text\_color:

‘’’

生成随机字符的颜色

‘’’

return (np.random.randint(32,127),np.random.randint(32.127),np.random.randint(32.127))

def generate\_code(): #产生验证码

width=200

height=50

image=Image.new(‘RGB’,(width,height),(255,255,255)) #画白色背景

draw=ImageDraw(image) #需要引入ImageDraw包，从而可以对其进行操作

for w in range(width): #遍历图片像素点

for h in range(height):

draw.point((w,h),fill=bgcolor()) #画一个像素点，调用上面函数颜色

myfont=ImageFont.truetype(‘arial.tff’,36) #字体与大小 ，需要导入ImageFont包

for i in range(4): #写四个字符

draw.text((60\*i+10,10),randomtxt(),font=myfont,fill=text\_color()) #位置，字符，字体,颜色

image.show()

generate\_code() #调用

requests使用

不建议使用py自带的urllib，而是用requests库

可以使用r=requests.get(),post()可以携带相关信息如header，后缀参数如页号,put(),delete(),head()

r.text 返回网页源代码

r.status\_code 返回状态码

当网页返回500时，可以通过添加headers参数（从network里面找requestheaders的useragent）来伪装

r=requests.get(url,headers=headers)

使用Jupyter notebook可以更好的进行数据分析

Series 是一维的，dataframe是二维的 两者是pandas的两种数据类型

ord函数可以字符作为参数，返回对应的ascll码；其中内置函数chr 与 ord函数作用相反，chr函数可以将ascll码转为对应的字符；

PIL图像处理库

Image：

From PIL import Image

Img=Image.open(r’D:\gg.jpg’,’r’)

w,h=img.size #获取图像的宽和高

geshi=img.format #获取图像格式

img.save(‘time’,’png’) #保存为time.png

img.thumbnail((50,50),resample=Image.BICUBIC) # 创建缩略图

img.show() #显示图片

# 色素分离

r, g, b = img.split()

# 调整大小

img1 = img.resize((100, 100))

# 旋转图像90度

img1 = img.rotate(90)

### socket编程思路

TCP服务端：

1 创建套接字，绑定套接字到本地IP与端口

   # socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) , s.bind()

2 开始监听连接                   #s.listen()

3 进入循环，不断接受客户端的连接请求              #s.accept()

4 然后接收传来的数据，并发送给对方数据         #s.recv() , s.sendall()

5 传输完毕后，关闭套接字                     #s.close()

TCP客户端:

1 创建套接字，连接远端地址

       # socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) , s.connect()

2 连接后发送数据和接收数据          # s.sendall(), s.recv()

3 传输完毕后，关闭套接字          #s.close()

HTTP请求返回值：

响应代码：200表示成功，3xx表示重定向，4xx表示客户端发送的请求有错误，5xx表示服务器端处理时发生了错误；

一个HTTP请求只处理一个资源

WSGI （网络服务器网关接口）

# C语言

复试C语言不用买什么资料做的，大题看看PAT乙级，结构体会考，其他题考的不难，小题填空，对错，说难一点就指针了，考的不复杂，建议谭浩强的书过一遍。还有，复试主要还是面试，好好准备

1.C语言中赋值表达式的值，就是表达式的右值，如while(k=0){}

2.当a为二维数组时：

\*(a+1)其中a表示整个数组首地址,指向对象是行,所以a+1表示首地址

所以\*(a+1)+2就表示向这一列后面移动2个位置

即为指向a[1][2]的地址

1. argc是命令行参数的个数
2. 返回值为char的函数，return 0是可以的
3. 以0开头的都是8进制，比如int x=010，就是x=8；
4. 外部变量跨文件使用的方法是，在文件1中函数外定义外部变量 int A；在文件2中函数外声明 extern A，即使用extern会先在本文件中找该定义，再到外面
5. 对于指向函数的指针，p++等无意义
6. 共用体所占内存等于最长成员的长度
7. 定义结构体变量Struct node i;
8. 共用体所占内存等于最长成员的长度

11.对于下面这题

t=a\*=a>b?a:b;

？：运算符优先级高于\*=和=运算符，所以先计算？：的值

12.

内联函数不适于while switch 这样复杂结构,且语句最好在1-5条这样的小型函数上. 递归不能定义为内联.

13.

结构体的总大小,也就是sizeof的结果,.必须是其内部最大成员的"最宽基本类型成员"的整数倍.不足的要补齐

32位编译器：

      char ：1个字节  
       char\*（即指针变量）: 4个字节（32位的寻址空间是2^32, 即32个bit，也就是4个字节。同理64位编译器）  
      short int : 2个字节  
      int： 4个字节  
      unsigned int : 4个字节  
      float: 4个字节  
      double: 8个字节  
      long: 4个字节  
      long long: 8个字节  
      unsigned long: 4个字节

  64位编译器：

      char ：1个字节  
      char\*(即指针变量): 8个字节  
      short int : 2个字节  
     int： 4个字节  
      unsigned int : 4个字节  
      float: 4个字节  
      double: 8个字节  
      long: 8个字节  
      long long: 8个字节

14.

C语言的标识符分为3类：关键字、预定义标识符和用户标识符。常量不属于标识符

15.

考选择题问哪个语句是正确的，注意后面有没有分号;

16.

c语言常量可以用十进制数、八进制数、十六进制数来表示：也可以用转义字符、长整型、科学计数法等来表示

而且科学计数的指数不能为小数，如1.2e0.5是错误的

17.

只能用字符数组存储字符串，什么字符指针，字符串变量都是错的

18.

函数内的复合语句中定义的变量，只能在该复合语句中起作用，在复合语句以外，该变量不起作用

19.

float a=b=10.0是错的，因为从右向左看10.0赋给b，但是此时b未定义

float \*a,b=&a；也错，因为此时a未初始化，不能赋值

20.

注意scanf(“%d”,xx) xx必须为地址而不是变量，即&x,或者指向x的指针

21，

c程序执行总是从main开始

22.

变量名不可以与关键字同名

23.

合法的标识符由字母（大、小写均可）、数字和下划线组成，并且必须以字母或下划线开头。

24.

a⊕b = (¬a ∧ b) ∨ (a ∧¬b) （异或）

即相同为0，相异为1

也即自身与自身异或为0

25.

数组名本身就是地址

26.

任何整数对30求余的范围是0~29

27.

scanf （ ）函数从输入设备按照指定的类型输入对应类型的若干个数据，遇到空格、制表符和回车时读取结束

28.

. double a = 1.5 ; a++ ;//结果是 a = 2.5（自增、自减的操作符可以是非整型的）

a%b 要求两操作数必须是整型，否则编译出错

/和！=可以不用是整型

29.

c语言隐含的数据类型是extern

30.

c语言中没有明确指出函数的类型，自动按整型处理

31.

逗号表达式：当顺序使用，结合顺序是从左至右，用来顺序求值，完毕之后整个表达式的值是最后一个表达式的值

z=（x+2，y=x）——带括号，赋值为z=y=x；

y=x+2，y=x；——赋值先y=x+2，之后为y=x；最终是y=x；

y=x+2，x；——赋值为y=x+2，x单独不是一个赋值语句。

32.

结构体指针两种使用方法：(\*p).x 和p->x

33.

0与‘\0’在条件判断中都属于0.

34.

二进制是机器识别的码，在c语言中不能被识别

0开始的是八进制数，数字里只能出现0-7

0x开始的是十六进制，数字里只能出现0-9到A-F

科学记数法

E或e前后必须是数字，后面必须是整数

且前后不能有空格

35.

字符串””

字符’’

转义字符前加 \ ，反斜杠

36.

函数strlen()计算的是字符的实际有效长度即不包含自负结束标志位’\0’

而array所占的空间为需要加1，以加上’/0’

37.

地址常量不能进行自增自减操作

38.

main函数的参数一个是整型变量，另一个是指针数组，即char \*a[]或char \*\*a

39.

scanf函数输入为变量的地址

40.

定义数组时,A[N] N必须为大于0的正整数

41.

字符串只能定义时赋值，而不能定义后再赋值 如char a[10]; a[10]="hello";这样是错误的。

也可以将字符串赋值给字符指针变量char \*a，或者赋给字符数组

42.

在printf(“\”\””) 即为输出””

\”即为转义输出”

43.

return 可以在一个函数里出现多次

42.

函数值类型缺省，则默认为int类型

43.

:宏替换不占用运行时间。宏替换是在编译时,编译器完成的。占用编译时间

（与函数调用相反）

44.

全局变量被分配在全局\静态存储区，而局部变量分配在栈区，动态内存分配在堆

45.

int s[4][5],(\*ps)[5]；

ps=s；

ps是一指向二维数组s的指针，ps+1表示指向数组s第2行首地址的指针；

\*(ps+3)表示数组s第4行的首地址；\*(ps+1)+3表示数组s第2行第4列元素的地址，

ps[0][2]是对数组元素的引用。

int (\*p)[4];表明p是行指针，即p指向二维数组中的一行。

p=a;这一句表示p指向a[0]，也就是指向a数组第0行的第0个元素（a[0][0])的地址。

p[2]表示p指向a数组的第二行，即指向a[2]行，也就指向a[2][0]的地址。

p[2]+3表示行指针指向a数组的第二行（从第0行开始），(从0开始)第3个元素，也就是a[2][3]，\*(p[2]+3)表示第2行，第3个元素的值。

46.

一个联合变量(Union)的所有成员共享同一个内存空间，因此一个联合变量在某一时刻只能保存其中一个成员的值。

一个联合变量要求的内存空间大小是有其最大的成员决定的

改变一个成员的值就会使之前存储在其他成员的值丢失

‘’

47.

任何类型的指针类型变量均占4个字节

48.

49.

共用体变量所占的字节为其中占字节数最多的成员所占的字节。

 结构体变量所占的字节为其中所有成员所占的字节数之和

50.

int x=040;求表达式x<<1

解：0是前缀，表示八进 制，即八进制40，转为二进制为100 000

左移一位为1 000 000 ，再转为八进制为100，即为,按八进制输出为100（输出时不输出前缀0）

按无符号八进制 %o

按无符号十六进制 %x

按无符号十进制 %u

按小数 %f

按指数 %e

按字符串 %c

按十进制 %d

51.

^为异或

&，| 且或

~为取反

52.

文件有随机存取和顺序存取

有文本文件和二进制文件

文件指，存储在外部介质上的数据集合

53.

c语言必须使用字符数组存储字符串

可以用%s来一次输出整个字符数组(字符串）  
而其他数组不可以，只能每次引用一个数组元素

54.

文件读取FILE \*fp;

fp=fopen(“file.txt”,”w”); //注意这里为双引号而非单引号

fprintf(fp,”%d”,x);

fclose(fp);

55.

