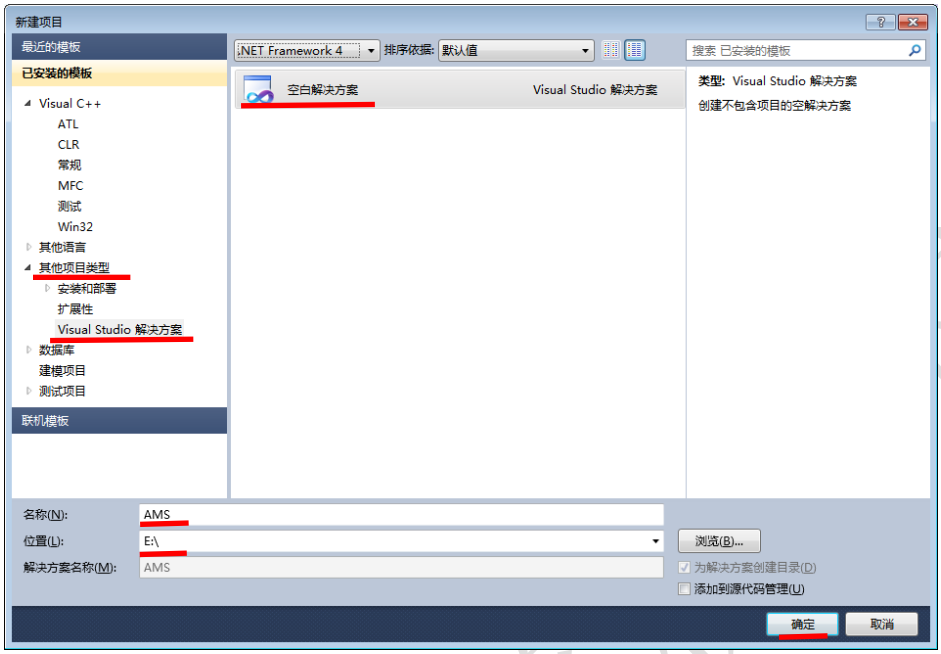
一．创建解决方案

1． 打开 VS2010；

2． 从菜单中选“文件→新建→项目”，出现对话框；

3．对话框中选“其他项目类型→Visual Studio 解决方案→空白解决方案”，输入解决方案名称为“AMS”,设置路径（根据自己实际情况，这里是 E 盘）；

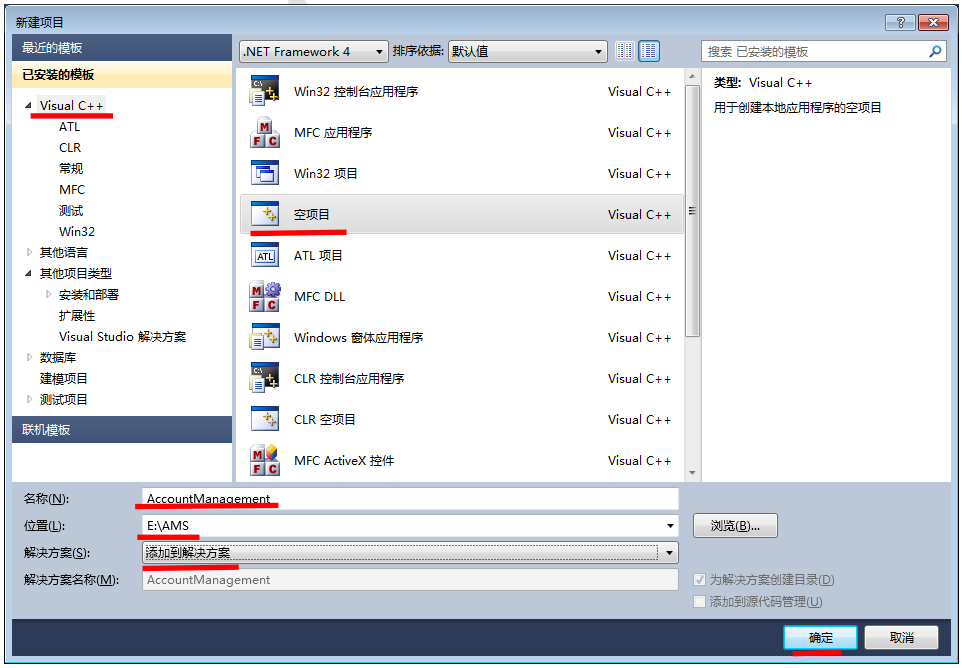


4.“确定”后，会打开 VS 下刚刚创建的 AMS 解决方案，同时在 E 盘会自动生成 AMS 解决方案文件夹，文件夹中自动生成 AMS.sln 文件。

