华南理工大学计算机考研复试，在 34 所 408 联考高校中比较特别，除去各个学校都存在的面试环节，对比于其它科目传统笔试，华南理工大学更加倾向于使用 C#窗体的程序制作来进行面试复试，这与部分学校使用传统的 C 语言编写算法不同。华南理工大学计算机考研复试的工程量、深度堪比一个小型的软件项目。

网上一直对此资料甚小，有也是部分同学的回忆版，草草地叙述一下题目，也写写简单编程思想。并没有一份详尽的资料。这对于部分没有接触过 C#窗体的同学，有点痛苦。

下面讲解如何利用 C#、SQL Server2005、VS2010 完成华南理工大学计算机考研复试，以最新的 2015 年 3 月份的真题为例。

首先，第一大题，10 分，送分的题目：

1、创建文件夹“d:\研究生复试\[你的中文姓名]\” 例: 张三，应创建“d:\研究生复试\张三”文件夹。 所有文档和答案都放在这个文件夹中。

2、在文件夹中建立一个 readme 文件（.txt 或.doc 均可），以说明所用的软件工具。

3、设计文档，文件名为 Info.doc，**数据库**连接说明：如用户名，密码，ODBC/JDBC 等数据源配置等（数据库连接方式及配置参数）如果必要，可以说明运行方式和相关参数 如果功能不能通过运行，则给出相应的源代码。

4、在你的目录中建立 SOURCE 目录，系统源文件放在该目录下

5、考完后请不要关机，人离开就可以了。

答：

在做题之前，自己先检测机器有没有 Visual Studio 2005/2008/2010、开始菜单，找到 Microsoft SQL Server 2005 这个文件夹中的 SQL Server Management Studio Express，没有这里工具，马上举手，换机器。毕竟做题时间非常紧，以下的所有内容不难，对于我这样平时写程序的人来说，甚至觉得就是无聊，但是如此之多的需求，我用了远超于 1 个半小时完成，各位考生好自为之。

第 1 题，建立文件夹就不展示了，相信每一个用过计算机的同学都会，第 5 题自己记得别关机，相信没有考生会这么傻吧？

第 4 题，一会在第三大题搞 C#的在一开始创建工程时候注意好路径的选择~。

第 2、3 题的说明文档留在最后那么的 10-20 分钟写，做完没做完都好，对整个工程做一个好好的总结，同时给改卷老师好好说明你做了什么，这里暂时不理会，没写完程序写毛线！

直接第二大题，数据库设计，35 分

一、建立数据库，并建立以下各表 一个员工可以到多个不同公司上班。

员工关系表 EMPLOYEE（员工号 EmpNo，员工姓名 EmpName，性别 EmpSex，年龄 EmpAge）

EmpNo	EmpName	EmpSex	EmpAge
E01	张三	女	32
E02	李四	男	28
E03	王五	女	42
E04	赵六	男	37
E05	陈七	男	51

工作关系表 WORKS（EmpNo 员工号，CmpNo 公司号，Salary 薪水）

EmpNo	CmpNo	Salary
E01	C01	3000
E01	C02	4000
E02	C02	5000
E02	C03	2500
E03	C01	3500
E04	C02	3000
E05	C03	2000

公司关系表 COMPANY（CmpNo 公司号，CmpName 公司名）

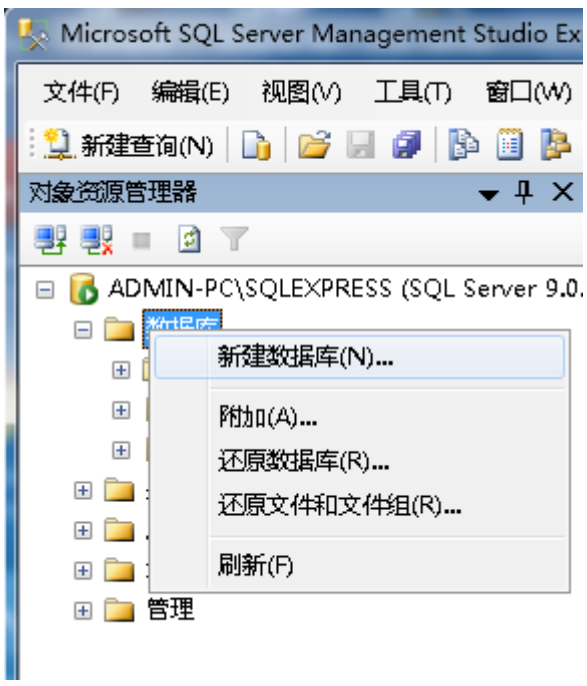
	CmpNo	CmpName
▶	C01	阳光科技
	C02	晨光科技
	C03	未来科技

要求：

1、在数据库中根据上述表的定义创建上述数据库，同时需建立相应的约束关系

答：（1）首先，你在开始菜单，找到 Microsoft SQL Server 2005 这个文件夹中的 SQL Server Management Studio Express 打开它，直接使用如下图的 Windows 身份验证登录。

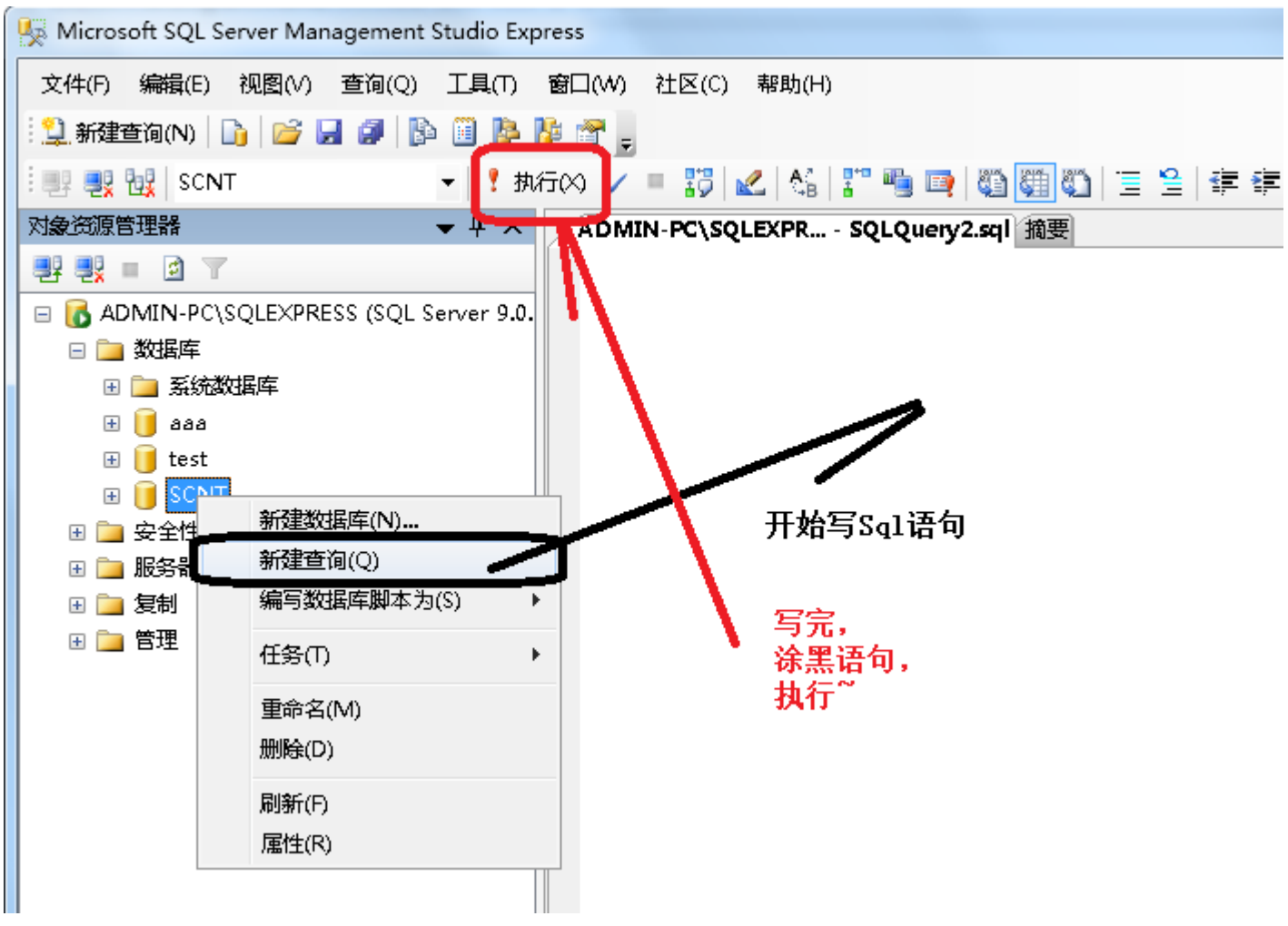
（2）之后，如下图，右击“数据库”，新建一个数据库



（3）如下图，这里以新建一个 SCNT（华南理工大学的英文简称）数据库为例，虽然平时做工程没有人会修改数据库路径的，直接让系统管理的，但是，现在是考试，那你就将日志、数据两个数据库文件都改到你在第 1 大题、第 1 小题，一上来就建立的文件夹。其余所有参数不改，也没时间改，点“确定”。

（4）之后，我们在 SCNT 这个数据库中新建查询。

记得写完的 Sql 语句，要涂黑之后再执行。



（5）执行如下的 SQL 语句，完成这一小题。题目设置得相当阴险。一般情况下，关系表是写在所有表的结尾，最后建立的。因为关系表中的所有数据都是来自于其它表，没有其它表的结构是建立不起来的。而改卷老师，故意将关系表摆在中间。

可以看到如下的 SQL 语句，必须将关系表的建立，摆在最后，这 3 张表才能顺利建立起来。

[sql] view plain copy

print?

```
1.  --在数据库中根据上述表的定义创建上述数据库，同时需建立相应的约束关系
2.  create table [EMPLOYEE](
```

```
    [EmpNo] varchar(8) not null primary key,
    [EmpName] varchar(50) not null,
    [EmpSex] varchar(2) check([EmpSex]='男' or [EmpSex]='女'),
    [EmpAge] int check([EmpAge]>0)
)
create table [COMPANY](
    [CmpNo] varchar(8) not null primary key,
    [CmpName] varchar(50) not null
)
create table [WORKS](
    [EmpNo] varchar(8) references [EMPLOYEE]([EmpNo]),
    [CmpNo] varchar(8) references [COMPANY]([CmpNo]),
    [Salary] int check([Salary]>0)
)
```

这里，所有表名、字段名补上[]，是为了避免，有某些表名、字段名触发系统的关键字。

同时注意，题目，需要同时建立约束关系。

因此，Sql 语句最后，该有的实体完整性、参照完整性、域完整性不能漏，没有就丢分。

2、将上面的数据输入到数据库中相应的表中

这里，推荐还是用最传统的 SQL 语句完成，也就是复制、粘贴改改数据的事情，比用图形界面操作要快。

[sql] view plain copy

print?

```
--将上面的数据输入到数据库中相应的表中
insert into [EMPLOYEE] values('E01','张三','女',32);
insert into [EMPLOYEE] values('E02','李四','男',28);
insert into [EMPLOYEE] values('E03','王五','女',42);
insert into [EMPLOYEE] values('E04','赵六','男',37);
insert into [EMPLOYEE] values('E05','陈七','男',51);

insert into [COMPANY] values('C01','阳光科技');
insert into [COMPANY] values('C02','晨光科技');
insert into [COMPANY] values('C03','未来科技');

insert into [WORKS] values('E01','C01',3000);
insert into [WORKS] values('E01','C02',4000);
insert into [WORKS] values('E02','C02',5000);
insert into [WORKS] values('E02','C03',2500);
insert into [WORKS] values('E03','C01',3500);
insert into [WORKS] values('E04','C02',3000);
insert into [WORKS] values('E05','C03',2000);
```