声明有两种

int a是分配存储空间的

而extern a不分配存储空间

56.

\*a++等价于\*（a++）

自增自减运算符优先级大于指针运算符

57.

位与& 作用：

可以清0（与0000&），可以取中间某位(与0000111000)

异或作用：

特定位反转（与111111异或），0^1为1,1^1为0，即可达到反转效果

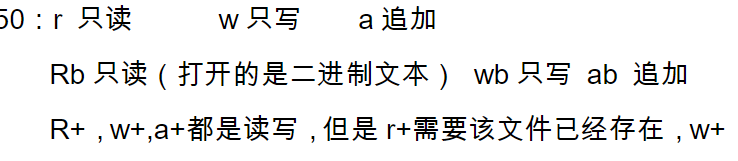
保留原位（与00000异或），0^0=0,1^0=1,不变

即与1异或反转，与0异或不变

交换两个值，不用临时变量：a=a^b,b=b^a,a=a^b

原理是一个整数与另外一个数进行两次异或运算仍然是其本身

(3) B = A ^B;(相当于道B = A ^ B ^ B ,即 B = A)  
(4) A= A ^ B;(相当于A = A ^B ^A ,即A = B)

1. 宏定义之间可以相互调用
2. 5\*7/9和5/9\*7是不一样的
3. argc是命令行参数个数
4. 以0开头是八进制，0x开头为十六进制
5. 
6. sqrt返回值是double类型

64.

源程序（高级语言） --> 目标程序（二进制）--> 与其他目标程序和系统库链接形成可执行程序。

65.

‘A’==65 ,’a’=97 两者差32

‘0’=48

float有32位 (float有4个字节，一个字节有8位）

其中24位表示小数，8位表示指数部分

66.

(int)x+y

x强制转换为int类型，而x+y的结果，有可能是y的类型，如果y不是int的话。

(int)(x+y)

这种写法是，先执行x+y，再强制转换为int类型，这样最终的结果，一定是int类型。

67.

int ,long赋值给char 就是把低8位(1个字节）赋过去，

68.

%-m.nd 的前面的 -表示左对齐，m表示输出的十进制总位数（即输出字段的宽度），n是小数点后的位数

69.

Scanf（“%3d%3d”,&a,&b）输入123456.

系统自动把123赋值给a，456赋值给b；同理，如果系统输入abc三个字符，但是ch只能容纳一个，就吧a给ch，bc给后面的

Scanf(“%s%s%s”,str1,str2,str3)； 而Scanf输入多个字符串的时候，用空格分隔，而输入%c字符时不用空格

而scanf(“%\*d”)表示输入项不赋给变量

%[^A]读入字符串以A结尾

用连字符可以说明一个范围。 例如，以下扫描集使 scanf() 接受字母 A 到 Z：%[A-Z]

要用scanf实现gets的功能，只需要这样写：scanf("%[^\n]",s);就可以了。

strlwr 表示将字符串都变小写

strupr 变大写

字符串（即字符数组）名字作为scanf参数不用取地址符&，其本来名字就是指针

70.

多维数组作为参数，只能省略第一维的参数

71，

数组只能在定义时赋值，而不能在后面的赋值语句中赋值

而用指针定义的数组可以在后面赋值，如char \*a;72.

分析数组还是指针 ()>[]>\*

一．首先看int \*p[4]，“[]”的优先级别高，所以它首先是个大小为4的数组，即p[4]；剩下的“int \*”作为补充说明，即说明该数组的每一个元素为指向一个整型类型的指针。

二．再看int (\*q)[4]。它首先是个指针，即\*q，剩下的“int [4]”作为补充说明，即说明指针q指向一个长度为4的数组

73.

scanf遇空格或回车结束

puts(str)和gets(str),是直到换行或EOF

EOF在c里面定义为-1，即遇到文件流缓冲满了会返回-1

74.

字符串的赋值不能用=，只能用strcpy

75.

函数只能嵌套调用不能嵌套定义

76.

变量类型有：

1. 自动变量（auto），变量的建立和撤销是系统自动进行的，也是局部变量，不初始化的话是随机值
2. 静态变量（static），也分为静态全局和静态局部，作用域不同，不初始化为0，静态全局变量作用域也只是在本程序中
3. 寄存器变量（register），存在CPU寄存器中，用于整型和字符型，用于使用频率高的变量，可以提高效率
4. 外部变量（extern），可在n个文件中共享变量，作用域是整个工程，比静态全局还牛逼

77.

预处理（#开头）有三种功能：宏定义，文件包含，条件编译

78.

条件编译(三种方式）

1.

#ifndef 标识符 //根据标识符是否定义来执行，可用#define定义

程序段1

#else

程序段2

#endif

2.

#ifdef 标识符 //如果定义这个标识符，可用#define定义

程序段1

#else

程序段2

#endif

3.

#if 表达式 //根据表达式真假来执行

程序段1

#else

程序段2

#endif

79.

宏定义分为不带参数和有参数的

无参数：#define X 5

...

#undef X

有参数：#define S(a,b) a\*b

注意带参数的宏定义与函数不同：带参宏定义调用时只是简单的字符替换，且不存在参数类型的问题，只代表一个符号

80.

文件包含有两种方式

#include “” //这种可以包含文件路径 常用！

而#include<文件名> //只能以指定的标准方式包含文件

81.

指针

可以直接处理内存地址，直接访问变量就是通过地址访问！而非通过变量名

将变量i的地址存储在另一个变量里，就是间接访问，这个变量就是指针变量

内存地址是无符号整型

82.

\*p++ ==>\*(p++) 自增运算符++优先级大于\*

又p++为先取p，再++，故先输出\*p，在把p指针值加1

\*++p ==>\*(++p)

先把p指针加1，再输出\*p

而（\*p）++，是把p指向的元素值加1

注意：

a+1 ==>a[1]

a[1]+1 ==>a[1][1]

83.

二维数组指针！

a[0],a[1],a[2],a[3] 可表示二维数组每行的首地址

即a[i]=&a[i][0]

a[i]+j表示第i行第j个元素的地址

即a[i]+j=&a[i][j]

84.

float \*f[10] 基本类型是数组，即指针数组

float (\*f)[10] 由于小括号的优先级大于中括号，所以基本类型是指针，即指向一维数组的指针

int (\*p)(int x) 为函数指针

int \*p(int x) 返回值是int类型指针

对于函数指针，++p，这种运算无意义

85.

只有字符串数组可以通过数组名输出全部元素，而数值型数组不行

86.

定义字符串两种方式：1.定义字符数组char str[]=”ABCDE”

2.定义字符指针char \*str=”ABCDE”

而且只有第2种能在已定义后赋值，因为指针牛逼。

87.

\*p+1 ==>\*优先级大于+所以是p指的元素值+1

而\*p++ ==>\*优先级小于++，所以==>\*(p++),而后置的++先不管，故返回值即为\*p,后面再把p++

即++>\*>+

88.

char \*a[] 即为指针数组，每个元素都是字符指针，可以分别指向一个字符串，即用来存储各种长度不定的字符串

如果是定的话，可以定义二维数组char a[4][5]

89.

命令行参数，即main的形参，int argc,char \*argv[]

90.

指针地址计算 p+c\*i (整型c=2,实型c=4，字符型c=1)

91.

p1，p2指向同一数组的，它们可以相减，相比较，但是相加无意义

92.

不允许对结构变量做整体输入，但是可以使用另一个相同类型的结构变量复制过来（共用体变量也可以这样）

93.

结构体变量.成员名 == (\*p).成员名 ==p->成员名

结构指针指向已经定义的结构变量的起始地址

94.

结构体数组也可以初始赋值struct person stu[10]={ { },{ },... }

95.

不能把共用体变量作为函数参数，但是可以使用指向共用体变量的指针

＊ 联合 : 可以让几种不同类型的变量存放到同一段内存单元中。

是一种类似于结构的复合类型。

任一时刻只能有一个变量起作用。

联合类型变量的定义形式是：

union 联合类型名

{

成员列表

} 变量名列表；

＊ 当编译程序看到关键字union时，就会为联合中最长的成员项保留一个足够大的空间。

联合通常用于保存函数调用的返回值，以便于为程序的其它部分按不同的数据类型引用值。

＊ 联合变量的引用方式

先定义了联合变量才能引用它，而且不能引用联合变量，只能引用联合变量中的成员。

＊ 联合的特点

﹡ 同一个内存段可用来存放几种不同类型的成员，但在每一瞬间只能存放其中一种，

即每个瞬间只有一个成员起作用，其它成员不起作用。

﹡ 联合中起作用的是最后一次存放的成员。

﹡ 联合变量的地址和它的各成员的地址相同。

﹡ 不能在定义联合变量的同时对其进行初始化。

﹡ 不能把联合变量作为函数的参数，也不能使函数带回联合变量，但可以使用指向联合变量的指针。

﹡ 使用联合比使用结构更能节省存储空间，但其代价是访问成员的速度要慢一些。

96.

结构体变量定义：

法1：

typedef struct{

int x;

char a[10];

}POINT;

POINT point1;

法2：

struct point{

int x;

char a[10];

}point;

法3：

struct point{

int x;

};

struct point point1;

字符数组char a[10],在输入时不用加&

其他的都要

再来一遍

96.

结构化程序设计方法：自顶向下，逐步细化，模块化设计，结构化编码

97.

字符是按照其整数存储的，C99把字符型数据作为整数的一种

98.

在编译时，一个源程序是一个编译单位，而不是以函数为编译单位

99.

形参是定义函数时后面的变量名

实参是在主调函数里调用时传进函数的

100.

++a||++b

||中当左边表达式为真时就不会计算右边表达式，所以只运算++a

101.

对局部变量使用static使由动态存储变为静态存储

对全局变量及函数使用static使其局部化，即局限于本文件中，函数变为内部函数。

总之，用了static，作用域都是局限的。

用了static，extern就无效了，即阻止extern扩展作用域

102.

gets的输入分割符只有回车，所以它能输入空格

而scanf(“%s”,a);

103.

int (\*PF)(int x); //PF为函数指针

int \*PF(int x); //返回值为指针

104.

char \*s; //作为函数输入参数

char \*p=s; //指针赋值

105.

定义变量时不能批量赋初始值，如int a,b=0; //a仍然不是0

106.

fseek(fp,offset(可负),begin)移动文件指示器到指定位置

rewind(fp)移动文件位置指示器到文件开头

ftell(fp)获取当前指向位置

fgets(str,n,fp) 从fp中读取n个字符到str中

fread(str,size,num,fp) 读取num个size大小的数据块存进str

w+ 可读可写，但是只能读刚刚写的那些

1. 结构体成员不能是该结构体变量，但可以是该结构体指针
2. 把缓冲区写入文件的过程叫做刷新
3. c语言合法字符常量有两种，1是普通字符如’b’,2是转义字符’\0’,
   1. 注意转义字符\后跟八进制,即每位上数字不超过8，如’\101’是字符，而’\85’不是， 且八进制为3位 ，0可省略, ‘\367’也行
   2. 也可跟十六进制’\x11’ ,,注意后面 为2位

110.