二．创建工程

1.从菜单中选“文件→新建→项目”，出现对话框；

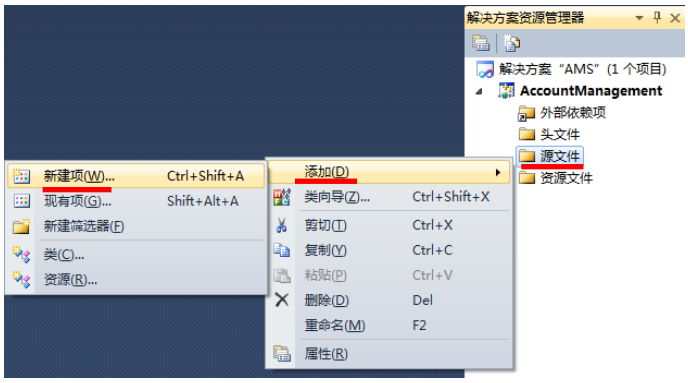
2. 对话框中选“Visual C++→空项目”，输入工程名称为“AccountManagement”,设置路径（前面生成的解决方案路径 E:\AMS），选择“添加到解决方案”；



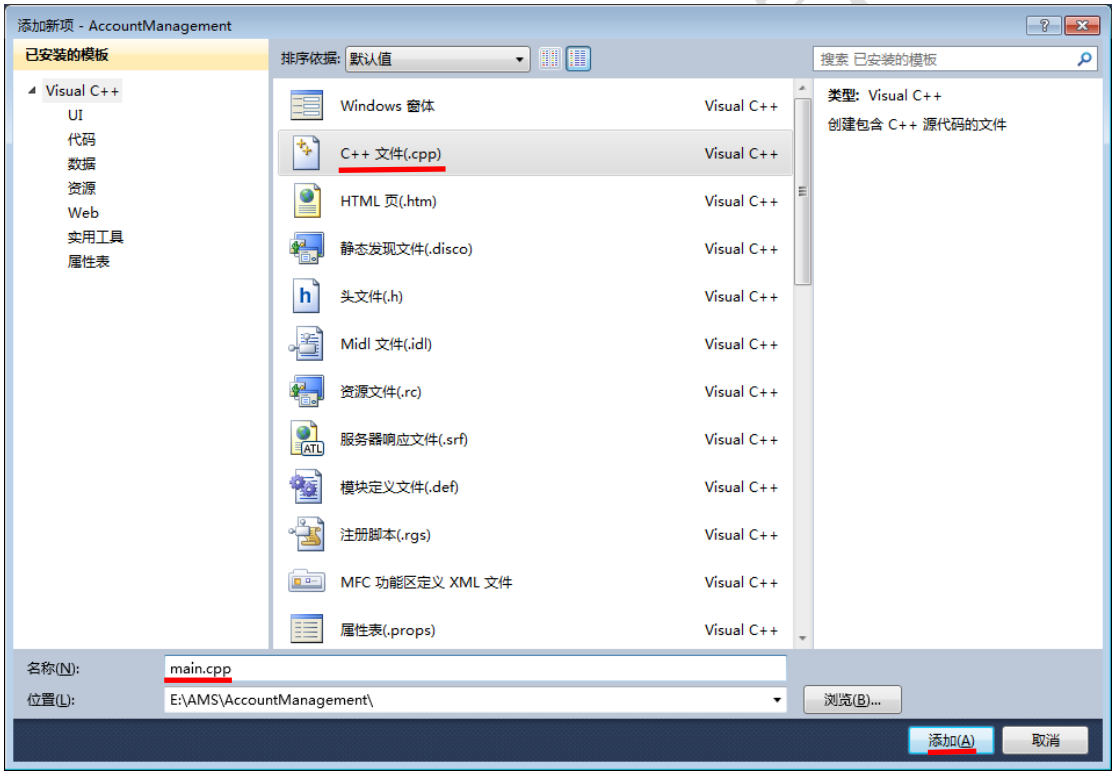
3. “确定”后，会打开刚刚创建的 AccountManagement 工程，同时 AMS 文件夹下会生成 AccountManagement 工程文件夹，以及相关文件。