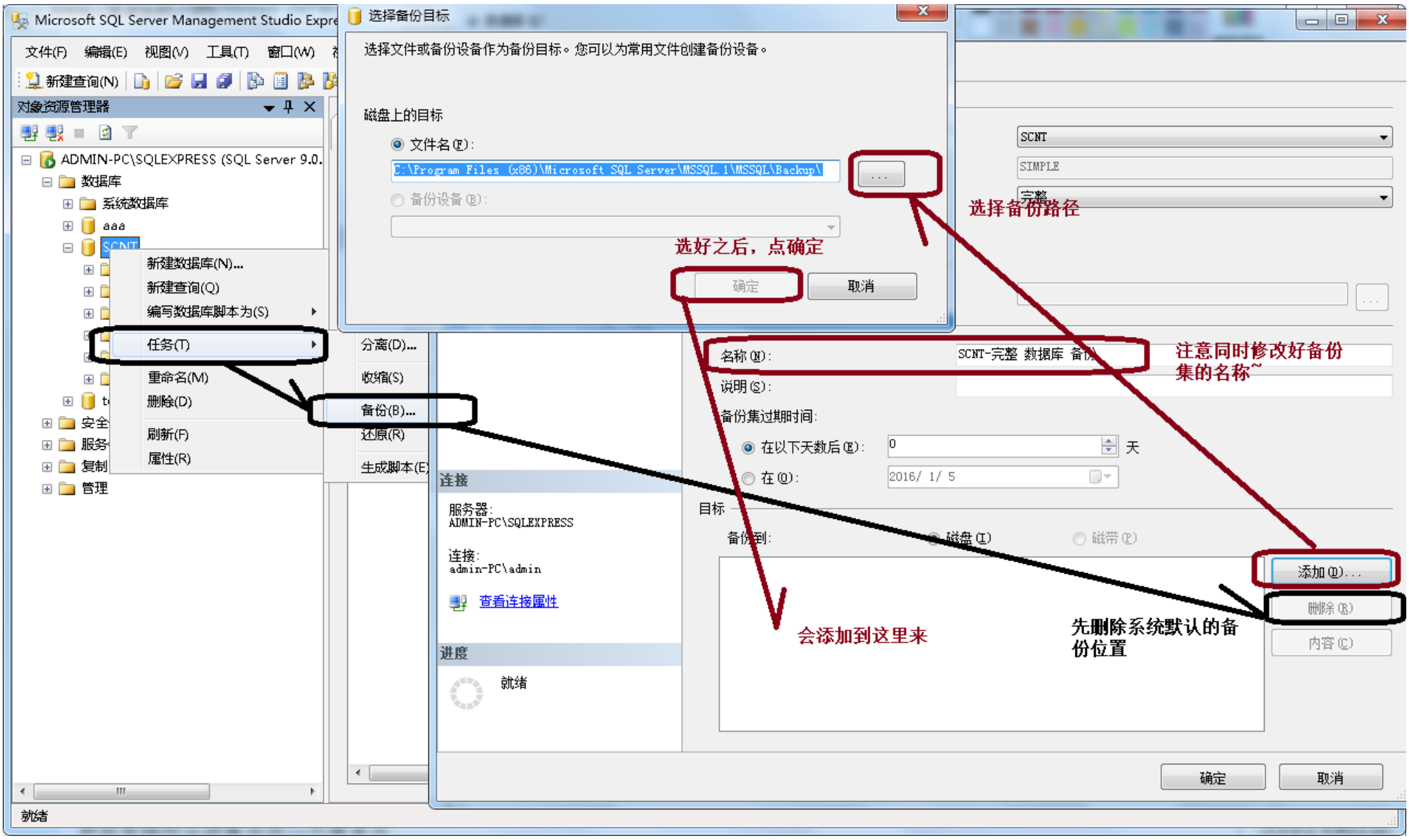
同样需要注意的是，关系表的数据最后才插入。

而且，清楚地了解该字段的数据类型，不是数字，自觉补上"，是数字，千万别有"

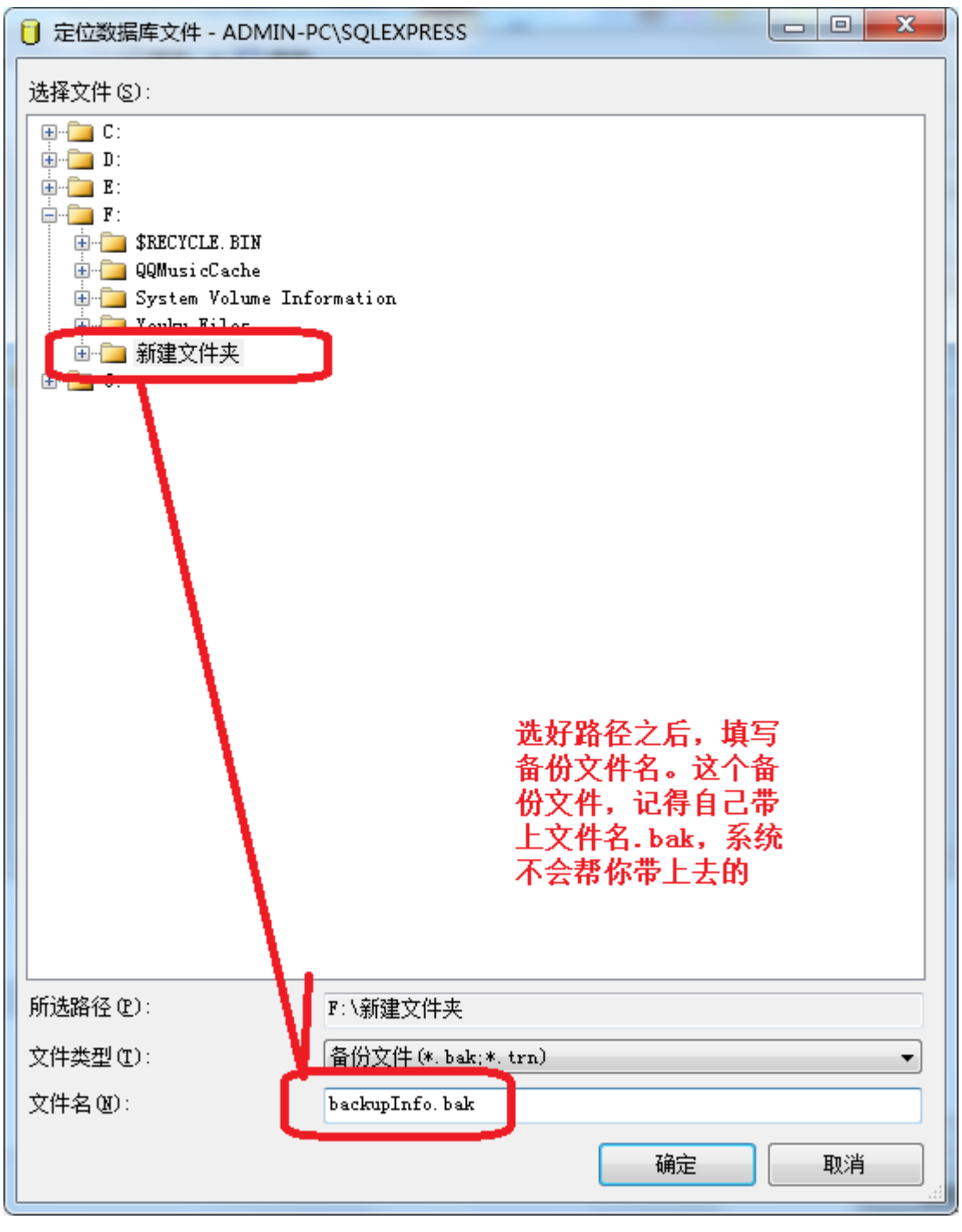
3、将数据库备份到你的文件夹下，命名为 backupInfo

备份，也就是几分钟的事，如下图，右击要备份的数据库，选择“任务”->"备份"

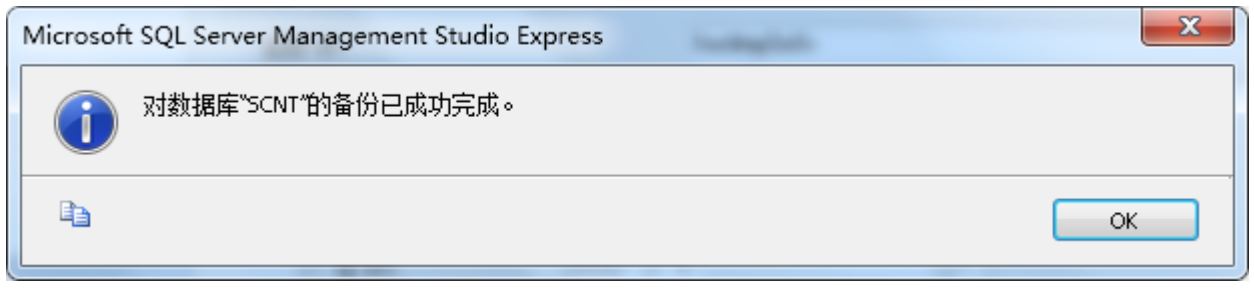
删除原有的默认备份路径之后，添加自定义的路径。



选好路径之后，自己写好备份文件名，注意带上后缀名，



弄好点确定，即可。



好，这个大题，至此做完，最好能够在 15 分钟之内做完。

下面进入到核心的 C#编程，建议这个 SQL Server Management Studio Express 别关。因为按照现在市场上软件编程 MVC 的思想，都是在数据库先组织好查询数据->推到相应的编程语言，这里是 C#窗体中->再组织好数据，打印出来，一会儿自然还要多次用到 SQL Server Management Studio Express。

三、基于上述数据库，请使用 sql server2005+vs2008 或 vs2010 完成员工信息管理系统，并生成相应的可运行文件（文件名为你的名字）。

具体要求如下：

1. 要求程序与数据库能进行有效连接，并具有完善的人机交互界面， 要求有参数输入界面和执行按钮，在界面上有结果输出展现区， 要求不要把所有操作全部集中在一个菜单内。

2.完成对员工关系表的添加，删除，修改和浏览四项功能。老师的性别要求用单选按钮实现。（15 分）

3.统计和查询：

（1）根据员工号或员工名查找员工所在的公司名和工资，员工号或员工名不能文本输入，要求使用下拉菜单实现，并与数据库中现有信息一致（10 分）

（2）统计年龄至少为 40 岁员工的总工资，工资按从大到小顺序排列；与数据库中现有信息一致（10 分）

（3）查询至少具有两份工作员工的姓名和其公司名。（10 分）

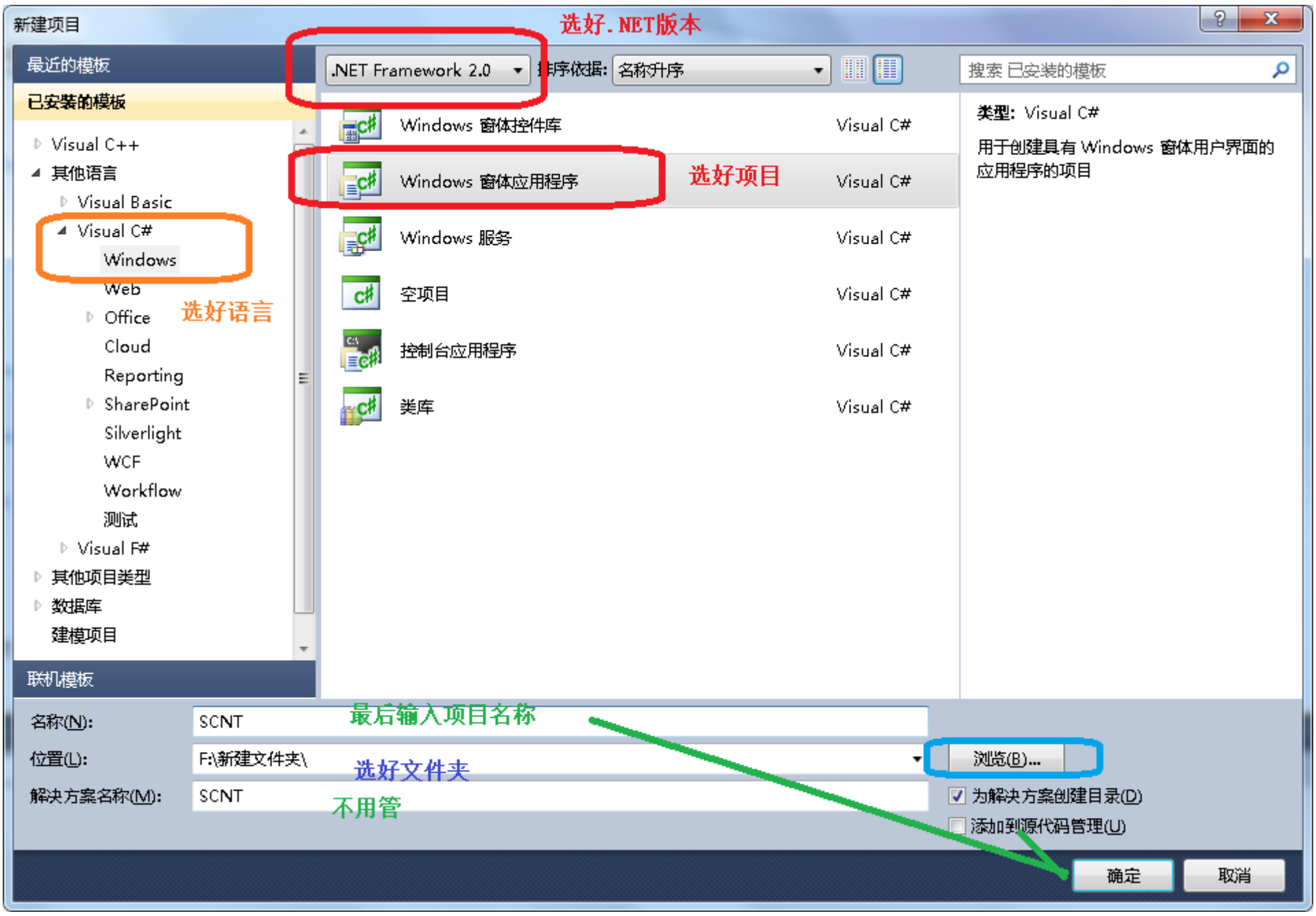
4. 具有数据完整性校验功能，当出现数据异常和操作异常时，程序应给出清楚完整的异常提示信息。（10 分）