那就是问号表达式的结合性，问号表达式是从右往左的运算方向

对于a<b?a:c<d?c:d，

因为先执行的是c<d?c:d,会返回c,然后执行a<b?a:c

111.

赋值表达式返回的为其右值

112.

数组下标允许数据类型为 整型常量或者整型表达式

113.

类型相同指针之间不能用 +号

114.

字符串定义后就不能在直接赋值了，只能指针

115.

char a[3]={‘b’,’o’,’y’} 不能作为字符串，因为字符数组不会自带’\0’，不能作为字符串使用。

116.

函数实参与形参各自占用独立的内存单元

形参在调用时才有内存，用完就释放

117.

返回值由 被调函数的类型（return语句由被调回函数向主调函数返回值）决定 ，是在定义函数的时候确定的

118.

用字符指针数组存储一些字符串，可想成一行一行的字符串排列

119.

求x&y，就转为二进制

120.

操作数左移两位，结果乘4

121.

fopen第一个参数为文件名，\必须为双斜杠，因为会转义

122.

r+具有读写属性，从文件头开始写，保留原文件中没有被覆盖的内容；

w+具有读写属性，写的时候如果文件存在，会被清空，从头开始写。

123.

fwrite函数 是以二进制存进文件中

fprintf函数 是以文本型存进文件中

124.

c语言 运行需要经过编译和链接两步

125.

函数首部包括 函数类型，函数名，函数参数

126.

函数返回值类型是由函数类型决定的

127.

形参在不被调用前不分配空间

128.

x+=y+=z 赋值语句从右往左看，故先执行y+=z

129.

strlen(“1234\0xy”) strlen以\0为结束符，且不计算\0.，故为1234即为4

130.

二维数组在内存中存放顺序为 行顺序存放

131.

描述x或y中有一个小于z 的表达式：x<z||y<z

132.

main函数可以放在任意

注释不一定在语句后

133.

char型字符在内存中存储形式为 ASCII码

int类型在内存中存储是补码

134.

注意12&&12 = 1

而 12&12 =12 一个是位运算一个是逻辑运算

135.

？：三目运算符是判断逻辑是判断非零，而非与1比较，即等价为exp!=0

136.

char a[10]={‘E’,’a’,’s’,’t’,’\0’}

注意初始化会将未赋值的都赋为0字符，即空字符（\0）

所以c[5]->c[9]都可引用，且其值确定，均为空字符，和c[4]一样，均为’\0’

137.

调用函数时，实参将其值传递给形参，调用结束时形参并不将其值回传给实参

138.

函数定义不能嵌套，但函数调用可以嵌套

139.

对于int a[4][3] ,则a[1]+3表示a数组第2行第4列元素的地址

140.

定义结构体时，不能为成员设置默认值！会报错

141.

同类型的结构变量直接可以赋值，数组不能整体赋值，但是结构体内部的数组会随着结构体变量的复制而复制

142.

不能对结构体变量进行整体输入输出

143.

结构中的位域是将整型或无符号整型的变量压缩到指定二进制位数，没有地址，不能用&，节省内存，但效率低

144.

对结构体变量按照某个元素大小排序，可以使用冒泡

Struct node temp;

for(i=0;i<n;i++){

for(j=0;j<n-i-1;j++){

if(text[j].age<text[j+1].age){

temp=text[j];

text[j]=text[j+1];

text[j+1]=temp;

}

145.

文本文件行末和文件末有标记，而二进制没有

146.

文件操作函数操作成功是0，否则返回EOF

147.

while((ch=fgetc(fp1))!=EOF)

putc(ch,fp2);

while(!feof(fp)){

ch=fgetc(fp);

}

FireFox浙大模拟题

1. fclose关闭文件成功返回为0，否则为EOF
2. 数组名是地址常量，不能改
3. char \*aa[2]={“abcd”,”sdsd”}存放的是地址而非字符串的值

aa数组中两个元素各自存放了”abcd”,”sdsd”的首地址

aa数组两个元素分别存放的是含有5（注意字符串自带一个\0）个字符的一维数组首地址

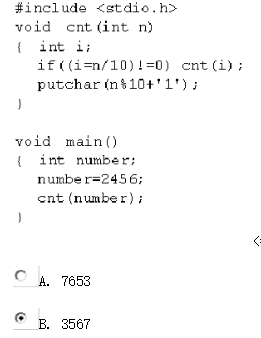
即数组存放的是地址，而非两个字符串的值

1. %\*3d 中的\*是赋值抑制符
2. &优先级大于&& 如1&&2&3 ,先算2&3，转为二进制010&011=010=2 然后1&&2 =1
3. int const \*x=&a const在\*左边，修饰的是指针所指向的变量,所以指针所指向的变量不能修改，指针本身的值是可以修改的。

int \* const x是修饰的是指针本身，即指针本身不可以更改

1. 宏定义的函数只是简单的替换，对计算会造成影响。如#define F(x) x\*x 调用F（a+b）时，会返回a+b\*a+b 就是简单的替换答案就错了
2. 011<<1^1 注意<<优先级大于^（异或），而011为8进制，转换为十进制为8+1=9，左移一位为乘2=18，二进制为10010，再与001异或==>10011=19，转为%o即8进制输出为2\*8+3\*1 ==>23

且与1异或即为最后一位取反，其他位不变。

1. 递归是要逆过来的，这个if下面的putchar就只有当执行到最后一个6时才会开始回调，即6-5-4-2，然后各位还得+1
2. int \*p, p++不是p指向的地址加1，因为int类型为4字节，p++可能一次加sizeof(int)=4
3. 把若干个独立的表达式结合成一个表达式，简称逗号表达式 。 它按从左到右的顺序执行组成逗号表达式的各表达式,而将最后一个表达式(即表达式n)的值作为逗号表达式的值。
4. &&被称为短路运算的原因是，当第一个操作数为false时，将不会判断第二个操作数,||运算符是短路或运算，被称为短路运算的原因是，当第一个操作数为true时，将不会判断第二个操作数。
5. 若打开文件的方式为w/w+,（只写），而该文件已经存在，则该文件内容被清空
6. (float)11/2是double型，因为float 11是float型，2是int型，两个都向double型上升
7. !((x>0)||(y>0)) <==>!(x>0)&&!(y>0)
8. <的优先级略高于==,对于int a=97; 表达式'b'<a==97<'1'的值。等价为(‘b’<a)==(97<’1’)

而’b’=98所以为0==0，答案为1

1. ++优先级大于\*
2. -->优先级大于\*，-->大于++
3. struct wc{ int a,\*b;}\*p;

int x0[]={11,12},x1[]={31,32};

static struct wc x[2]={100,x0,300,x1}; p=x; 则表达式\*(++p)->b用%d输出的值是

注意p为结构体指针，注意！！x0，x1是数组名，也就是指针！，也就是说数组x中100,x0为一组，可构成一个结构体，所以++p后就是下一个结构体，就是300,x1这一组，而取x1时即为取其首地址，也就是31

20.

构造斐波那契数列

如果m>n,需要交换m，n，则m=m+n,n=m-n,m=m-n

然后while(cur<=n){

cur=pre1+pre2; //pre1，pre2初始为1

pre1=pre2;

pre2=cur;

}

21.

注意每次循环输出的s可能是在上次输出的s的基础上加的

22.

将一个数分解为三个素数之和a+b+c

a=n%2+1

然后就在2到n/2中找b和c(n-b),找到同时满足b和n-b均为素数的就行

23.

return a+b就行不用加括号

24.

在c程序中，用非0值代表真，而不是1

25.

printf(“%d”,EOF) 为-1

系统幂函数为pow

26.

输入ABCDEFG#后输出ACE

for(c=getchar();getchar()!=’#’;c=getchar())

putchar(c);

注意在中间判断条件getchar()也会读取一个字符，所以c会隔一个输出

27.

赋值表达式是从右往左，逗号表达式是从左往右

x=5; x+=x-=x+x; 就是+优先级大于赋值号，且从右往左，x-=10 ==>x=-5

然后x+=x ==>x=-5+-5=-10

1. 对于a[5] , int \*p=a+1,int \*q=&a[3]，求q-p， 注意这里p，q为指针，所以与a[i]的值无关=2
2. if(k=0)..else.. if中的值为0，进入else
3. 宏定义是直接代换，

#define A 8

#define B (A<A+2)-2

printf(“%d”,B\*2);

代换得 (A<A+2)-2\*2=1-4=-3

1. ++p->y 注意->的优先级大于++，所以先取出y的值，再把y的值加1
2. 求π的近似值

do{

item=1.0/i;

pi+=fact\*item;

i+=2;

fact=-fact;

}while(item>=eps);

最后呼出4倍的pi

1. 求三个数的最小公倍数

aa=a;

bb=b;

c=a%b;

while(c!=0){

a=b

b=c

c=a%b

}

return aa\*bb/b;

调用时 ff(ff(x1,x2),x3)

1. 统计文件中字符数

while(!feof(fp)){  
 fgetc(fp);

count+=1;

}

printf(“%d”,count-1);

fclose(fp);

1. 删除字符串第i个位置后的n个字符

如果i直接大于字符串长度，直接return

如果s+i+n大于字符串长度，直接在s+i后截断，即加个’\0’

否则的话就是先把s移到s+i处，把s+i+n后面的复制到s+i后面，\*s=\*(s+n)

1. !(x>0||y>0)等价于!(x>0)&&!(y>0) ,少了括号都不行，!优先级大于>,<
2. 一个整数可以以%d,%o,%x(十六进制)输出，而不能以%f输出
3. 注意双引号围起来的是字符串，转到字符数组里中元素个数要加1（\0），而字符数组不用加
4. ++优先级大于+-等算术运算符
5. &a[0] 就是a数组的首地址，就是a
6. ！>&&>|| ++优先于\*
7. p=&j,q=&k,p=q,(\*p)++, 这样p，q都指向了k，所以p对k的改变会影响到q
8. 表达式!!5 =1
9. 输出0x19<<1&0x17

就先把十六进制的19每位转为4位的二进制0001 1001 左移一位变成0011 0010

而17 --> 0001 0111

两者位与，得0001 0010 即为十六进制的12H

1. #define MIN(x,y)   (x)<(y)?(x) : (y)     printf("%d",10\*MIN(10,15));

宏定义就替换10\*(10)<15?15:10 由于\*优先级大于< ,所以100<15为0，结果为10

1. char \*s; gets(s); 可能会出现错误，因为s没有赋值
2. char \*a[]={“avb”,”sdss”}

\*(a+1)+1 以%s输出为 dss，即进入第二个字符串的第二个开始输出

48.arr1[4][0];

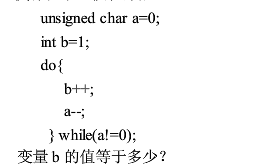
\*arr2[4];

arr1[4]是字符指针常量，arr2[4]是字符指针变量

1. char arr1[10][100];

str2 = \*( arr1 + 4 )

arr1+4是指向arr1第4行字符的指针，进行间接访问后，得到第4行字符这个一维数组的数组名,再访问字符串就\*(arr1+1)+1

1. typedef int a[10] ，定义了a为10个元素的整型数组类型，注意这时a为一个类型，可以定义一个其他的变量， 如 a b;
2. 函数名前没有返回值则默认为int ，max()的函数头决定了max()的返回值必定是int， return 后面的表达式必须转化成int
3. color[k][j]等价于\*(color[k]+j)也可以写成\*(\*(color+k)+j)
4. 数组名a是数组首地址，不能改变(a++)和进行赋值操作，可以用其他的变量（char \*k=a）来引用它再进行这些操作
5. m=a<b<c没有语法错误！！
6. 而n=(-m)++ 有语法错误 因为++运算符的对象必须是变量,不能是常量或表达式
7. ’A’+’B’返回值是int
8. 