三．添加主文件

1.在“解决方案资源管理器”下的“源文件”目录上，单击右键，选择“添加→新建项”， 出现对话框；



2. 对话框中选“C++文件（.cpp）”,输入名称为 main.cpp；



3. “添加”后，打开 main.cpp 的编辑窗口，同时添加到“解决方案资源管理器”下的“源文件”目录下。

四．添加编辑主文件代码

在主文件编辑窗口输入以下代码：

```cpp
#include <iostream>
using namespace std;

//[函数名]   main
//[功能]     程序入口函数
//[参数]     void
//[返回值]   0: 程序正常退出; 非零:程序异常
int main()
{
    //输入的菜单项编号
    int  nSelection;

    cout << endl;
    cout << "★欢迎进入计费管理系统★" << endl;
    cout << endl;

    do
    {
        //输出系统菜单
        cout << "--------计费系统菜单--------" << endl << endl;
        cout << "1.添加卡" << endl;
        cout << "2.查询卡" << endl;
        cout << "3.上机" << endl;
        cout << "4.下机" << endl;
        cout << "5.充值" << endl;
        cout << "6.退费" << endl;
        cout << "7.查询统计" << endl;
        cout << "8.注销卡" << endl;
        cout << "0.退出" << endl << endl;
        cout << "请选择菜单项编号（0~8）：";

        nSelection=-1;
        // 输入菜单项编号
        cin >> nSelection;
        cin.clear();
        cin.sync();
        //输出选择的子菜单
        switch(nSelection)
        {
            case 1:
            {
                cout << endl << "-------添加卡--------" << endl << endl;
                break;
            }
            case 2:
            {
                cout << endl << "-------查询卡--------" << endl << endl;
                break;
            }
            case 3:
            {
                cout << endl << "-------上机----------" << endl << endl;
                break;
            }
            case 4:
            {
                cout << endl << "-------下机----------" << endl << endl;
                break;
            }
            case 5:
            {
                cout << endl << "-------充值----------" << endl << endl;
                break;
            }
```

```
64          case 6:
65          {
66              cout << endl << "-------退费----------" << endl << endl;
67              break;
68          }
69        | case 7:
70          {
71              cout << endl << "-----查询统计--------" << endl << endl;
72              break;
73          }
74          case 8:
75          {
76              cout << endl << "------注销卡---------" << endl << endl;
77              break;
78          }
79          case 0:
80          {
81              cout << endl << "谢谢你使用本系统！" << endl << endl;
82              break;
83          }
84          default:
85          {
86              cout << "输入的菜单编号错误！\n请重新输入！\n";
87              break;
88          }
89      }
90      cout << endl;
91    }while(nSelection !=0);
92
93    return 0;
94  }
```

思考：

1. 代码 `#include <iostream>`
   `using namespace std;`
2. 代码 `cin.clear();`
   `cin.sync();`

五．编译并连接程序

1. 选择“生成→生成解决方案”，在 AMS 文件夹中出现 Debug 文件夹，其中生成 AccountManagement.exe 文件
2. 程序代码修改后需要重新编译链接