答：

由于题目要求，上来新建 C#工程就尤为注意：

打开 VS2010 之后，点击文件->新建->项目，出现如下对话框，在左方的语言栏，请先找到 C#编程语言，有可能藏在“其它语言”中，之后选择 C#旗下的 Windows。

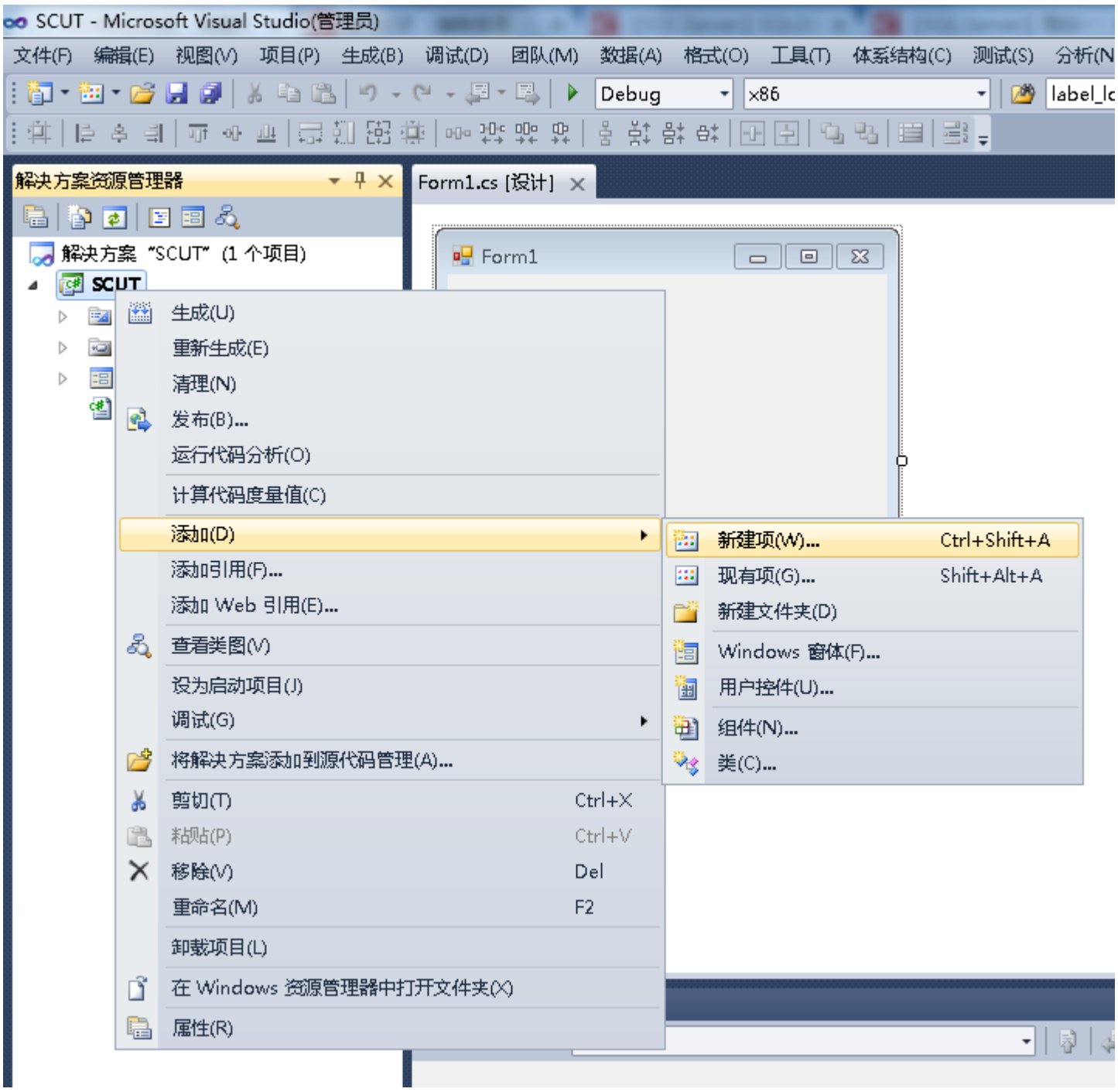
在选择“Windows 窗体应用程序”的时候，注意，是否有.NET 版本的选择，如果有，这个.NET 选择 2.0。不选，如果有.net4.x 也可以，编程写法也一样的，程序运行起来卡而已。



最后，最最最关键的是，就算你代码不会写，也要做好的是，选择好，工程的位置就是你第一题目建立的文件夹下的 Source，名称，按照题目要求，就是你的真实名称。

之后解决方案的名称，在你填完 名称 就自动完成填写，点击“确定”之后，就可以正式开始编程。

1、在正式的窗体开发之前，我们先如下图，在解决方案中，新建两个类.cs，磨刀不误砍柴工，



一个是用户数据库连接、查询的 DB.cs，这个类的具体作用在《【C#】利用 C#窗体与 SQL Server 的连接、Treeview 制作 SQL Server 数据库查看器》（点击打开链接）我已经写过了，这里不再复制、粘贴一次了，有兴趣可以去看看，具体代码如下：

[csharp] view plain copy

print?

```csharp
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;//DataTable 用到
using System.Data.SqlClient;//一系列的数据库操作类用到

namespace SCUT
{
    class DB : IDisposable
    {
        private SqlConnection sqlConnection;

        public DB()//私有无参构造函数
        {
            sqlConnection = new SqlConnection(@"server=.\SQLEXPRESS;database=SCNT;Trusted_Connection=SSPI;");
            sqlConnection.Open();
        }

        public DataTable getBySql(string sql)
        {//查询
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(new SqlCommand(sql, sqlConnection));
            DataTable dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);
            return dataTable;
        }

        public void setBySql(string sql)
        { //修改
            new SqlCommand(sql, sqlConnection).ExecuteNonQuery();
        }
```

```
31.
32.         public void Dispose()
33.         {//相当于析构函数
34.             sqlConnection.Close();
35.             //在 C#中关闭数据库连接不能在类的析构函数中关，否则会抛“内部 .Net Framework 数据提供程序错误 1”的异常
36.             //通过实现 C#中 IDisposable 接口中的 Dispose()方法主要用途是释放非托管资源。
37.         }
38.     }
39. }
```

这个类，直接使用《【C#】使用 Windows 身份验证连接 Sql Server，ListView 随窗体大小的变化而调节列宽》（点击打开链接）中介绍的不安全，但开发快速的 Windows 身份验证，去连接数据库，为了是不用像《【SQL Server】用户的设置与授权、sa 用户登录、查询一个数据库中有多少张表》（点击打开链接），再去 SQL Server 中开用户。

同时，去掉了这个类单例化的部分，单例主要作用是保证数据库连接只有一个，不会出现多次连接导致数据库的拥堵，这里题目没要求，不这样写，省些代码，感兴趣的，可以到《【php】利用单例模式设计数据库连接 Model 类》（点击打开链接）了解。

最后，还去掉 set、getBySql 中对于 Sql 注入过滤的部分，这些题目都没要求，不做，省时间，反正就算在实际开发怎么不及格，考试高分就行。

自己知道这是为了考试快速答题，但不符合实际开发就可以了。没必要纠结。

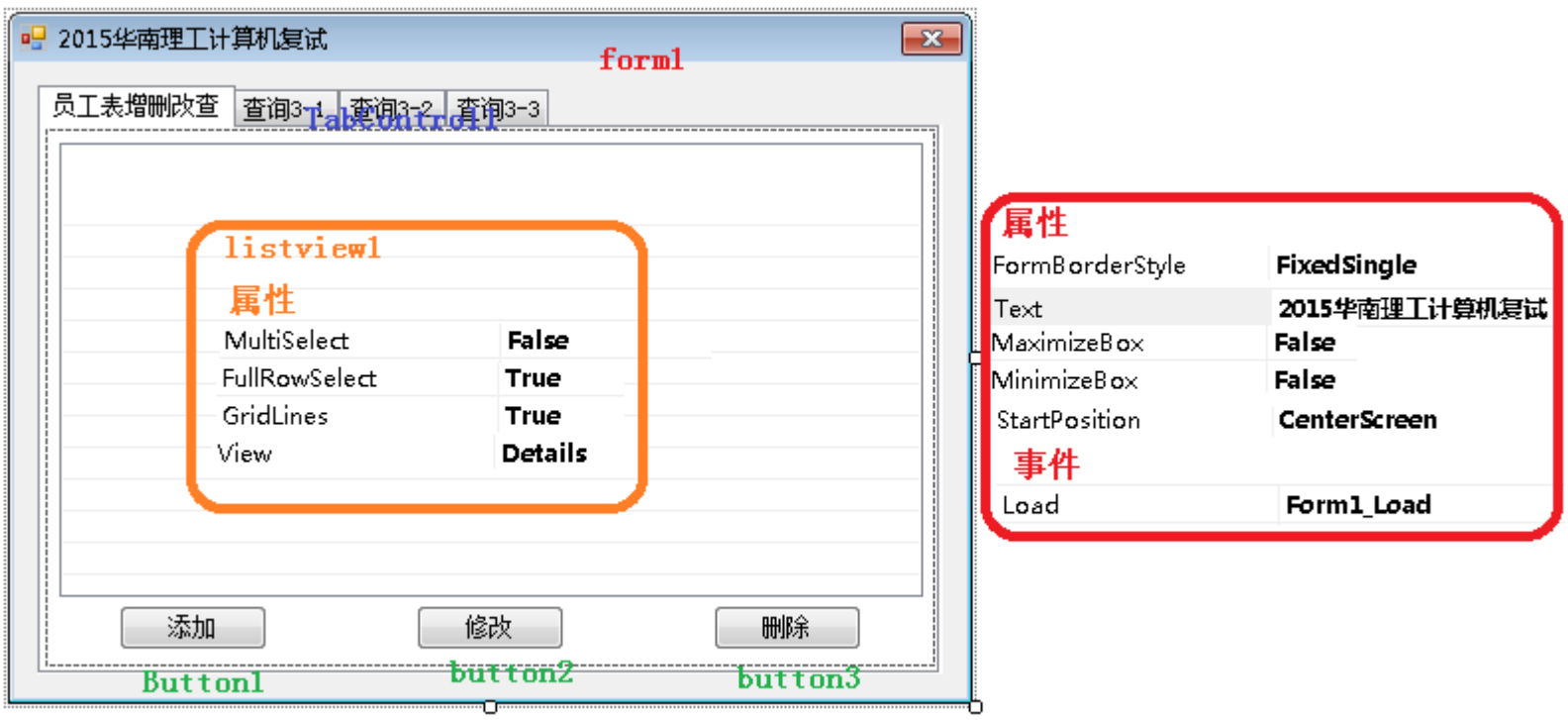
之后，我们再搞一个便于窗体之间与窗体之间传递数据的 Intent.cs，具体代码如下，这在《【C#】窗体间互相传值》（点击打开链接）说过了，这里不再赘述。就一个存数据的字典容器。

[csharp] view plain copy

print?