因为unsigned char是8bits，范围是0到255.  
所以当a累加到256时，由于位数限制，就变成0了

1. 在内存中，char与unsigned char没有什么不同，都是一个字节 8个bit，唯一的区别是，char的最高位为符号位，因此char能表示-128~127, unsigned char没有符号位，因此能表示0~255（2^8）
2. \x开头是十六进制,\0是八进制
3. char a[]=”ABC\\XYZ\007\xFF” 注意右边为字符串常量而不是字符数组，自带一个\0,所以sizeof还得加1
4. char a=’0’, printf(“%x”,a); 因为’0’的ASCII为48，转为十六进制为30,而且输出时不带这些0x前缀

p是结构体数组名，也就是指针

1. \*p.name[0]=’B’,表面上看起来对，实际上由于.的优先级大于\* 不能实现原有功能
2. 而对于\*(\*p).name=’B’ ,\*p为变量，再由优先级.>\* 取出name，得到实际为&name[0],再用\*解引用，得name[0]=’B’
3. strcmp（“ABC”,”AB”）大于返回一个正整数n>1
4. 注意定义#define For while(x--); 注意后面有个分号！！，就是自循环结束x=-1
5. long int a也是32位，要把a的最高位与1比较（不准用函数），可以a<<31
6. 对于char a[][3]={“ABCD”,”SD”,”BSD”}

\*a是进入第一个字符串，而\*a+1就是指向了BCD的首地址

\*a[2]就是进入第3个字符串，\*a[2]+1是第一个B加1=C

a是数组的首地址为char \*[]类型，加个\*变成char []型，即第一行的数组名是指针了，而a[2]本身是第三行指针为char []类型，再加个\*就是取出char值了，是“BSD”的首地址取值即为B

1. 二维数组中 \*a==a[0] //此时还为指针，\*\*a才为

二维数组指针运算~！！！

1. \*a：以a[0][0]为首元素的一维数组.可带一个索引：(\*a)[i] 即a[0][i]

2. a：以a[0][0]为首元素的二维数组.可带两个索引：a[i][j]

3. a[0]：同\*a ，而\*\*a就是取了首值，和\*a[0]一样

4. a[0][0]：首元素

5. a[1]：以a[1][0]为首元素的一维数组.可带一个索引.

6. &a[1]：以a[1][0]为首元素的二维数组.可带两个索引：(&a[1])[i][j]即a[i+1][j]

要掌握它们的区别,关键是掌握它们各自支持的运算符的不同处理.

对于运算符'+'：a[0][0]执行的是数学上的加法；而其余的由于都是指针,执行的都是地址的加法,而且有的加1是指向同一行中的下一列上的元素(如\*a、a[0]和a[1]),有的加1是指向同一列中的下一行上的元素(如a和&a[1]).

对于运算符'\*'：

a[0][0]执行的是数学上的乘法,如a[0][0] \* 2；

\*a、a[0]和a[1]执行的是取值,如\*\*a + \*a[0] + \*a[1]

a和&a[1]执行的是由原来的二维数组变为一维数组,也可理解为改变指针的类型,其加1的意义就不同了,如a+1和(\*a)+1的不同.

1. int a[4][5]; 对数组元素不正确的引用：\*(a+3)[4] ，因为[]的优先级大于\*
2. 指针还可以取地址&,但是普通int变量不能用\*，如int \*p,q；不能\*q，但能&p
3. 在没有任何标记的时候 整型常数默认都是int型,所以 4962710 就是int型,而不是长整型 long。 要默认Long,需要在结尾加L 于是 0L是合法的长整型数据。
4. 一旦数组定义，数组的首地址是一个常量，不能作++运算，而a+i表示数组第i个元素的地址
5. -->比++高，++比\*高
6. int c,\*s; c=\*s是错的，因为s未赋值，指向不明，取不出值
7. 求7<<1>>2^2的值，由于<<,>>,^都是左结合，就是从左向右就行

转换为((7<<1)>>2)^2 ,0111左移一位再右移两位得0011,再与0010异或得0001，即为1

1. NULL是0，EOF是1
2. 文件打开失败判断if((fp=fopen(filename,”r”)==NULL)
3. 遍历文件while(!feof(fp)){ ch=fgetc(fp)}

浙大总复习题

1. c语言定义的变量，代表内存中的一个连续的存储单元（大小由其类型确定）
2. %o，%x是指输出格式，而不是表示这个数是什么数
3. 018不是合法的八进制，因为八进制为0-7内整数
4. scanf和printf是标准输入输出函数，而不是语句
5. printf默认右对齐
6. c语言可执行语句是指：可以通过编译产生对应的汇编代码和机器码的语句
   1. 些不是呢？比如：括号、预处理语句（行首是#的如#include）、注释 等等。
   2. 像赋值、逻辑运算、算术运算、分支、循环等，都可以转化成相应的汇编代码，他们都是可执行语句
7. 预处理也称预编译，在编译之前为其做准备，主要进行文本替换
8. 表达式不要分号，语句一定要分号，注意题目
9. c语言中简单数据类型为整型、实型、字符型
10. char str[3]={‘k’,’e’,’y’}不是正确字符串，因为没有’\0’,且字符长度不够
11. C语言中四种存储类别auto static extern register，除非标了，否则都是auto。
12. 读文本文件a.dat用’r’而不是’rb’，因为前面已经说是文本而不是二进制
13. 运算符的优先级：算术>关系>赋值
14. 常量后缀有u（U）、f、L（l），LU或UL，前缀0,0x,0b
15. 当p为int变量时，\*p是非法的。因为\*p非法，而
    1. 当p为int \*指针时，么\*&p = p，&\*p = p，都是p。
16. 关系运算符中最小的是==和!=
17. a,b也是一个表达式
18. !x等价于x==0 x等价于x!=0
19. 与或非三者中优先级为!>&&>||
20. 实型常量可以由指数，e大写小写均可，前面为实数，后面必须是十进制整数。
21. strcat(a,b)，把b追加到a后面，a必须有足够空间
22. fgets,fputs用于文本文件，而fread,fwrite用于二进制文件
23. 使指针变为空指针用p=NULL,’\0’,0都行，但是EOF不行，因为EOF=-1
24. b=12345，输出用%4d,会直接输出12345，而不是1234
    1. 如果位数不够的，输出格式是：右对齐，且左边用空格补全；如果位数超出的，直接输出。
25. 字符函数需要ctype.h，而字符串处理函数需要string.h
26. y=int(x)不是合法语句
27. %%d就不能输出值了，只会输出%d，而且后面多出来的变量就没用了
28. 函数不用先定义后使用，而是声明。
29. 同一类型的指针可以相减，为相差的内存地址，如int \*p,\*q;
30. 数组名不可以++a
31. ‘1’<’0’<’2’ 比较表达式是从左向右，因此先比较1和0返回false 0，然后再比较为true ：1
32. 八位二进制表示带符号整数，范围是-128到127，如为无符号整数，是0-255
33. ~3&7 由于~在为运算符中优先级最大，3为0011，按位取反是1100，再与0111取位与，得0100即为4
34. do while 里面必须是个语句，即必须要有分号
35. w[i][4]=\*(w+i)[4] ,w,w+1,,都是指向四个元素的数组指针，和int (\*p)[4]=w一样，所以这时p等价于w。而p++;\*(p+1)并不是值，而需\*(\*p+1)才行
36. 当ch为大写字符将其修改为小写式:ch=(ch>=’A’&&ch<=’Z’)?ch+32:ch
37. x为student型变量，则可以通过sizeof(student)或sizeof(x)计算x字节
38. 读取文件字符 (ch=fgetc(f1))!=EOF
39. 逆序输出小写字母 for(i=0;i<26;i++) putchar(‘z’-i)
40. 011&0x1a，011是八进制转换为001001 ,0x1a转为00011010,与得1000=8
41. 遍历字符串s1和s2， for(;s1!=’\0’;++s1,++s2) ... return \*s1-\*s2;
42. union EXAMPLE {

struct {int x,y; }in;

int a,b; }e;

e.a=1;e.b=2; e.in.x=e.a\*e.b; e.in.y=e.a+e.b;求x，y (4,8)

在union中，所有的共用体成员共用一个空间，并且同一时间只能储存其中一个成员变量的值。

从而在输入e.b=2之后，前面存的a没用了，这个共用体里只存了个2，从而e.in.x=e.a\*e.b=2\*2=4,然后现在共用体里又只存了个4，所以y=e.a+e.b=4+4=8

1. #define F(x) x-2 #define D(x) x\*F(x) printf(“%d,%d”,D(3),D(D(3)));

输出结果是7，-13 ，D(3)=3\*3-2 再代入:3\*3-2\*3\*3-2-2=-13

1. char \*s=”ABC”; while(\*s++) putchar(\*s); ++优先级大于\*，所以是s++， 所以指向了B，输出，再指向C输出
2. p指向结构体数组（含x，y），p++->x，++p->y，前者是输出p[0]的p->x， 再++p指向下一个，而后者在前者基础上指向p[1]，取出y在，并将其加1 再输出。
3. 用typedef定义一个长度为10的整型数组NUM:typedef int NUM[10]（注意是基本类型写中间，数组名写在[]前面）
4. x&(x-1) 可以实现将x二进制最后一个1变为0，如果一直运行到0并统计次数的话，值即为x的二进制中1的个数
   1. 原理:如x的二进制后面为xxx100,减1后变为xxx011,两者与之后为xxx000,即将原来的最后一个1消为0
   2. 如x为xx111，减1后变为xx110，再与之后为xx110,也得证
5. strcat函数：将两个char类型连接。

str[N] 是N个字符串，要把它们全部弄到string中作为一个字符串，方式是 : 先把str[0] strcpy到string，再把str[1]->str[k] strcat 到string下面