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Text;
4.
5.  namespace SCUT
6.  {
7.      class Intent
8.      {
9.          public static Dictionary<string, Object> dict = new Dictionary<string, Object>();
10.     }
11. }
```

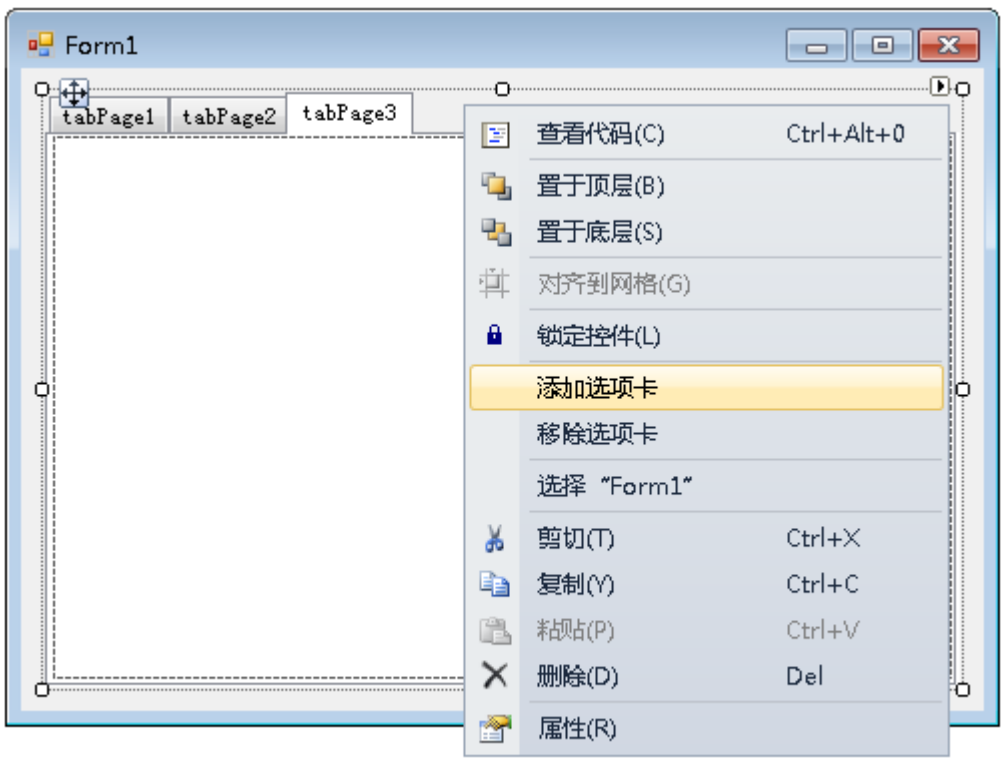
上述两个工具性的 cs 类做完之后，可以正式开始窗体的编写。Form1.cs 的布局、要修改的属性、添加的事件如下：



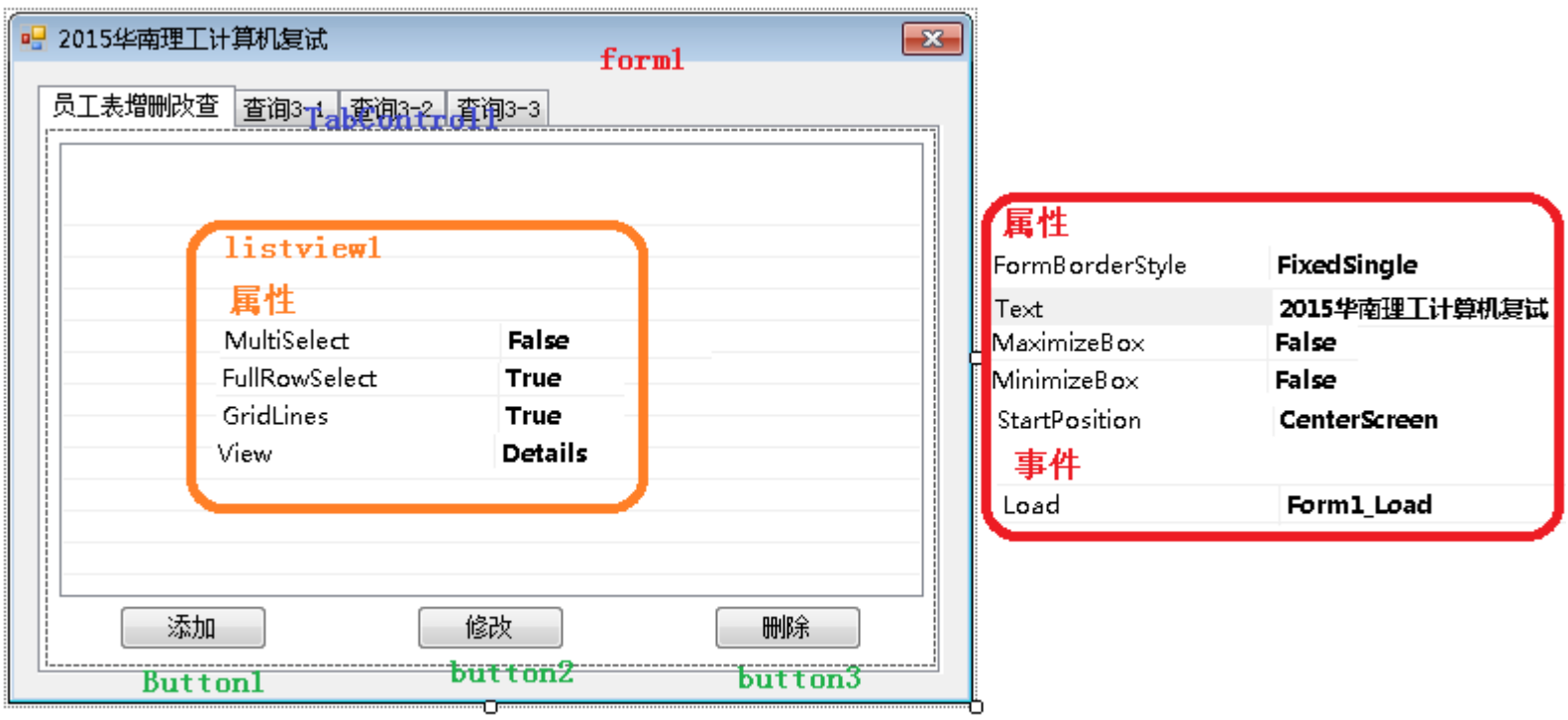
对于 C#控件毫无概念的同学，可以参考《【C#】简单窗体程序，判断是否闰年，禁止窗体调整大小，关闭窗体前的判断 》（点击打开链接）

这里摆了一个 TabControl，具体可以看《【C#】标签页》（点击打开链接），完成题目 1，不要将所有功能摆在一个窗体的要求。

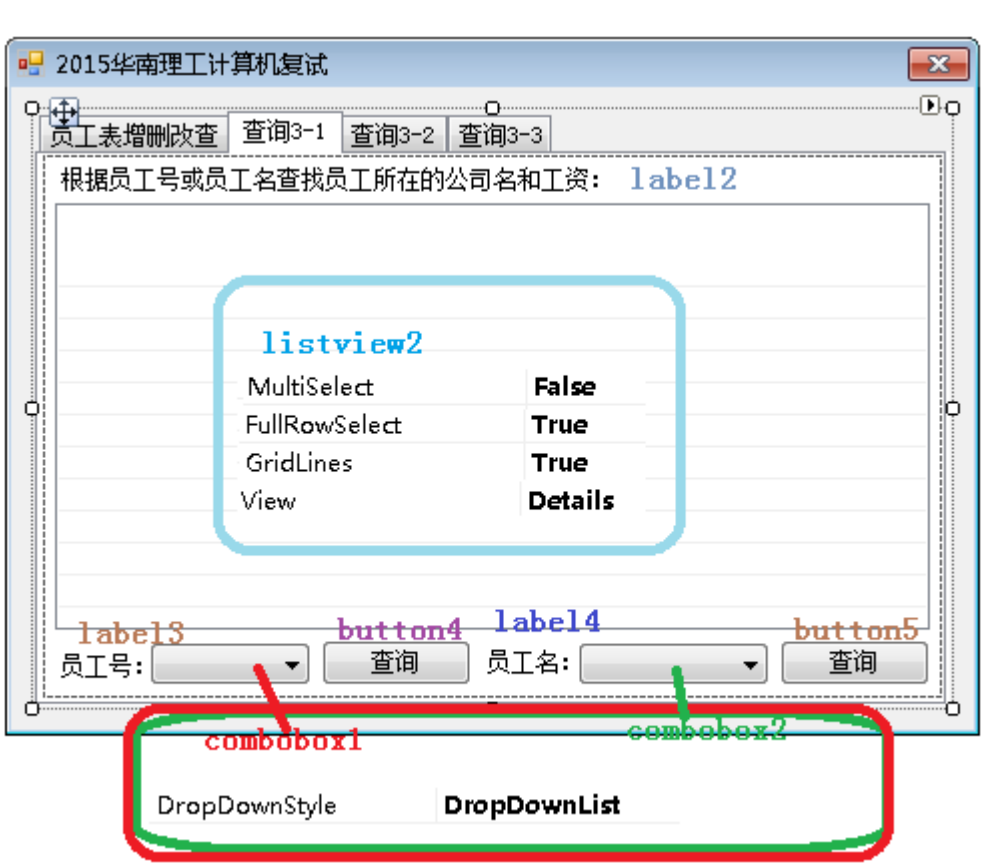
因为接下来要完成 4 个功能，如下图，右击这个 TabControl，为其添加到 4 个选项卡



在 TabControl 中的第一个选项卡摆了一个 listview，修改的属性如下图所示，listview 的具体使用，请看《【C#】ListView 的使用，对 Access 数据库的增删改查》（点击打开链接）



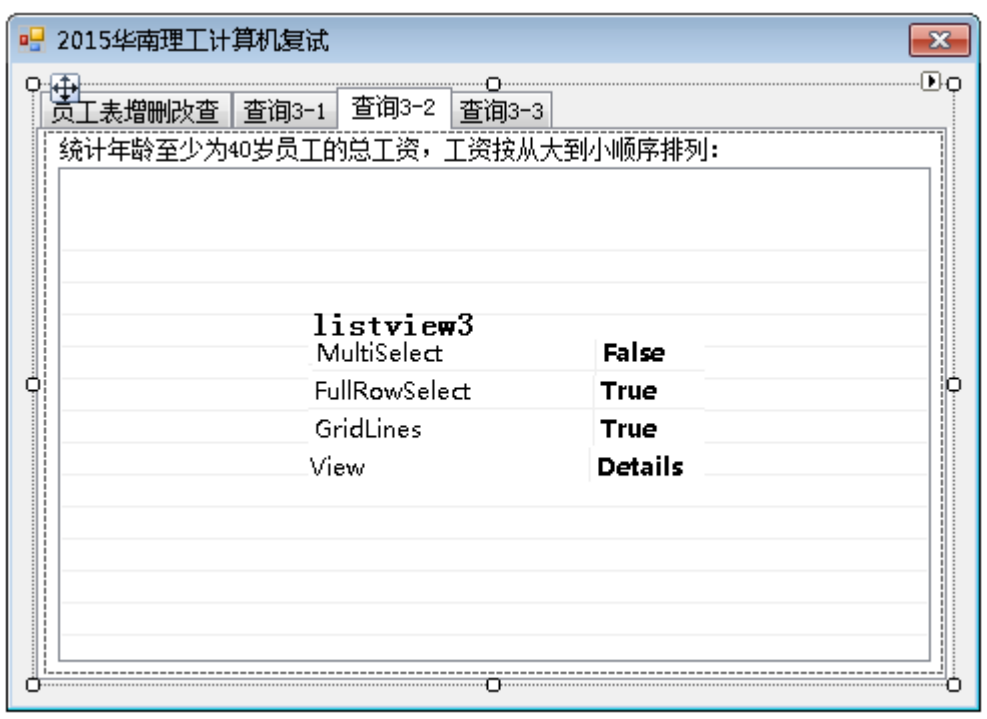
之后，第 2 个标签页的布局如下：

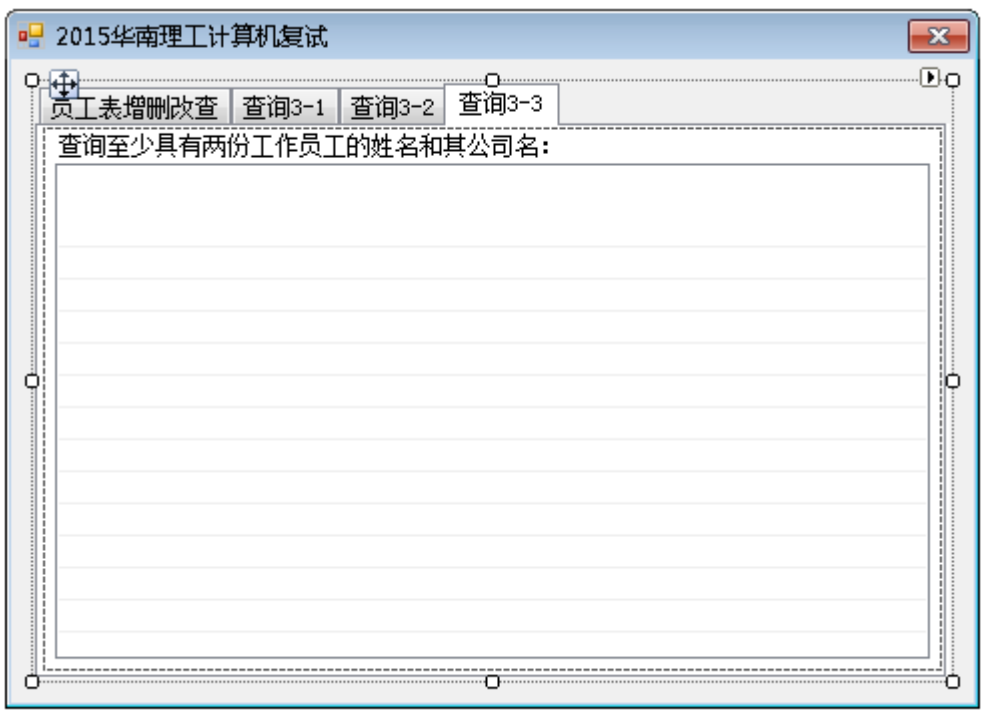


这里应题目的要求，要使用下拉菜单 combobox，这个组件在《【C#】文件选择对话框 OpenFileDialog 与下列列表 ComboBox》（点击打开链接）说过了，这里不再赘述，注意，将其的 DropDownStyle，改成不能编辑的 DropDownList。

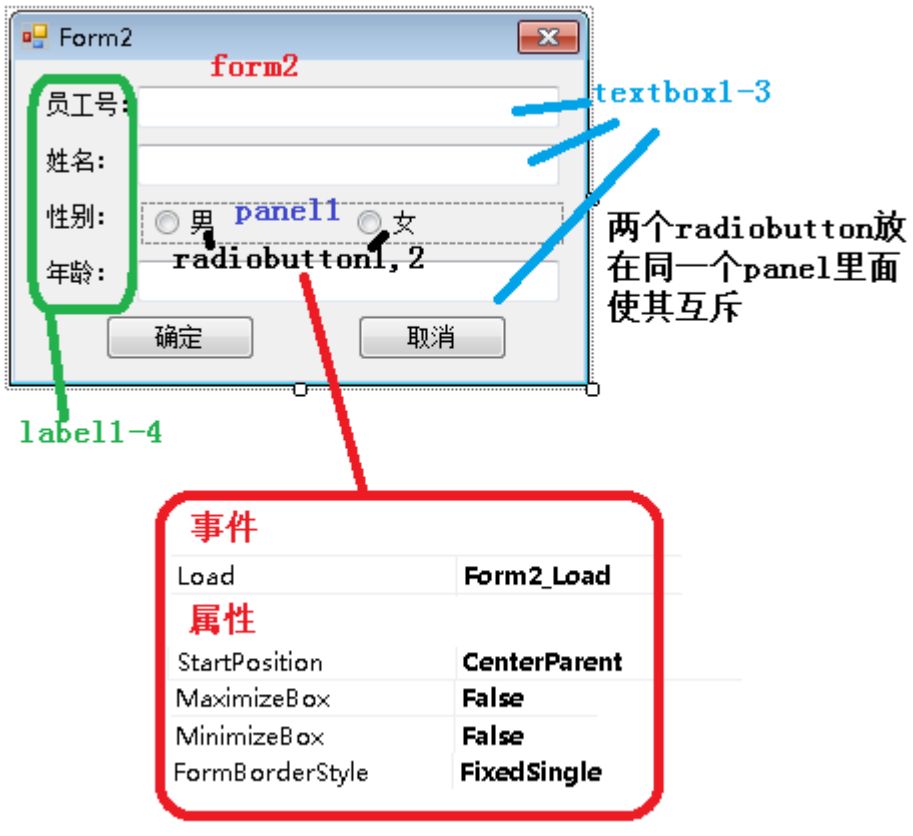
之后，第三个、第四个标签页的布局内容如下，这两页的布局、所修改的属性内容基本相同，都是一个 label 加 listview，因为，题目只是要求将查询结果打印到窗体上来。

不过，精华在最后的 SQL 查询代码~





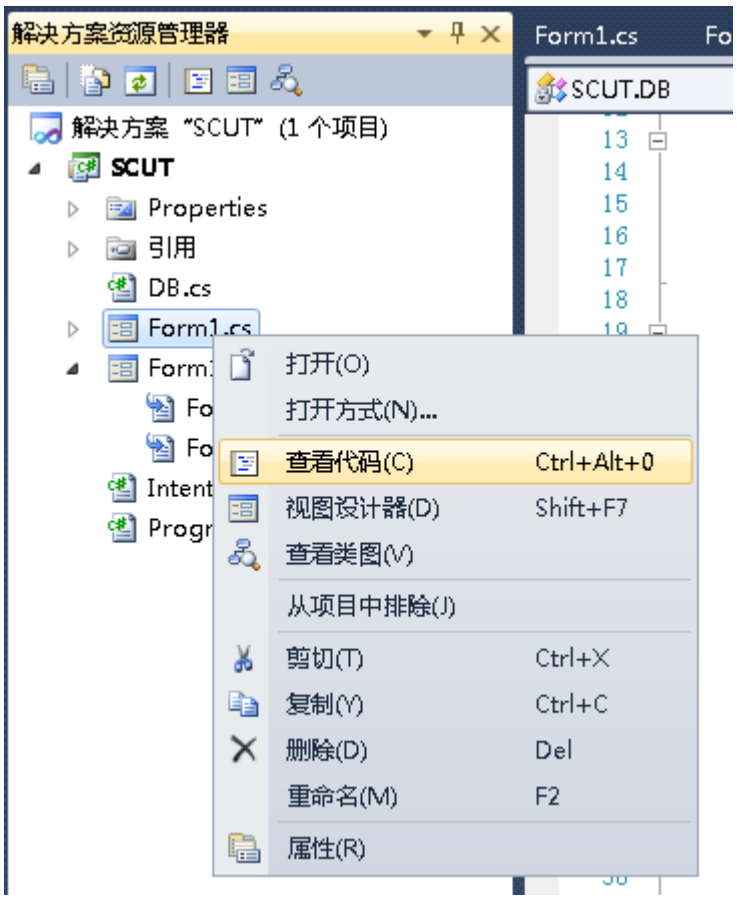
接下来，为了完成第 2 题对员工表的增删改查。我们要新建一个窗体 Form2，基本的流程同《【C#】窗体间互相传值》（点击打开链接），其布局、要修改的属性、添加的事件如下图所示：



这里唯一注意的是，由于题目要求老师的性别要求用单选按钮实现。那么你只好在姓名那里，先拖一个 panel1，再放上两个 radiobutton，这样，两个 radiobutton 就自动互斥了。

接下来，双击 form1.cs、form2.cs 各个标签页的各个 button，为各个按钮添加点击事件。

如果你的 vs 没有自动跳转，请按如下图的方式，进入代码编写界面。



Form1.cs 的代码如下：

[csharp] view plain copy

print?

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;//用到了正则表达式

namespace SCUT
{
    public partial class Form1 : Form
    {
        DB db;
        public Form1()
        {
            InitializeComponent();
            db = new DB();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //员工表 增删改查 中“查部分”
            //生成表头
            listView1.Columns.Add("员工号", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView1.Columns.Add("员工姓名", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView1.Columns.Add("性别", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView1.Columns.Add("年龄", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            //表的内容
            DataTable table = db.getBySql(@"select * from [EMPLOYEE]");
            listView1.BeginUpdate();//数据更新，UI 暂时挂起，直到 EndUpdate 绘制控件，可以有效避免闪烁并大大提高加载速度
            for (int i = 0; i < table.Rows.Count; i++)
            {
                ListViewItem listViewItem = new ListViewItem();//生成每一列
                for (int j = 0; j < table.Columns.Count; j++)
                {
```

```
                if (j <= 0)
                {
                    listViewItem.Text = table.Rows[i][j] + "";
                }
                else
                {
                    listViewItem.SubItems.Add(table.Rows[i][j] + "");
                }
            }
            listView1.Items.Add(listViewItem);
        }
        listView1.EndUpdate();//结束数据处理，UI 界面一次性绘制

        //查询 3-1
        //加载现有的员工号
        table = db.getBySql(@"select [EmpNo] from [EMPLOYEE]");
        for (int i = 0; i < table.Rows.Count; i++)
        {
            for (int j = 0; j < table.Columns.Count; j++)
            {
                comboBox1.Items.Add(table.Rows[i][j] + "");
            }
        }
        comboBox1.SelectedIndex = 0;
        //加载现有的员工名
        table = db.getBySql(@"select [EmpName] from [EMPLOYEE]");
        for (int i = 0; i < table.Rows.Count; i++)
        {
            for (int j = 0; j < table.Columns.Count; j++)
            {
                comboBox2.Items.Add(table.Rows[i][j] + "");
            }
        }
        comboBox2.SelectedIndex = 0;

        //表 3-2
        //统计年龄至少为 40 岁员工的总工资，工资按从大到小顺序排列
        //生成表头
        listView3.Columns.Add("员工号", listView1.Width / 5 - 1, HorizontalAlignment.Left);
        listView3.Columns.Add("员工姓名", listView1.Width / 5 - 1, HorizontalAlignment.Left);
        listView3.Columns.Add("性别", listView1.Width / 5 - 1, HorizontalAlignment.Left);
        listView3.Columns.Add("年龄", listView1.Width / 5 - 1, HorizontalAlignment.Left);
        listView3.Columns.Add("总工资", listView1.Width / 5 - 1, HorizontalAlignment.Left);
        //表的内容
        table = db.getBySql(@"select [EMPLOYEE].[EmpNo],[EMPLOYEE].[EmpName],[EMPLOYEE].[EmpSex],[EMPLOYEE].[EmpAge],sum([WORKS].[Salary]) as '
    总工资' " +
            " from [EMPLOYEE],[WORKS]" +
            " where [EMPLOYEE].[EmpAge]>=40" +
            " and [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
            " group by [EMPLOYEE].[EmpNo],[EMPLOYEE].[EmpName],[EMPLOYEE].[EmpSex],[EMPLOYEE].[EmpAge]" +
            " order by '总工资' desc");
        listView3.BeginUpdate();//数据更新，UI 暂时挂起，直到 EndUpdate 绘制控件，可以有效避免闪烁并大大提高加载速度
        for (int i = 0; i < table.Rows.Count; i++)
        {
            ListViewItem listViewItem = new ListViewItem();//生成每一列
            for (int j = 0; j < table.Columns.Count; j++)
            {
                if (j <= 0)
                {
                    listViewItem.Text = table.Rows[i][j] + "";
                }
                else
                {
                    listViewItem.SubItems.Add(table.Rows[i][j] + "");
                 }
             }
             listView3.Items.Add(listViewItem);
         }
         listView3.EndUpdate();//结束数据处理，UI 界面一次性绘制。

         //表 3-3
```

```
            //查询至少具有两份工作员工的姓名和其公司名
            //生成表头
            listView4.Columns.Add("员工姓名", listView1.Width / 2 - 2, HorizontalAlignment.Left);
            listView4.Columns.Add("公司名", listView1.Width / 2 - 2, HorizontalAlignment.Left);
            //表的内容
            table = db.getBySql(@"select [EMPLOYEE].[EmpName],[COMPANY].[CmpName] from [EMPLOYEE],[COMPANY],[WORKS],(" +
            " select [EmpName],count([CmpName]) as 'CmpNum'" +
            " from [EMPLOYEE],[WORKS],[COMPANY]" +
            " where [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
            " and [COMPANY].[CmpNo]=[WORKS].[CmpNo]" +
            " group by [EmpName]" +
            " having count([CmpName])>1" +
            " ) as t1" +
            " where [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
            " and [COMPANY].[CmpNo]=[WORKS].[CmpNo]" +
            " and [EMPLOYEE].[EmpName]=t1.[EmpName]");
            listView4.BeginUpdate();//数据更新，UI 暂时挂起，直到 EndUpdate 绘制控件，可以有效避免闪烁并大大提高加载速度
            for (int i = 0; i < table.Rows.Count; i++)
            {
                ListViewItem listViewItem = new ListViewItem();//生成每一列
                for (int j = 0; j < table.Columns.Count; j++)
                {
                    if (j <= 0)
                    {
                        listViewItem.Text = table.Rows[i][j] + "";
                    }
                    else
                    {
                        listViewItem.SubItems.Add(table.Rows[i][j] + "");
                    }
                }
                listView4.Items.Add(listViewItem);
            }
            listView4.EndUpdate();//结束数据处理，UI 界面一次性绘制。
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();//声明要使用 form2 窗体
            //将 form1 当前的位置压入 Intent 中的 dict
            Intent.dict["form1_text"] = this.Text;
            Intent.dict["form1_flag"] = 0;//传个 flag 进去代表这是“添加”
            if (form2.ShowDialog() == DialogResult.OK)
            {//这个判断，将会等到 form2 被关闭之后才执行，如果 form2 返回一个 OK 值
                bool canAdd = true;
                foreach (ListViewItem item in this.listView1.Items)
                {
                    if (Intent.dict["form2_textbox1_text"] + "" == item.SubItems[0].Text)
                    {
                        canAdd = false;
                        MessageBox.Show("已存在该员工号！", this.Text);
                        break;
                    }
                }
                Regex regex = new Regex("^[0-9]*$");
                if (!regex.IsMatch(Intent.dict["form2_textbox3_text"] + ""))//利用正则表达式判断是否输入的是数字
                {
                    canAdd = false;
                    MessageBox.Show("年龄不为正数！", this.Text);
                }
                if (canAdd)
                {
                    ListViewItem listViewItem = new ListViewItem();//在 listview 中添加一项
                    listViewItem.Text = Intent.dict["form2_textbox1_text"] + "";
                    listViewItem.SubItems.Add(Intent.dict["form2_textbox2_text"] + "");
                    listViewItem.SubItems.Add(Intent.dict["form2_radioButton"] + "");
                    listViewItem.SubItems.Add(Intent.dict["form2_textbox3_text"] + "");
                    listView1.Items.Add(listViewItem);
                    db.setBySql("insert into [EMPLOYEE] values('" + Intent.dict["form2_textbox1_text"] + "','" + Intent.dict["form2_textbox2_text"]
        + "','" + Intent.dict["form2_radioButton"] + "'," + Intent.dict["form2_textbox3_text"] + ")");
                }
```

```
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();//声明要使用 form2 窗体
            //将 form1 当前的位置压入 Intent 中的 dict
            Intent.dict["form1_text"] = this.Text;
            Intent.dict["form1_flag"] = 1;//传个 flag 进去代表这是“修改”
            Intent.dict["form1_selectedItems0"] = listView1.SelectedItems[0].SubItems[0].Text;
            Intent.dict["form1_selectedItems1"] = listView1.SelectedItems[0].SubItems[1].Text;
            Intent.dict["form1_selectedItems2"] = listView1.SelectedItems[0].SubItems[2].Text;
            Intent.dict["form1_selectedItems3"] = listView1.SelectedItems[0].SubItems[3].Text;
            Intent.dict["form1_flag"] = 1;//传个 flag 进去代表这是“修改”
            if (form2.ShowDialog() == DialogResult.OK)
            {//这个判断，将会等到 form2 被关闭之后才执行，如果 form2 返回一个 OK 值
                bool canUpdate = true;
                if (!((Intent.dict["form1_selectedItems0"] + "") == (Intent.dict["form2_textbox1_text"] + "")))//仅当用户修改过员工号才进行遍历
                {
                    foreach (ListViewItem item in this.listView1.Items)
                    {
                        if (Intent.dict["form2_textbox1_text"] + "" == item.SubItems[0].Text)
                        {
                            canUpdate = false;
                            MessageBox.Show("已存在该员工号！", this.Text);
                            break;
                        }
                    }
                }
                Regex regex = new Regex("^[0-9]*$");
                if (!regex.IsMatch(Intent.dict["form2_textbox3_text"] + ""))//利用正则表达式判断是否输入的是数字
                {
                    canUpdate = false;
                    MessageBox.Show("年龄不为正数！", this.Text);
                }
                if (canUpdate)
                {
                    ListViewItem listViewItem = new ListViewItem();//在 listview 中添加一项
                    listView1.SelectedItems[0].SubItems[0].Text = Intent.dict["form2_textbox1_text"] + "";
                    listView1.SelectedItems[0].SubItems[1].Text = Intent.dict["form2_textbox2_text"] + "";
                    listView1.SelectedItems[0].SubItems[2].Text = Intent.dict["form2_radioButton"] + "";
                    listView1.SelectedItems[0].SubItems[3].Text = Intent.dict["form2_textbox3_text"] + "";
                    db.setBySql("update [EMPLOYEE] set [EmpNo]='" + Intent.dict["form2_textbox1_text"] + "' where [EmpNo]='" + Intent.dict["form1_selectedItems0"] + "';");
                    db.setBySql("update [EMPLOYEE] set [EmpName]='" + Intent.dict["form2_textbox2_text"] + "' where [EmpName]='" + Intent.dict["form1_selectedItems1"] + "';");
                    db.setBySql("update [EMPLOYEE] set [EmpSex]='" + Intent.dict["form2_radioButton"] + "' where [EmpSex]='" + Intent.dict["form1_selectedItems2"] + "';");
                    db.setBySql("update [EMPLOYEE] set [EmpAge]='" + Intent.dict["form2_textbox3_text"] + "' where [EmpAge]='" + Intent.dict["form1_selectedItems3"] + "';");
                }
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            db.setBySql("delete from [EMPLOYEE] where [EmpNo]='" + listView1.SelectedItems[0].SubItems[0].Text + "';");
            listView1.SelectedItems[0].Remove();//删除一定要放在数据库操作之后，不然选中项再也取不到了
        }

        private void button4_Click(object sender, EventArgs e)
        {
            listView2.Clear();
            //生成表头
            listView2.Columns.Add("员工号", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("员工名", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("公司名", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("薪水", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            //表的内容
            DataTable table = db.getBySql(@"select [EMPLOYEE].[EmpNo],[EMPLOYEE].[EmpName],[COMPANY].[CmpName],[WORKS].[Salary] from [EMPLOYEE],[COMPANY],[WORKS]" +
```

```
            " where [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
            " and [COMPANY].[CmpNo]=[WORKS].[CmpNo]" +
            " and [EMPLOYEE].[EmpNo]='" + comboBox1.Text + "'");
            listView2.BeginUpdate();//数据更新，UI 暂时挂起，直到 EndUpdate 绘制控件，可以有效避免闪烁并大大提高加载速度
            for (int i = 0; i < table.Rows.Count; i++)
            {
                ListViewItem listViewItem = new ListViewItem();//生成每一列
                for (int j = 0; j < table.Columns.Count; j++)
                {
                    if (j <= 0)
                    {
                        listViewItem.Text = table.Rows[i][j] + "";
                    }
                    else
                    {
                        listViewItem.SubItems.Add(table.Rows[i][j] + "");
                    }
                }
                listView2.Items.Add(listViewItem);
            }
            listView2.EndUpdate();//结束数据处理，UI 界面一次性绘制
        }

        private void button5_Click(object sender, EventArgs e)
        {
            listView2.Clear();
            //生成表头
            listView2.Columns.Add("员工号", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("员工名", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("公司名", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            listView2.Columns.Add("薪水", listView1.Width / 4 - 1, HorizontalAlignment.Left);
            //表的内容
            DataTable table = db.getBySql(@"select [EMPLOYEE].[EmpNo],[EMPLOYEE].[EmpName],[COMPANY].[CmpName],[WORKS].[Salary] from [EMPLOYEE],[CO
MPANY],[WORKS]" +
            " where [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
            " and [COMPANY].[CmpNo]=[WORKS].[CmpNo]" +
            " and [EMPLOYEE].[EmpName]='" + comboBox2.Text + "'");
            listView2.BeginUpdate();//数据更新，UI 暂时挂起，直到 EndUpdate 绘制控件，可以有效避免闪烁并大大提高加载速度
            for (int i = 0; i < table.Rows.Count; i++)
            {
                ListViewItem listViewItem = new ListViewItem();//生成每一列
                for (int j = 0; j < table.Columns.Count; j++)
                {
                    if (j <= 0)
                    {
                        listViewItem.Text = table.Rows[i][j] + "";
                    }
                    else
                    {
                        listViewItem.SubItems.Add(table.Rows[i][j] + "");
                    }
                }
                listView2.Items.Add(listViewItem);
            }
            listView2.EndUpdate();//结束数据处理，UI 界面一次性绘制
        }
    }
}
```