1. 把字符串中的十六进制数转换为十进制
   1. 如果为数字：d=d\*16+s[k]-’0’
   2. 如果为a-f：d=d\*16+s[k]-’a’+10
2. 将某年第几天转换为相应的月和日，可以使用两行十二列的数组保存闰年和平年的月份天数，计算leap=year%4==0&&year%100!=0||year%400==0;然后计算日的时候就是从k=1月开始减，yeardays-=daytab[leap][k],如果大于1月就减1月。。。，直到yeardays比某个月份的天数少了，此时的k即为月数，yeardays即为该月的日数
3. 文件输入输出：一个文件读出来经过一个函数处理再存进另一个函数

File \*fp1,\*fp2;

if((fp1=fopen(“input.txt”,”r”))==NULL){

printf(“打开不了”);

exit(0);

}

if((fp2=fopen(“output”,”w”))==NULL){

}

while(!feof(fp1)){

fscanf(fp1,”%d”,&x);

fprintf(fp2,”%d”,sum(x));

}

fclose(fp1);

fclose(fp2);

}

1. 每次读取一个字符（文件）

while((ch=fgetc(f1))!=EOF)

1. 递归一般是倒过来输出的
2. 二维数组每行元素都除以该行绝对值最大的那个元素的绝对值。函数原型 是void f(float \*\*a,int m,int n);

for(i=0;i<m;++i){

max=\*(\*(a+i)); #先取每行第一个元素作为最大值

for(j=0;j<n;++j){

if(fabs(\*(\*(a+i)+j)>fabs(max))

max=fabs(\*(\*(a+i)+j));

}

for(j=0j<n;++j){

\*(\*(a+i)+j)/=max;

}

}

1. 感觉while((ch=fgetc(fp))!=EOF){

putchar(tolower(ch));

fputc(ch,fp2);

}

和 while(!feof(fp)){

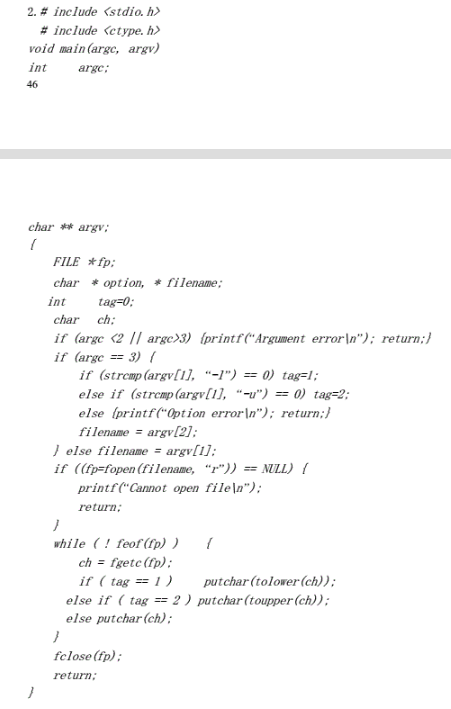
ch=fgetc(fp);

}

1. 输入prog filename原样输出，prog -l filename 小写输出 -u 大写输出

一样

综合题2



牛客网

1.double a = 1.5 ; a++ ;//结果是 a = 2.5（自增、自减的操作符可以是非整型的）

而%只能是整型， /,!=都可以是浮点数

2.交换字符串不能仅仅\*，而是要达到交换str的目的，需要交换两个地址

要改为swap(char \*\*str1, char \*\*str2)，并且传递的时候swap(&str\_a, &str\_b)

1. C++部分兼容C
2. 执行a<<b时，编译器会先将b与31进行and操作，以限制左移的次数小于等于31。b&31=0，则a<<b=1 执行1<<32时，编译器直接执行算术左移的操作。
3. 一个函数定义的返回值是float，则他return char,int,double都可以
4. int \*p[3]，a[3]; 你不能给一个数组名（p）赋值，所以A和C全跪。

\*p等价于\*(p+0)等价于p[0]，而p[0]中存放的是指针变量（int \*），a[0]是int型。B（\*p=a [0]）也不对。

1. .结构体的大小必须是该结构体字节数最大成员的倍数。

struct s1 {

    char ch, \*ptr;

    union {

        short a, b;

        unsigned int c:2, d:1;

    }

    struct s1 \*next;

};

大小为16字节，因为指针和int均为4字节，而ch需字节对齐，也为4字节。

编程技巧

输入double必须是%lf，不能是%f

而输出可以是%f

1. 使用sqrt，fabs要先include<math.h>,而使用字符串函数strcmp,strcpy,strlen,strlwr,strupr(返回值即为变化后字符串的指针)。需要用include<string.h>

一个数加100是完全平方数 ：

设该数为x,则先 //可在0到10万间中的每个x进行遍历

int i=sqrt(x+100); //求出该数的平方根

if(i\*i=x+100) //然后判断这个平方根相乘是不是就是这个数，如果是，说明是完全平方数

1. 判断闰年满足其中一个条件即可（1）能被4整除但不能被100整除（2）能被400整除
2. 求方程的根有公式-b+-根号b2-4ac /2a 表示为两根的话可以先拆成p1=-b/(2.0\*a)

p2=sqrt(b\*b-4\*a\*c)/(2.0\*a) 然后x1=p1+p2, x2=p1-p2 虚根也可以这样搞

4.fabs(a)<=1e-6 可以用于浮点数与0进行比较，ps:1e-6是1乘10的-6次方,fabs是对浮点数取绝对值

在求方程组时，可以先判断fabs(a)<=1e-6，说明a==0即不是二次式，再判断fabs(b\*b-4\*a\*c)<=1e-6说明单根，再判断b\*b-4\*a\*c>1e-6还是<-(1e-6)

1. max函数即为 return x>y?x:y;
2. 汉诺塔：

void hanoi(int n,char a,char b,char c){

void move(char x,char y);//函数内也可声明

if(n==1){

move(one,three); //move函数就是printf一下

}

else{

hanoi(n-1,a,c,b); //先把上面n-1个借助c移至b

move(a,c); //再把最大的移到c

hanoi(n-1,b,a,c); //再把n-1个从b移c

}

}

1. 遍历字符串 ：scanf(“s”,s); while(s[i++]!=’\0’){ 其中使用s[i] }
2. 输出前面补0 ：%02d 就是输出控制为2位，不足前面补0
3. 小写转大写， ch=(‘a’<=ch)&&(ch<=’z’)?ch-32:ch;
4. 判断闰年 (year%4==0&&year%100!=0)||(year%400==0)
5. 判断素数的函数

int prime(int x){

int i,flag=1;

for(i=2;i<(int)sqrt(x);++i){

if(x%i==0){

flag=0;

break;

}

}

return flag;

}

1. 分数统计为等级ABCDE

注意A为100>=x>=90 与其他不同多了个上界100，所以在分数为100时要特殊对待，存进90+的那 部分里。所以做题时就利用100这个特值代，

1. 打印菱形

以4为分界，小于4就输出4-m个空格，大于4就输出4-m%4个空格

输出\*号也是小于4输出2\*m-1个\*，大于4就输出2\*(4-m%4)-1个

13.

典型例题

统计与计数问题

1、连接方面区别

TCP面向连接（如打电话要先拨号建立连接）。

UDP是无连接的，即发送数据之前不需要zd建立连接。

2、安全方面的区

TCP提供可靠的服务，通过TCP连接传送的数据，无差错，不内丢失，不重复，且按序到达。

UDP尽最大努力交付，即不保证可靠交付。

3、传输效率的区别容

TCP传输效率相对较低。

UDP传输效率高，适用于对高速传输和实时性有较高的通信或广播通信。

4、连接对象数量的区别

TCP连接只能是点到点、一对一的。

UDP支持一对一，一对多，多对一和多对多的交互通信。

在CPU中，32位的CPU代表在CPU的单位时间内，

一次可以处理32位数据，

而64位则一次可以处理64位数据。

1、描述⼀下gcc的编译过程？（Linux 下最常用的 C 语言编译器是 GCC （ GNU Compiler Collection ））

gcc编译过程分为4个阶段：预处理、编译、汇编、链接。

预处理：头⽂件包含、宏替换、条件编译、删除注释 C 编译器对各种预处理命令进行处理

编译：主要进⾏词法、语法、语义分析等，检查⽆误后将预处理好的⽂件编译成汇编⽂件翻译”成汇编代码。

汇编：将汇编⽂件转换成 ⼆进制⽬标⽂件

链接：将项⽬中的各个⼆进制⽂件+所需的库+启动代码链接成可执⾏⽂件

2、内存的最⼩存储单位以及内存的最⼩计量单位分别是？

内存的最⼩存储单位为 ⼆进制位， 内存的最⼩计量单位 字节

3、#include<> 与#include ""的区别？

include<>到系统指定⽬录寻找头⽂件，#include ""先到项⽬所在⽬录寻找头⽂件，如果没有找再到系统指定的⽬录下寻找

4、描述⼀下变量的命名规则

变量名有字⺟、数值、下划线组成，但不能以数值开头

5、变量的声明与定义有啥区别？

声明变量 不需要 建⽴存储空间， 变量的定义需要建⽴存储空间

6、谈谈c语⾔中有符号和⽆符号的区别？

有符号：数据的最⾼位为符号位，0表示正数，1表示负数

⽆符号：数据的最⾼位不是符号位，⽽是数据的⼀部分

7、谈谈计算机中补码的意义

统⼀了零的编码

将符号位与其他位统⼀处理将减法运算转换成加法运算

8、谈谈数组的特点

同⼀个数组所有的成员都是相同的数据类型，同时所有的成员在内存中的地址是连续的

9、数组的分类

数组的分类主要是：静态数组、动态数组两类。

静态数组：类似int arr[5];在程序运⾏就确定了数组的⼤⼩，运⾏过程不能更改数组的⼤⼩。

动态数组：主要是在堆区申请的空间，数组的⼤⼩是在程序运⾏过程中确定，可以更改数组的⼤⼩。

10、描述⼀下⼀维数组的不初始化、部分初始化、完全初 始化的不同点

不初始化：如果是局部数组 数组元素的内容随机 如果是全局数组，数组的元素内容为0

部分初始化：未被初始化的部分⾃动补0

完全初始化：如果⼀个数组全部初始化 可以省略元素的个数 数组的⼤⼩由初始化的个数确定

11、谈谈数组名作为类型、作为地址、对数组名取地址的 区别？

数组名作为类型：代表的是整个数组的⼤⼩

数组名作为地址：代表的是数组⾸元素的地址

对数组名取地址：代表的是数组的⾸地址

12、谈谈你对⼆维数组在物理上以及逻辑上的数组维度理解

⼆维数组在逻辑上是⼆维的，在物理上是⼀维的

13、描述⼀下函数的定义与函数的声明的区别

函数定义：是指对函数功能的确⽴，包括指定函数名、函数类型、形参及其类型、函数体等，它是⼀个完整的、独⽴的函数单位。

函数的声明：是把函数的名字、函数类型以及形参的个数、类型和顺序(注意，不包括函数体)通知编译系统，以便在对包含函数调⽤的语句进⾏编译时，据此对其进⾏对照检查（例如函数名是否正确，实参与形参的类型和个数是否⼀致）

14、描述⼀下指针与指针变量的区别

指针：没存中每⼀个字节都会分配⼀个32位或64位的编号，这个编号就是地址, ⽽指针就是内存单元的编号。

指针变量：本质是变量 只是该变量存放的是空间的地址编号

15、描述⼀下32位或64位平台下指针的⼤⼩

32位平台：任意类型的指针⼤⼩为4字节

64位平台：任意类型的指针⼤⼩为8字节

16、描述⼀下指针数组的概念

指针数组本质是数组，只是数组的每个元素是⼀个指针（地址）

17、描述⼀下普通局部变量、普通全局变量、静态局部变量、静态全局变量的区别

普通局部变量： 存在栈区、不初始化内容随机、只在定义所在的复合语句中有效、符合语句结束变量空间释放

普通全局变量 ：存在全局区、不初始化内容为0、进程结束空间才被释放，能被当前源⽂件或其他源⽂件使⽤，只是其他源⽂件使⽤的时候，记得使⽤extern修饰

静态局部变量： 存在全局区、不初始化内容为0、整个进程结束空间才被释放，只能在定义所在的复合语句中有效

静态全局变量 ：存在全局区、不初始化内容为0、整个进程结束空间才被释放，只能被当前源⽂件使⽤

18、描述⼀下内存分区

程序在运⾏前：分为代码区、BSS段（未初始化数据区）、data段(初始化数据区)

程序在运⾏后：堆区、栈区、全局区（静态区）、⽂字常量区、代码区

19、在使⽤realloc给已分配的堆区空间追加空间时需要注意啥？

记得⽤指针变量保存realloc的返回值

20、结构体与共⽤体的区别是啥

结构体中的成员拥有独⽴的空间，共⽤体的成员共享同⼀块空间，但是每个共⽤体成员能访问共⽤区的空间⼤⼩是由成员⾃身的类型决定

21、谈谈⽂件的分类21、谈谈⽂件的分类

⽂件分为⼆进制和⽂本⽂件

⼆进制⽂件基于值编码，需要根据具体的应⽤才能知道某个值具体的含义

⽂本⽂件基于字符编码，⼀个字节⼀个意思，可以通过记事本打开

22、⽂件缓冲区刷新⽅式有⼏种

⾏刷新、满刷新、强制刷新、关闭刷新

23、哪些情况下会出现野指针

指针变量未初始化、指针释放后未为置空、指针操作超越变量作⽤域

24、如何理解指针作为函数参数的输⼊和输出特性

输⼊特性：主调函数分配空间 背调函数使⽤该空间

输出特性：被调⽤分配空间 主调函数使⽤该空间

25、如何理解结构体的浅拷⻉与深拷⻉

当结构体中有指针成员的时候容易出现浅拷⻉与深拷⻉的问题。

浅拷⻉就是，两个结构体变量的指针成员指向同⼀块堆区空间，在各个结构体变量释放的时候会出现多次释放同⼀段堆区空间

深拷⻉就是，让两个结构体变量的指针成员分别指向不同的堆区空间，只是空间内容拷⻉⼀份，这样在各个结构体变量释放的时候就不会出现多次释放同⼀段堆区空间的问题

26、描述⼀下结构体对⻬规则

1. 数组成员对⻬规则。第⼀个数组成员应该放在offffset为0的地⽅，以后每个数组成员应该放在offffset 为min（当前成员的⼤⼩，#pargama pack(n)）整数倍的地⽅开始（⽐如int在32位机器为４字节，#pargama pack(2)，那么从2的倍数地⽅开始存储）。

2. 结构体总的⼤⼩，也就是sizeof的结果，必须是min（结构体内部最⼤成员，#pargama pack(n)）的整数倍，不⾜要补⻬。

3. 结构体做为成员的对⻬规则。如果⼀个结构体B⾥嵌套另⼀个结构体A,还是以最⼤成员类型的⼤⼩对⻬，但是结构体A的起点为A内部最⼤成员的整数倍的地⽅。（struct B⾥存有struct A，A⾥有 char，int，double等成员，那A应该从8的整数倍开始存储。），结构体A中的成员的对⻬规则仍 满⾜原则1、原则2。

27、啥叫宏函数以及作⽤

在项⽬中，经常把⼀些短⼩⽽⼜频繁使⽤的函数写成宏函数，这是由于宏函数没有普通函数参数压栈、跳转、返回等的开销，可以调⾼程序的效率。 宏通过使⽤参数，可以创建外形和作⽤都与函数类似地类 函数宏(function-like macro). 宏的参数也⽤圆括号括起来，来保证宏函数的完整性。

28、如何理解库函数

库是已经写好的、成熟的、可复⽤的代码。每个程序都需要依赖很多底层库，不可能每个⼈的代码从零开始编写代码，因此库的存在具有⾮常重要的意义。 在我们的开发的应⽤中经常有⼀些公共代码是需要反复使⽤的，就把这些代码编译为库⽂件。 库可以简单看成⼀组⽬标⽂件的集合，将这些⽬标⽂件经过压缩打包之后形成的⼀个⽂件。像在Windows这样的平台上，最常⽤的c语⾔库是由集成按开发环境所附带的运⾏库，这些库⼀般由编译⼚商提供

1、输出9\*9口诀。共9行9列，i控制行，j控制列。

#include "stdio.h"

int main(){

int i,j,result;

for (i=1;i<10;i++)

{

for(j=1;j<10;j++)

{

result=i\*j;

printf("%d\*%d=%-3d",i,j,result);/\*-3d表示左对齐，占3位\*/

}

printf("\n");/\*每一行后换行\*/

}}

2、古典问题：有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？ 兔子的规律为数列1,1,2,3,5,8,13,21....

#include "stdio.h"int main(){

long f1,f2;

int i;

f1=f2=1;

for(i=1;i<=20;i++)

{

printf("%12ld %12ld",f1,f2);

if(i%2==0) printf("\n");/\*控制输出，每行四个\*/

f1=f1+f2; /\*前两个月加起来赋值给第三个月\*/

f2=f1+f2; /\*前两个月加起来赋值给第三个月\*/

}}

3、判断101-200之间有多少个素数，并输出所有素数及素数的个数。 程序分析：判断素数的方法：用一个数分别去除2到sqrt(这个数)，如果能被整除，则表明此数不是素数，反之是素数。

#include "math.h"#include "stdio.h"int main(){

int m,i,k,h=0,leap=1;

printf("\n");

for(m=101;m<=200;m++)

{

k=sqrt(m+1);

for(i=2;i<=k;i++){

if(m%i==0)

{

leap=0;

break;

}

}

if(leap) /\*内循环结束后，leap依然为1，则m是素数\*/

{

printf("%-4d",m);h++;

if(h%10==0)

printf("\n");

}

leap=1;

}

printf("\nThe total is %d",h);}

4、一个数如果恰好等于它的因子之和，这个数就称为"完数"。例如6=1＋2＋3.编程找出1000以内的所有完数。

#include "stdio.h"int main(){

static int k[10];

int i,j,n,s;

for(j=2;j<1000;j++)

{

n=-1;

s=j;

for(i=1;i<j;i++)

{

if((j%i)==0)

{

n++;

s=s-i;

k[n]=i;

}

}

if(s==0)

{

printf("%d is a wanshu: ",j);

for(i=0;i<n;i++)

printf("%d,",k[i]);

printf("%d\n",k[n]);

}

}}

5、下面程序的功能是将一个4×4的数组进行逆时针旋转90度后输出，要求原始数组的数据随机输入，新数组以4行4列的方式输出，请在空白处完善程序。

*//原矩阵的第i行变为了旋转矩阵的第i列;*

*//原矩阵的第j列变为了旋转矩阵的倒数第j行*

#include <stdio.h>#include <string.h>void main(){

int a[4][4],b[4][4],i,j; /\*a存放原始数组数据，b存放旋转后数组数据\*/

printf("input 16 numbers: ");

/\*输入一组数据存放到数组a中，然后旋转存放到b数组中\*/

for(i=0;i<4;i++)

for(j=0;j<4;j++)

{

scanf("%d",&a[i][j]);

b[3-j][i]=a[i][j];

}

printf("array b:\n");

for(i=0;i<4;i++)

{

for(j=0;j<4;j++)

printf("%6d",b[i][j]);printf("\n");

}}

6、编程打印直角杨辉三角形

#include <stdio.h>

void main(){

int i,j,a[6][6];

for(i=0;i<=5;i++)

{

a[i][i]=1;

a[i][0]=1;

}

for(i=2;i<=5;i++)

for(j=1;j<=i-1;j++)

a[i][j]=a[i-1][j]+a[i-1][j-1];

for(i=0;i<=5;i++)

{

for(j=0;j<=i;j++)

printf("%4d",a[i][j]);

printf("\n");

}}

7、通过键盘输入3名学生4门课程的成绩，分别求每个学生的平均成绩和每门课程的平均成绩。要求所有成绩均放入一个4行5列的数组中，输入时同一人数据间用空格,不同人用回车其中最后一列和最后一行分别放每个学生的平均成绩、每门课程的平均成绩及班级总平均分。

#include <stdio.h>#include <stdlib.h>void main(){

float a[4][5],sum1,sum2;

int i,j;

for(i=0;i<3;i++)

for(j=0;j<4;j++)

scanf("%f",&a[i][j]);

for(i=0;i<3;i++)

{

sum1=0;

for(j=0;j<4;j++)

sum1+=a[i][j];

a[i][4]=sum1/4;

}

for(j=0;j<5;j++)

{

sum2=0;

for(i=0;i<3;i++)

sum2+=a[i][j];

a[3][j]=sum2/3;

}

for(i=0;i<4;i++)

{

for(j=0;j<5;j++)

printf("%6.2f",a[i][j]);

printf("\n");