Form2.cs 的代码如下：

[csharp] view plain copy

print?

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

```

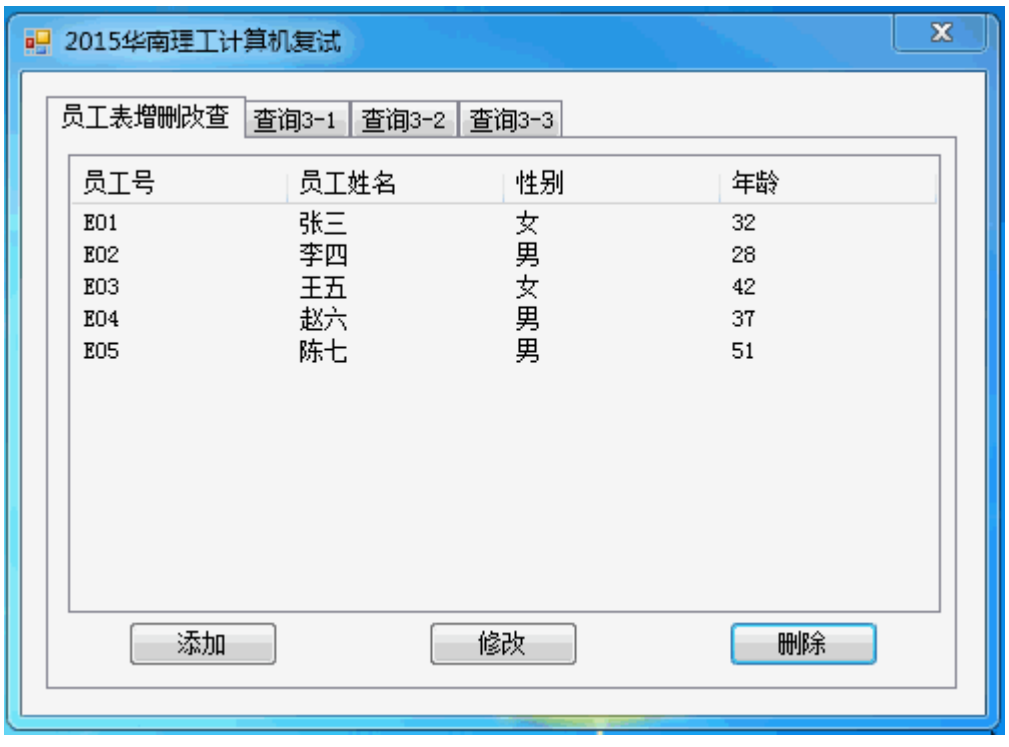
```
namespace SCUT
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            if ((int)Intent.dict["form1_flag"] == 0)
            {
                this.Text = Intent.dict["form1_text"] + "";
                textBox1.Focus();//设置焦点停留在 textBox1 中
            }
            else
            {
                this.Text = Intent.dict["form1_text"] + "";
                textBox1.Text = Intent.dict["form1_selectedItems0"] + "";
                textBox2.Text = Intent.dict["form1_selectedItems1"] + "";
                if (Intent.dict["form1_selectedItems2"] + "" == "男")
                {
                    radioButton1.Checked = true;
                }
                else {
                    radioButton2.Checked = true;
                }
                textBox3.Text = Intent.dict["form1_selectedItems3"] + "";
                textBox1.Focus();//设置焦点停留在 textBox1 中
                textBox1.SelectAll();//要先有焦点才能全选
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || (!radioButton1.Checked && !radioButton2.Checked))
            {
                MessageBox.Show("任意一项没有完成填写！", this.Text);
            }
            else
            {
                //关闭 form2 之间，将要传给 form1 的值压入 Intent 中的 dict
                Intent.dict["form2_textbox1_text"] = textBox1.Text;
                Intent.dict["form2_textbox2_text"] = textBox2.Text;
                if (radioButton1.Checked)
                {
                    Intent.dict["form2_radioButton"] = "男";
                }
                else
                {
                    Intent.dict["form2_radioButton"] = "女";
                }
                Intent.dict["form2_textbox3_text"] = textBox3.Text;
                this.DialogResult = DialogResult.OK;//同时设置返回值为 OK，不设置的话，默认返回 Cancel
                this.Close();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