}}

8、完善程序，实现将输入的字符串反序输出， 如输入windows 输出swodniw。

#include <string.h>#include <stdio.h>void main(){

char c[200],c1;

int i,j,k;

printf("Enter a string: ");

scanf("%s",c);

k=strlen(c);

for (i=0,j=k-1;i<k/2;i++,j--)

{

c1=c[i];

c[i]=c[j];

c[j]=c1;

}

printf("%s\n",c);}

指针法：

#include <string.h>#include <stdio.h>void invert(char \*s){

int i,j,k;

char t;

k=strlen(s);

for(i=0,j=k-1;i<k/2;i++,j--)

{

t=\*(s+i);

\*(s+i)=\*(s+j);

\*(s+j)=t;

}}void main(){

FILE \*fp;

char str[200],\*p,i,j;

if((fp=fopen("p9\_2.out","w"))==NULL)

{

printf("cannot open the file\n");

exit(0);

}

printf("input str:\n");

gets(str);

printf("\n%s",str);

fprintf(fp,"%s",str);

invert(str);

printf("\n%s",str);

fprintf(fp,"\n%s",str);

fclose(fp);}

9、下面程序的功能是从字符数组s中删除存放在c中的字符。

#include <stdio.h>void main(){

char s[80],c;

int j,k;

printf("\nEnter a string: ");

gets(s);

printf("\nEnter a character: ");

c=getchar( );

for(j=k=0;s[j]!= '\0';j++)

if(s[j]!=c) //如果等于c就不赋，变相删除

s[k++]=s[j];

s[k]= '\0';

printf("\n%s",s);}

10、编写一个void sort(int \*x,int n)实现将x数组中的n个数据从大到小 排序。n及数组元素在主函数中输入。将结果显示在屏幕上并输出到文件p9\_1.out中

#include<stdio.h>void sort(int \*x,int n){

int i,j,k,t;

for(i=0;i<n-1;i++)

{

k=i;

for(j=i+1;j<n;j++)

if(x[j]>x[k]) k=j;

if(k!=i)

{

t=x[i];

x[i]=x[k];

x[k]=t;

}

}}void main(){

FILE \*fp;

int \*p,i,a[10];

fp=fopen("p9\_1.out","w");

p=a;

printf("Input 10 numbers:");

for(i=0;i<10;i++)

scanf("%d",p++);

p=a;

sort(p,10);

for(;p<a+10;p++)

{

printf("%d ",\*p);

fprintf(fp,"%d ",\*p);

}

system("pause");

fclose(fp);}

11、已知数组a中的元素已按由小到大顺序排列，以下程序的功能是将输入的一个数插入数组a中，插入后，数组a中的元素仍然由小到大顺序排列

#include<stdio.h>void main(){

int a[10]={0,12,17,20,25,28,30}; /\*a[0]为工作单元，从a[1]开始存放数据\*/

int x , i, j=6; /\*j为元素个数\*/

printf("Enter a number: ");

scanf("%d",&x);

a[0]=x;

i=j; /\*从最后一个单元开始\*/

while(a[i]>x)

{

a[i+1]=a[i];

i--;

} /\*将比x大的数往后移动一个位置\*/

a[++i]=x;

j++; /\*插入x后元素总个数增加\*/

for(i=1;i<=j;i++)

printf("%8d",a[i]);

printf("\n");}

12、编写函数replace(char \*s,char c1,char c2)实现将s所指向的字符串中所有字符c1用c2替换，字符串、字符c1和c2均在主函数中输入，将原始字符串和替换后的字符串显示在屏幕上，并输出到文件p10\_2.out中

#include<stdio.h>void replace(char \*s,char c1,char c2){

while(\*s!='\0')

{

if (\*s==c1)

\*s=c2;

s++;

}}void main(){

FILE \*fp;

char str[100],a,b;

if((fp=fopen("p10\_2.out","w"))==NULL)

{ printf("cannot open the file\n");

exit(0); }

printf("Enter a string:\n");

gets(str);

printf("Enter a&&b:\n");

scanf("%c,%c",&a,&b);

printf("%s\n",str);

fprintf(fp,"%s\n",str);

replace(str,a,b);

printf("The new string is----%s\n",str);

fprintf(fp,"The new string is----%s\n",str);

fclose(fp);}

13、在一个字串s1中查找一子串s2，若存在则返回子串在主串中的起始位置，不存在则返回-1。

#include<stdio.h>void replace(char \*s,char c1,char c2)void main(){

char s1[6]="thisis";

char s2[5]="is";

printf("%d\n",search(s1,s2));

system("pause");}int search(char s1[],char s2[]){

int i=0,j,len=strlen(s2);

while(s1[i])

{

for(j=0;j<len;j++)

if(s1[i+j]!=s2[j]) #j是子串下标，i是母串下标，i+j和j对比，不一样即退出再找

break;

if(j>=len) #如果大于len说明找到了，字符均对应，记录i为开头标

return i;

else

i++;

}

return -1;}

14、用指针变量输出结构体数组元素。

#include<stdio.h>struct student{int num;char \*name;char sex;int age;}

stu[5]={{1001,"lihua",'F',18},{1002,"liuxing",'M',19},{1003,"huangke",'F',19},{1004,"fengshou",'F',19},{1005,"Wangming",'M',18}};void main(){

int i;

struct student \*ps;

printf("Num \tName\t\t\tSex\tAge\t\n");

/\*用指针变量输出结构体数组元素。\*/

for(ps=stu;ps<stu+5;ps++)

printf("%d\t%-10s\t\t%c\t%d\t\n",ps->num,ps->name,ps->sex,ps->age);

/\*用数组下标法输出结构体数组元素学号和年龄。\*/

for(i=0;i<5;i++)

printf("%d\t%d\t\n",stu[i].num,stu[i].age);}

15、建立一个有三个结点的简单链表：

#include<stdio.h>#define NULL 0struct student{

int num;

char \*name;

int age ;

struct student \*next;};void main(){

struct student a,b,c,\*head,\*p;

a.num=1001; a.name="lihua"; a.age=18; /\* 对结点成员进行赋值 \*/

b.num=1002; b.name="liuxing"; b.age=19;

c.num=1003; c.name="huangke"; c.age=18;

head=&a; /\* 建立链表，a为头结点 \*/

a.next=&b;

b.next=&c;

c.next=NULL;

p=head; /\* 输出链表 \*/

do{

printf("%5d,%s,%3d\n",p->num,p->name,p->age);

p=p->next;

}while(p!=NULL);}

16、输入一个字符串，判断其是否为回文。回文字符串是指从左到右读和从右到左读完全相同的字符串。

#include <stdio.h>#include <string.h>

void main(){

char s[100];

int i,j,n;

printf("输入字符串：\n");

gets(s);

n=strlen(s);

for(i=0,j=n-1;i<j;i++,j--)

if(s[i]!=s[j])

break;

if(i>=j)

printf("是回文串\n");

else

printf("不是回文串\n");}

17、冒泡排序，从小到大，排序后结果输出到屏幕及文件myf2.out

#include<stdio.h>void fun(int a[],int n){

int i,j,t;

for(i=0;i<=n-1;i++)

for(j=0;j<i;j++)

if(a[j]>a[j+1])

{

t=a[j];a[j]=a[j+1];a[j+1]=t;

}}void main(){

int a[10]={12,45,7,8,96,4,10,48,2,46},n=10,i;

FILE \*f;

if((f=fopen("myf2.out","w"))==NULL)

printf("open file myf2.out failed!\n");

fun(a,10);

for(i=0;i<10;i++)

{

printf("%4d",a[i]);

fprintf(f,"%4d",a[i]);

}

fclose(f);}

18、编写函数countpi，利用公式

计算π的近似值，当某一项的值小于10-5时，认为达到精度要求，请完善函数。将结果显示在屏幕上并输出到文件p7\_3.out中。

#include<stdio.h>double countpi(double eps) /\*eps为允许误差\*/{

int m=1;

double temp=1.0,s=0;

while(temp>=eps)

{

s+=temp;

temp=temp\*m/(2\*m+1);

m++;

}

return(2\*s);}void main(){

FILE \*fp;

double eps=1e-5,pi;

if((fp=fopen("p7\_3.out","w"))==NULL)

{

printf("cannot open the file\n");

exit(0);

}

pi= countpi(eps);

printf("pi=%lf\n",pi);

fprintf(fp,"pi=%lf\n",pi);

fclose(fp);}

1、变量的声明和定义有什么区别

为变量分配地址和存储空间的称为定义，不分配地址的称为声明。一个变量可以在多个地方声明，但是只在一个地方定义。加入 extern 修饰的是变量的声明，说明此变量将在文件以外或在文件后面部分定义。说明：很多时候一个变量，只是声明不分配内存空间，直到具体使用时才初始化，分配内存空间，如外部变量。

2、写出 bool 、int、 float、指针变量与“零值”比较的 if 语句

bool 型数据： if( flag ) {

A; } else {

B； } int 型数据： if( 0 != flag ) {

A; } else {

B； }

指针型数： if( NULL == flag ) {

A; } else {

B； } float 型数据： if ( ( flag >= NORM ) && ( flag <= NORM ) ) {

A； }

注意：应特别注意在 int、指针型变量和“零值”比较的时候，把“零值”放在左边，这样当把“==” 误写成“=”时，编译器可以报错，否则这种逻辑错误不容易发现，并且可能导致很严重的后果。

3、sizeof 和 strlen 的区别

sizeof 和 strlen 有以下区别：

1. sizeof 是一个操作符，strlen 是库函数。

2. sizeof 的参数可以是数据的类型，也可以是变量，而 strlen 只能以结尾为‘\0‘的字符串作参数。

3. 编译器在编译时就计算出了 sizeof 的结果。而 strlen 函数必须在运行时才能计算出来。并且 sizeof 计算的是数据类型占内存的大小，而 strlen 计算的是字符串实际的长度。

4. 数组做 sizeof 的参数不退化，传递给 strlen 就退化为指针了。

注意：有些是操作符看起来像是函数，而有些函数名看起来又像操作符，这类容易混淆的名称一定要加以区分，否则遇到数组名这类特殊数据类型作参数时就很容易出错。最容易混淆为函数的操作符就是 sizeof。