在上述的代码中：

（1）对于第 2 题的增删改查，就是《【C#】ListView 的使用，对 Access 数据库的增删改查》（点击打开链接）中的内容，第 4 大题，要保证输入的完整性校验，使用的 C#的正则表达式，具体请看《【C#】利用正则表达式判断输入是否为纯数字、容器类》（点击打开链接）。

这属于 C#对 Sql Server 操作的基础，是每一个学习 C#窗体，投身 C#开发程序猿的必修课。

做好之后，程序运行结果如下：



（2）对于题目第 3-1 中，在程序一开始的 Load 事件，先直接将数据库员工表 EMPLOYEE 表中的员工号 EmpNo，员工名 EmpName 查询出来，加载到两个 Combobox 中，用户点击 button4 或者 button5，利用 Combobox 的 Text 属性获得当前 Combobox 选定的值。

此时，问题演变成，已知员工号、员工名查找员工所在的公司名和工资。

利用多表连接查询，其实说白了，因为属性分布于 EMPLOYEE、WORKS、COMPANY 三张表，就是一次性，查询 EMPLOYEE、WORKS、COMPANY 三张表，根据建表时的参照完整性，也就是外键，添加一个连接条件。

以下是语句：

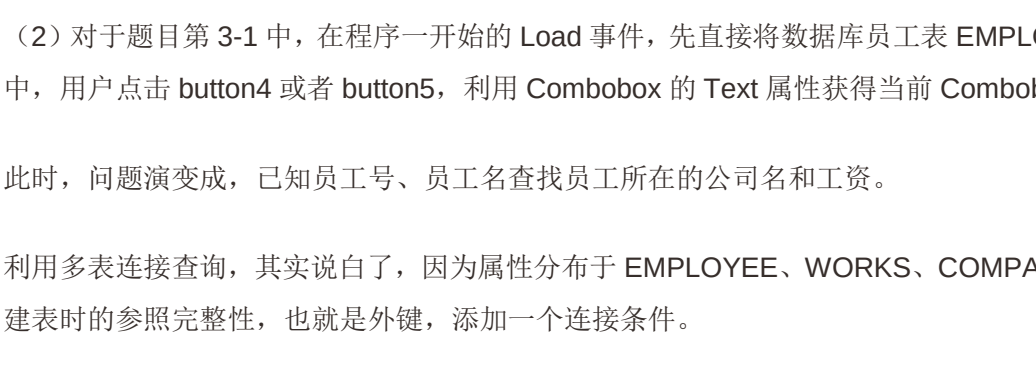
[sql] view plain copy

print?

```sql
select * from EMPLOYEE,COMPANY,WORKS
where EMPLOYEE.EmpNo=WORKS.EmpNo
and COMPANY.CmpNo=WORKS.CmpNo
```

的查询结果，注意上述语句，写在 C#代码中，涉及换行，每行的第一个字符，一定是空格，要么每行的最后一个字符是空格。

同时，表名、字段名加上[]，以免触发系统关键字产生歧义。在字符串上补上@，让此字符串所有要涉及转义的字符，自动转义。



在上述查询结果中，只要取出相应的列，再补上已知的员工号或员工名，构造到 C#中就好，将上述的 sql 语句中的*换成 某某表.某某字段，某某表.某某字段…… 的形式就好，

因此 C#中的语句也就演变成：

[csharp] view plain copy

print?

```csharp
DataTable table = db.getBySql(@"select [EMPLOYEE].[EmpNo],[EMPLOYEE].[EmpName],[COMPANY].[CmpName],[WORKS].[Salary] from [EMPLOYEE],[COMPANY],[WORKS]" +
" where [EMPLOYEE].[EmpNo]=[WORKS].[EmpNo]" +
" and [COMPANY].[CmpNo]=[WORKS].[CmpNo]" +
" and [EMPLOYEE].[EmpName]='" + comboBox2.Text + "'");
```

以上是根据员工名查询的，根据员工号查询的同理。

完成之后的截图如下：

（3）题目 3-2 中的查询，没有任何关于 C#的事，核心在于 SQL 语句的构造，这里涉及到《【MySQL】利用 group by 附带 having 进行聚类查询》（点击打开链接）中提到过的聚类查询。

思考过程如下，同样还是从：

[sql] view plain copy

print?

```sql
select * from EMPLOYEE,COMPANY,WORKS
where EMPLOYEE.EmpNo=WORKS.EmpNo
and COMPANY.CmpNo=WORKS.CmpNo
```

三张表联立起来的查询结果入手，

接下来，我们要将查询结果中，出现 相同的 员工号、员工姓名、性别、年龄 的行，合起来，

	EmpNo	EmpName	EmpSex	EmpAge	CmpNo	CmpName	EmpNo	CmpNo	Salary
1	E01	张三	女	32	C01	阳光科技	E01	C01	3000
2	E01	张三	女	32	C02	晨光科技	E01	C02	4000
3	E02	李四	男	28	C02	晨光科技	E02	C02	5000
4	E02	李四	男	28	C03	未来科技	E02	C03	2500
5	E03	王五	女	42	C01	阳光科技	E03	C01	3500
6	E04	赵六	男	37	C02	晨光科技	E04	C02	3000
7	E05	陈七	男	51	C03	未来科技	E05	C03	2000

整合过程中，对于不相同的公司号、公司名，我们舍弃，而工资，我们采取相加的形式合起来，因此，也就得到如下的查询语句：

[sql] view plain copy

print?

```sql
select EMPLOYEE.EmpNo as '员工号',EMPLOYEE.EmpName as '员工姓名',EMPLOYEE.EmpSex as '性别',EMPLOYEE.EmpAge as '年龄',sum(WORKS.Salary) as '总工资'
from EMPLOYEE,WORKS
where EMPLOYEE.EmpAge>=40
and EMPLOYEE.EmpNo=WORKS.EmpNo
group by EMPLOYEE.EmpNo,EMPLOYEE.EmpName,EMPLOYEE.EmpSex,EMPLOYEE.EmpAge
```

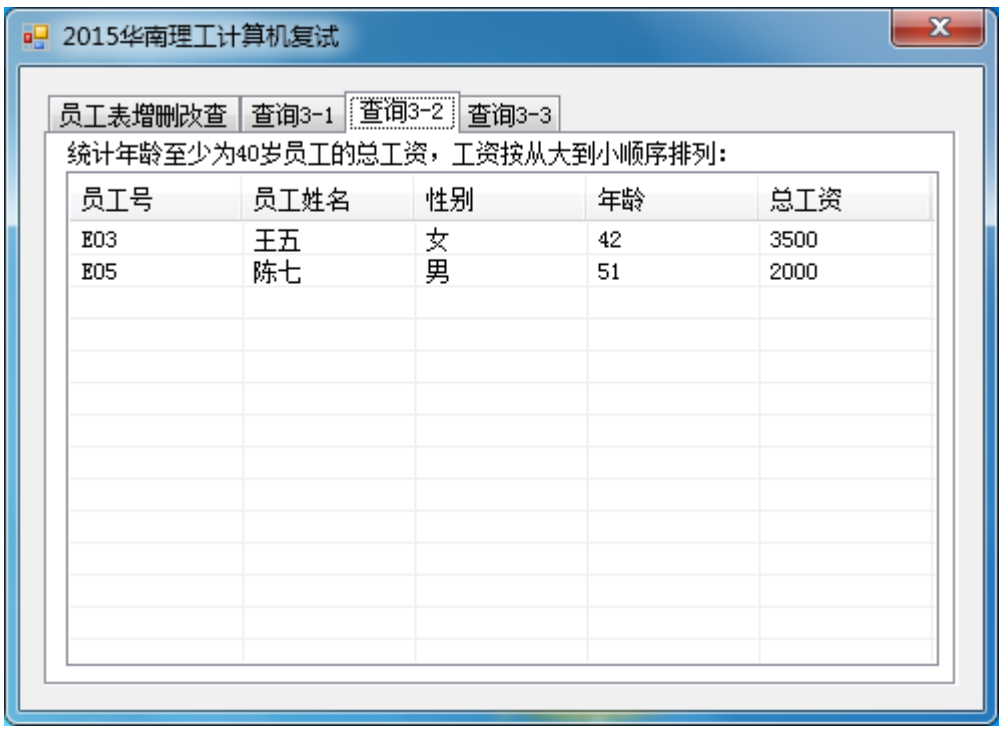
查询结果如下：

```
select EMPLOYEE.EmpNo as '员工号',EMPLOYEE.EmpName as '员工姓名',EMPLOYEE.EmpSex as '性别',EMPLOYEE.EmpAge as '年龄
from EMPLOYEE,WORKS
where EMPLOYEE.EmpAge>=40
and EMPLOYEE.EmpNo=WORKS.EmpNo
group by EMPLOYEE.EmpNo,EMPLOYEE.EmpName,EMPLOYEE.EmpSex,EMPLOYEE.EmpAge
```

结果 消息

	员工号	员工姓名	性别	年龄	总工资
1	E03	王五	女	42	3500
2	E05	陈七	男	51	2000

最后，在 Form1 的 Load 方法中，将这个查询结果的语句，后面补个 order by '总工资' desc，按题目要求工资按从大到小顺序排列，再构造到 C#窗体的 listview3 中就好，完成的结果如下：



这题，相对来说，没有这么多代码量，耗时比较少，就是思想过程有点复杂，不过理解了之后就是送分的。

建议先做，分也不少，也简单。

（4）最后一题，也是差不多性质，只是构造出来的 SQL 语句更加复杂。

它要查询至少具有两份工作员工的姓名和其公司名，我们还是不管，先跟上题一样，我们先将索要需求的字段，查询出来的说，之后，同样聚类，只是这次聚类的方式不同，我们仅需要将查询结果中，出现 相同的 员工姓名 的行，合起来，合的过程中，行与行之间的 公司名 字段是不同，这个字段通过，计算出现次数合并，其余字段不要。

因此，Sql 语句演变成如下：

[sql] view plain copy

print?

```sql
1. select EmpName,count(CmpName) as 'CmpNum'
2. from EMPLOYEE,WORKS,COMPANY
3. where EMPLOYEE.EmpNo=WORKS.EmpNo
4. and COMPANY.CmpNo=WORKS.CmpNo
5. group by EmpName
6. having count(CmpName)>1
```

运行结果如下：

	EmpName	CmpNum
1	李四	2
2	张三	2

这显然还不是最后的查询结果，但离查询结果已经很接近了。

只要设这个查询结果为表 t1，再与固有 EMPLOYEE、WORKS、COMPANY 三张表连接起来查询即可，连接条件补上 t1 的 EmpName 等于 EMPLOYEE 的 EmpName 完事，

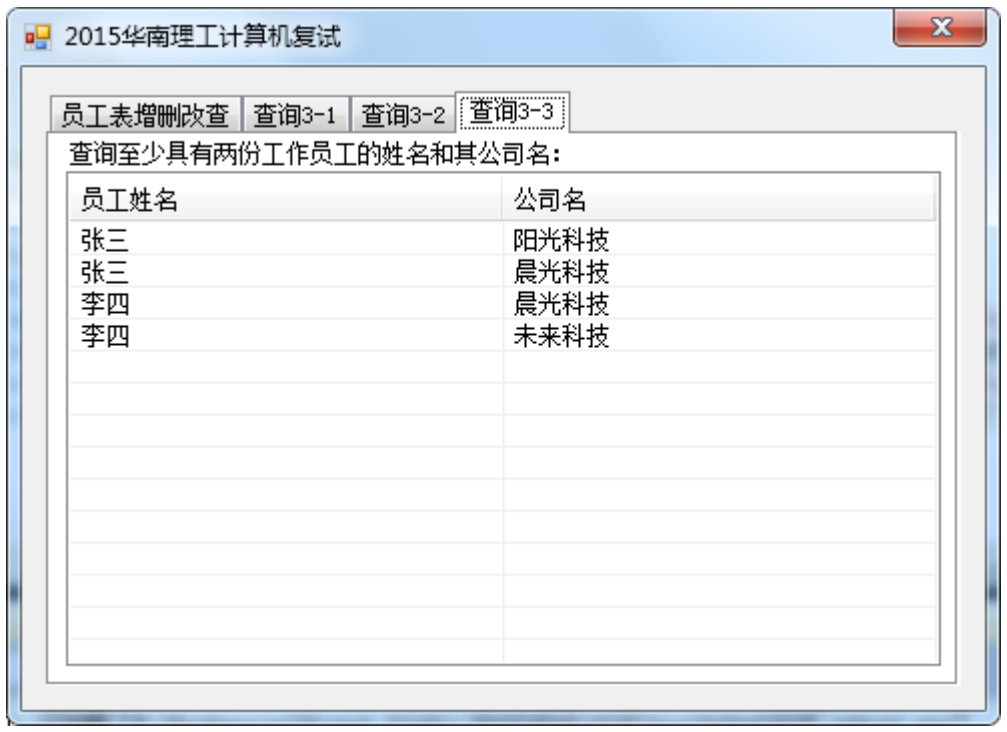
因此，最终的 Sql 语句演变成这个样子：

[sql] view plain copy

print?

```
select EMPLOYEE.EmpName,COMPANY.CmpName from EMPLOYEE,COMPANY,WORKS,(
select EmpName,count(CmpName) as 'CmpNum'
from EMPLOYEE,WORKS,COMPANY
where EMPLOYEE.EmpNo=WORKS.EmpNo
and COMPANY.CmpNo=WORKS.CmpNo
group by EmpName
having count(CmpName)>1
) as t1
where EMPLOYEE.EmpNo=WORKS.EmpNo
and COMPANY.CmpNo=WORKS.CmpNo
and EMPLOYEE.EmpName=t1.EmpName
```