4、C 语言的关键字 static 和 C++ 的关键字 static 有什么区别

在 C 中 static 用来修饰局部静态变量和外部静态变量、函数。而 C++中除了上述功能外，还用来定义类的成员变量和函数。即静态成员和静态成员函数。

注意：编程时 static 的记忆性，和全局性的特点可以让在不同时期调用的函数进行通信，传递信息，而 C++的静态成员则可以在多个对象实例间进行通信，传递信息。

5、Ｃ中的 malloc 和Ｃ＋＋中的 new 有什么区别

malloc 和 new 有以下不同：

（1） new、delete 是操作符，可以重载，只能在 C++中使用。

（2） malloc、free 是函数，可以覆盖，C、C++中都可以使用。

（3） new 可以调用对象的构造函数，对应的 delete 调用相应的析构函数。

（4） malloc 仅仅分配内存，free 仅仅回收内存，并不执行构造和析构函数

（5） new、delete 返回的是某种数据类型指针，malloc、free 返回的是 void 指针。

注意：malloc 申请的内存空间要用 free 释放，而 new 申请的内存空间要用 delete 释放，不要混用。  
因为两者实现的机理不同。

6、写一个“标准”宏 MIN

#define min(a,b)((a)<=(b)?(a):(b))

注意：在调用时一定要注意这个宏定义的副作用，如下调用：

((++\*p)<=(x)?(++\*p):(x)。

p 指针就自加了两次，违背了 MIN 的本意。

7、一个指针可以是 volatile 吗

可以，因为指针和普通变量一样，有时也有变化程序的不可控性。常见例：子中断服务子程序修改一个指向一个 buffer 的指针时，必须用 volatile 来修饰这个指针。

说明：指针是一种普通的变量，从访问上没有什么不同于其他变量的特性。其保存的数值是个整型数据，和整型变量不同的是，这个整型数据指向的是一段内存地址。

8、a 和&a 有什么区别

#include<stdio.h>

void main( void )

{

int a[5]={1,2,3,4,5};

int \*ptr=(int \*)(&a+1);

printf("%d,%d",\*(a+1),\*(ptr-1));

return;

}

输出结果：2，5。

9、简述 C、C++程序编译的内存分配情况

C、C++中内存分配方式可以分为三种：

（1） 从静态存储区域分配：

内存在程序编译时就已经分配好，这块内存在程序的整个运行期间都存在。速度快、不容易出错，因为有系统会善后。例如全局变量，static 变量等。

（2） 在栈上分配：

在执行函数时，函数内局部变量的存储单元都在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，效率很高，但是分配的内存容量有限。

（3） 从堆上分配：

即动态内存分配。程序在运行的时候用 malloc 或 new 申请任意大小的内存，程序员自己负责在何时用 free 或 delete 释放内存。动态内存的生存期由程序员决定，使用非常灵活。如果在堆上分配了空间，就有责任回收它，否则运行的程序会出现内存泄漏，另外频繁地分配和释放不同大小的堆空间将会产生堆内碎块。

一个 C、C++程序编译时内存分为 5 大存储区：堆区、栈区、全局区、文字常量区、程序代码区。

10、简述 strcpy、sprintf 与 memcpy 的区别

三者主要有以下不同之处：

（1） 操作对象不同，strcpy 的两个操作对象均为字符串，sprintf 的操作源对象可以是多种数据类型，目的操作对象是字符串，memcpy 的两个对象就是两个任意可操作的内存地址，并不限于何种数据类型。

（2） 执行效率不同，memcpy 最高，strcpy 次之，sprintf 的效率最低。

（3） 实现功能不同，strcpy 主要实现字符串变量间的拷贝，sprintf 主要实现其他数据类型格式到字符串的转化，memcpy 主要是内存块间的拷贝。

说明：strcpy、sprintf 与 memcpy 都可以实现拷贝的功能，但是针对的对象不同，根据实际需求，来选择合适的函数实现拷贝功能。

11、设置地址为 0x67a9 的整型变量的值为 0xaa66

int \*ptr; ptr = (int \*)0x67a9; \*ptr = 0xaa66;

12、谈谈你对编程规范的理解或认识

编程规范可总结为：程序的可行性，可读性、可移植性以及可测试性。

说明：这是编程规范的总纲目，面试者不一定要去背诵上面给出的那几个例子，应该去理解这几个例子说明的问题，想一想，自己如何解决可行性、可读性、可移植性以及可测试性这几个问题，结合以上几个例子和自己平时的编程习惯来回答这个问题。

13、short i = 0; i = i + 1L；这两句有错吗

代码一是错的，代码二是正确的。

说明：在数据安全的情况下大类型的数据向小类型的数据转换一定要显示的强制类型转换。

14、&&和&、||和|有什么区别

（1）&和|对操作数进行求值运算，&&和||只是判断逻辑关系。（2）&&和||在在判断左侧操作数就能确定结果的情况下就不再对右侧操作数求值。

注意：在编程的时候有些时候将&&或||替换成&或|没有出错，但是其逻辑是错误的，可能会导致不可预想的后果（比如当两个操作数一个是 1 另一个是 2 时。

15、typedef 和 define 有什么区别

（1） 用法不同：typedef 用来定义一种数据类型的别名，增强程序的可读性。define 主要用来定义常量，以及书写复杂使用频繁的宏。

（2） 执行时间不同：typedef 是编译过程的一部分，有类型检查的功能。define 是宏定义，是预编译的部分，其发生在编译之前，只是简单的进行字符串的替换，不进行类型的检查。

（3） 作用域不同：typedef 有作用域限定。define 不受作用域约束，只要是在 define 声明后的引用都是正确的。

（4） 对指针的操作不同：typedef 和 define 定义的指针时有很大的区别。

注意：typedef 定义是语句，因为句尾要加上分号。而 define 不是语句，千万不能在句尾加分号。

16、链表和数组有什么区别

数组和链表有以下几点不同：

（1） 存储形式：数组是一块连续的空间，声明时就要确定长度。链表是一块可不连续的动态空间，长度可变，每个结点要保存相邻结点指针。

（2） 数据查找：数组的线性查找速度快，查找操作直接使用偏移地址。链表需要按顺序检索结点，效率低。

（3） 数据插入或删除：链表可以快速插入和删除结点，而数组则可能需要大量数据移动。

（4） 越界问题：链表不存在越界问题，数组有越界问题。

说明：在选择数组或链表数据结构时，一定要根据实际需要进行选择。数组便于查询，链表便于插入删除。数组节省空间但是长度固定，链表虽然变长但是占了更多的存储空间。

17、如何避免“野指针”

“野指针”产生原因及解决办法如下：

（1） 指针变量声明时没有被初始化。解决办法：指针声明时初始化，可以是具体的地址值，也可让它指向 NULL。

（2） 指针 p 被 free 或者 delete 之后，没有置为 NULL。解决办法：指针指向的内存空间被释放后指针应该指向 NULL。

（3） 指针操作超越了变量的作用范围。解决办法：在变量的作用域结束前释放掉变量的地址空间并且让指针指向 NULL。

注意：“野指针”的解决方法也是编程规范的基本原则，平时使用指针时一定要避免产生“野指针”，在使用指针前一定要检验指针的合法性。

18、怎样把一个单链表反序

（1） 反转一个链表。循环算法。

List reverse(List n) {

if(!n) //判断链表是否为空，为空即退出。

{

return n;

}

list cur = n.next; //保存头结点的下个结点

list pre = n;

list tmp; //保存头结点

pre.next = null; //头结点的指针指空，转换后变尾结点

while ( NULL != cur.next ) //循环直到 cur.next 为空

{

tmp = cur;

}

tmp.next = pre;

pre = tmp;

cur = cur.next;

}

return tmp; //f 返回头指针 }

（2） 反转一个链表。递归算法。

List \*reverse( List \*oldList, List \*newHead = NULL )

{

List \*next = oldList-> next; //记录上次翻转后的链表

oldList-> next = newHead; //将当前结点插入到翻转后链表的开头

newHead = oldList; //递归处理剩余的链表

return ( next==NULL )? newHead: reverse( t, newHead );

}

说明：循环算法就是移动过程，比较好理解和想到。递归算法的设计虽有一点难度，但是理解了循环算法，再设计递归算法就简单多了。

19、简述队列和栈的异同

队列和栈都是线性存储结构，但是两者的插入和删除数据的操作不同，队列是“先进先出”，栈是

“后进先出”。

注意：区别栈区和堆区。堆区的存取是“顺序随意”，而栈区是“后进先出”。栈由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。堆一般由程序员分配释放， 若程序员不释放，程序结束时可能由 OS 回收。分配方式类似于链表。

它与本题中的堆和栈是两回事。堆栈只是一种数据结构，而堆区和栈区是程序的不同内存存储区域。

20、能否用两个栈实现一个队列的功能

说明：用两个栈能够实现一个队列的功能，那用两个队列能否实现一个队列的功能呢？结果是否定的，因为栈是先进后出，将两个栈连在一起，就是先进先出。而队列是现先进先出，无论多少个连在一起都是先进先出，而无法实现先进后出。

//结点结构体：typedef struct node{

int data;

node \*next;}node, \*LinkStack;//创建空栈：

LinkStack CreateNULLStack(LinkStack &S){

S = (LinkStack)malloc(sizeof(node)); //申请新结点

if (NULL == S)

{

printf("Fail to malloc a new node.\n");

9

return NULL;

}

S->data = 0; //初始化新结点

S->next = NULL;

return S;}//栈的插入函数：

LinkStack Push(LinkStack &S, int data){

if (NULL == S) //检验栈

{

printf("There no node in stack!");

return NULL;

}

LinkStack p = NULL;

p = (LinkStack)malloc(sizeof(node)); //申请新结点

if (NULL == p)

{

printf("Fail to malloc a new node.\n");

return S;

}

if (NULL == S->next)

{

p->next = NULL;

}

else

{

p->next = S->next;

}

p->data = data; //初始化新结点

S->next = p; //插入新结点

return S;}//出栈函数：

node Pop(LinkStack &S){

node temp;

temp.data = 0;

temp.next = NULL;

if (NULL == S) //检验栈

{

printf("There no node in stack!");

return temp;

}

temp = \*S;

10

if (S->next == NULL)

{

printf("The stack is NULL,can't pop!\n");

return temp;

}

LinkStack p = S->next; //节点出栈

S->next = S->next->next;

temp = \*p;

free(p);

p = NULL;

return temp;}//双栈实现队列的入队函数：

LinkStack StackToQueuPush(LinkStack &S, int data){

node n;

LinkStack S1 = NULL;

CreateNULLStack(S1); //创建空栈

while (NULL != S->next) //S 出栈入 S1

{

n = Pop(S);

Push(S1, n.data);

}

Push(S1, data); //新结点入栈

while (NULL != S1->next) //S1 出栈入 S

{

n = Pop(S1);

Push(S, n.data);

}}

21、计算一颗二叉树的深度

深度的计算函数：

int depth(BiTree T) {

if(!T) return 0; //判断当前结点是否为叶子结点

int d1= depth(T->lchild); //求当前结点的左孩子树的深度

int d2= depth(T->rchild); //求当前结点的右孩子树的深度

} return (d1>d2?d1:d2)+1;

。Java语言的可移植性正是建立在Java虚拟机的基础上