就是一个 4 表连接查询而已，不过其中一个表，是我们的查询结果，最终构造的 C#的 listview4，结果如下：

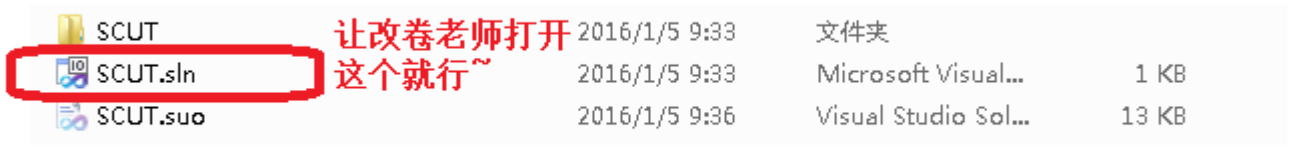


至此整个程序做完。

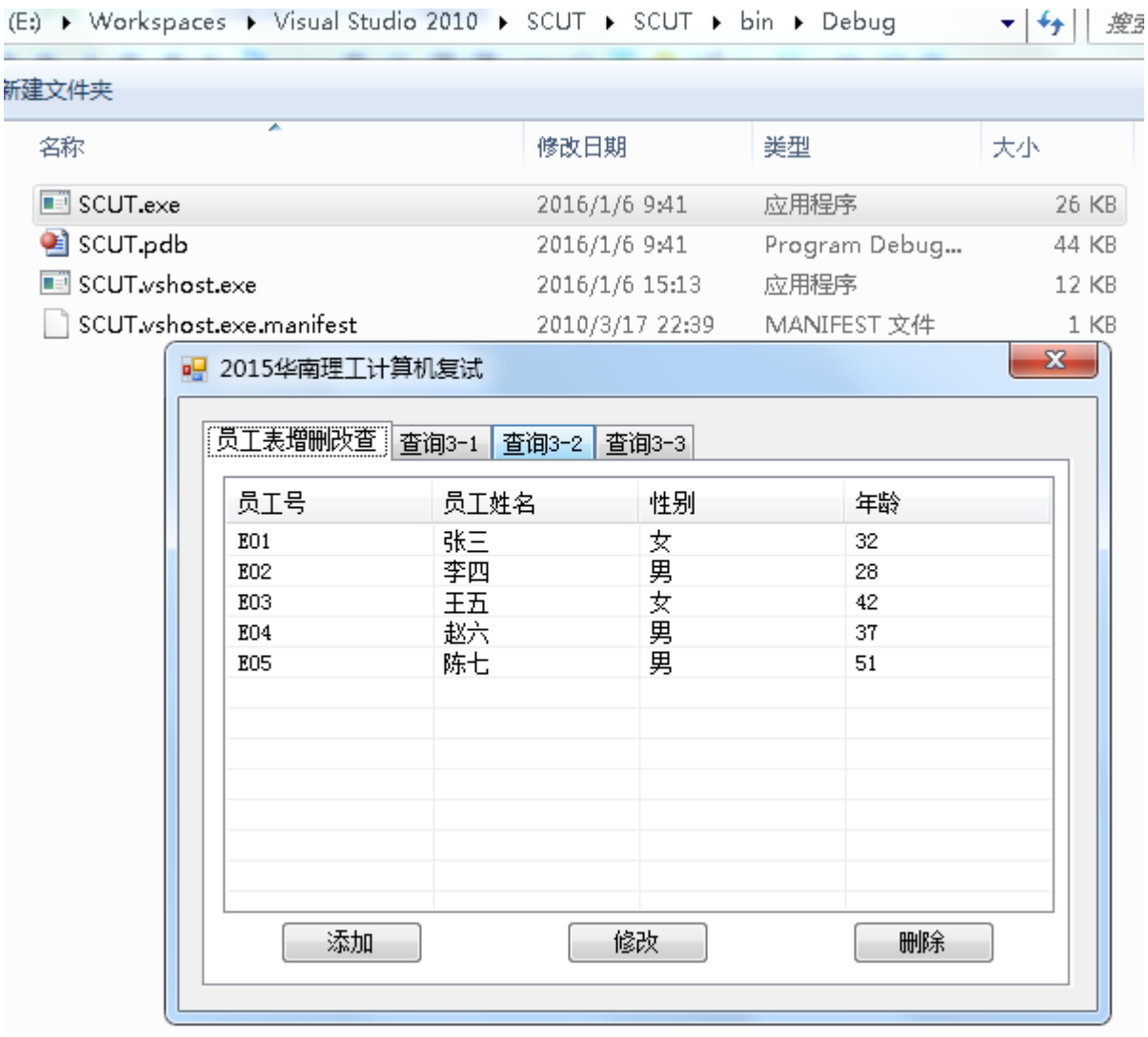
不过题目还没有完成，我们要回过头来完成第一题的文档。

首先，打开你 C#解决方案所在的文件夹：

1、在这个文件夹中，直接打开.sln 就能够重新进入源代码的编辑界面，因此你在文档中，写好打开这个文件，进入解决方案，再点击左侧的各个.cs 文件，能够读到你的源代码。



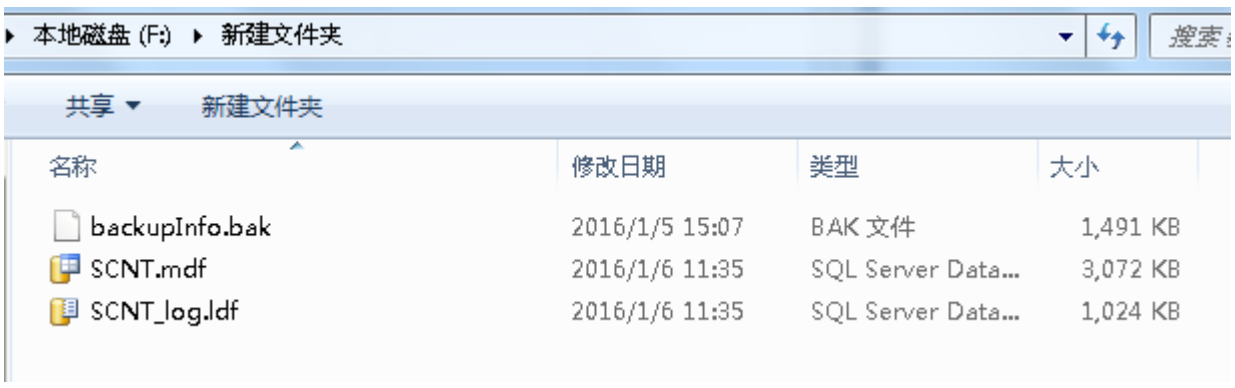
2、关于直接运行的 exe，在 SCUT（这里应该会变成你的名字）\SCUT（这里应该会变成你的名字）\bin\Debug 下的.exe，这一点也是需要在文档中，给改卷老师提到的。



这个文件夹只要你写程序，编译过就会自动生成。根本不用浪费时间去生成什么 exe。

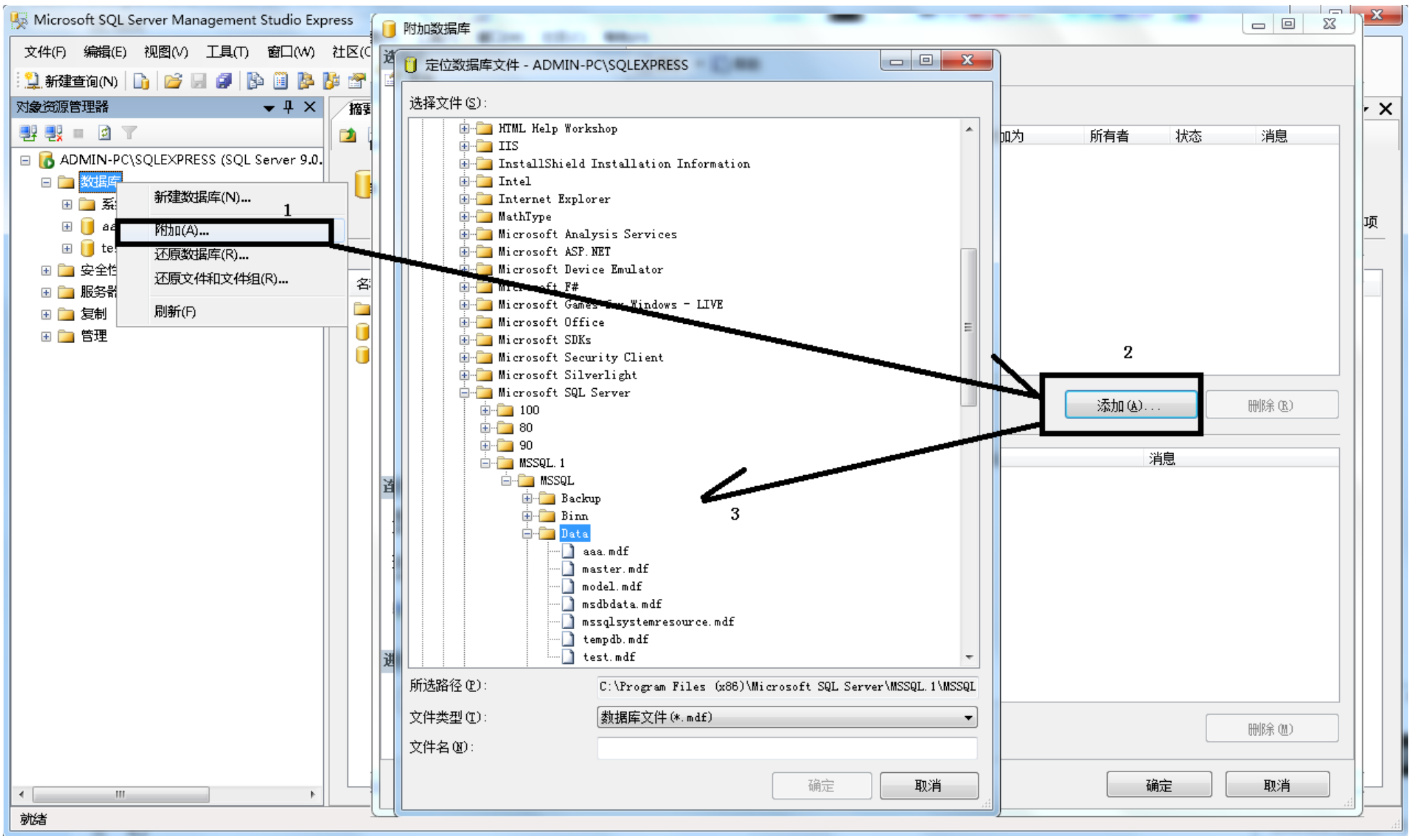
其实一般情况做窗体，将 Debug 文件夹，重命名拖给用户，让用户按装好.NET Framework 2.0，就 OK

3、最后，数据库，先说明你的数据库、备份，在什么什么文件夹：



然后写，数据库，直接负载于 Sql Server 中，不设密码，开放权限，直接通过 Windows 身份登录就能够查看。

如不能发现，请根据如下图的过程，将上图的 mdf，附加到 Sql Server 中，即可。



反正整个流程就是这样，当然试题、数据是每年不同的，关键是理解其中的方法。

在考试过程不能上网、就只有 1.5 小时，自己好好背背关键代码，好自为之。

我整个程序做下来，用的时间远多于 1.5 小时，因为这个软件规模实在是太大了。

当然整个过程的核心是不变了，SQL 查好->C#读->C#构造数据到前台，万变不离其宗。

最后，祝各位通过考研初试，有机会到华南理工复试的计算机考生，在考研复试的过程中，没有任何意外，稳稳当当，别说做完，能够在 1.5 小时中，完成上述 80%的内容，基本稳了。