六．运行程序 和测试

1. 选择“调试→开始执行”，或快捷键 Ctrl+F5
2. 选择“调试→启动调试”，会同时打开几个调试窗口
3. 系统的测试应该包含两个方面，即：

- 功能正确测试：正常输入，检测是否能得到预期的正确结果。
- 错误测试：错误输入，包括数据内容、数据格式异常等，检测系统是否能给出正确的提示，没有非正常退出。

七．封装系统菜单输出函数

1.将上面虚线中代码封装到 outputMenu（）函数中，主函数 main（）中对应位置直接调用 outputMenu（）；

2.主函数前面添加 outputMenu（）函数声明。

调整后代码如下：

```
#include <iostream>
using namespace std;
//函数声明
void outputMenu();

//[函数名]   main
//[功能]     程序入口函数
//[参数]     void
//[返回值]   0：程序正常退出；非零:程序异常
int main()
{
    //输入的菜单项编号
    int  nSelection;

    cout << endl;
    cout << "★欢迎进入计费管理系统★" << endl;
    cout << endl;

   do
  {    //输出系统菜单
       outputMenu();

       nSelection=-1;
       // 输入菜单项编号
       cin >> nSelection;
       cin.clear();
       cin.sync();
       //输出选择的子菜单
       switch(nSelection)
      {
          case 1:
         {
              cout << endl << "-------添加卡--------" << endl << endl;
              break;
          }

          case 2:
         {
              cout << endl << "-------查询卡--------" << endl << endl;
              break;
         }
          case 3:
         {
             cout << endl << "-------上机----------" << endl << endl;
             break;
         }
         case 4:
        {
             cout << endl << "-------下机----------" << endl << endl;
             break;
         }
         case 5:
         {
             cout << endl << "-------充值----------" << endl << endl;
             break;
         }
         case 6:
         {
             cout << endl << "-------退费----------" << endl << endl;
             break;
         }
         case 7:
        {
             cout << endl << "-----查询统计--------" << endl << endl;
             break;
         }
         case 8:
        {
             cout << endl << "------注销卡---------" << endl << endl;
             break;
          }
```

```
71          case 0:
72          {
73              cout << endl << "谢谢你使用本系统！" << endl << endl;
74              break;
75          }
76          default:
77          {
78              cout << "输入的菜单编号错误！\n请重新输入！\n";
79              break;
80          }
81      }
82      cout << endl;
83  }while(nSelection !=0);
84
85  return 0;
86 }
87
88 //[函数名]   outputMenu
89 //[功能]     输出系统菜单
90 //[参数]     void
91 //[返回值]   void
92 void outputMenu()
93 {
94     //输出系统菜单
95     cout << "--------计费系统菜单--------" << endl << endl;
96     cout << "1.添加卡" << endl;
97     cout << "2.查询卡" << endl;
98     cout << "3.上机" << endl;
99     cout << "4.下机" << endl;
100    cout << "5.充值" << endl;
101    cout << "6.退费" << endl;
102    cout << "7.查询统计" << endl;
103    cout << "8.注销卡" << endl;
104    cout << "0.退出" << endl << endl;
105    cout << "请选择菜单项编号（0~8）：";
106 }
```

思考：

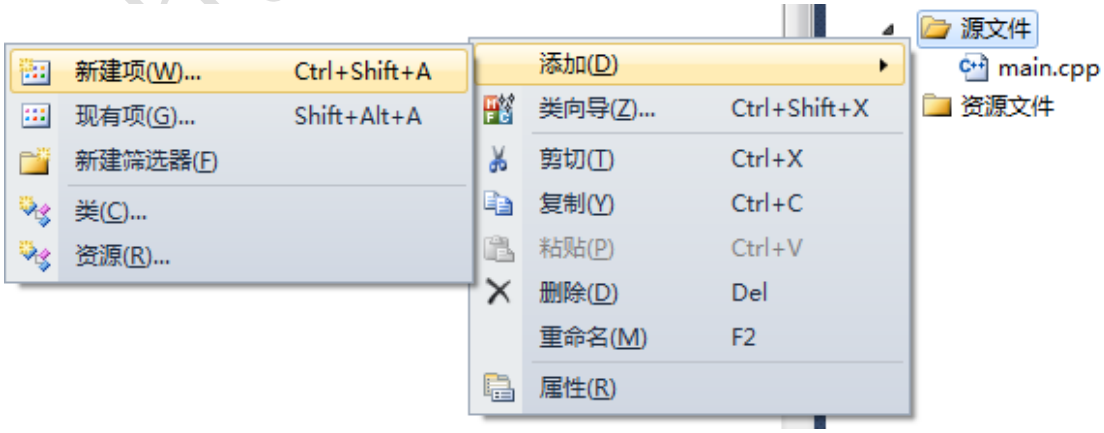
1） 封装部分程序代码的好处？

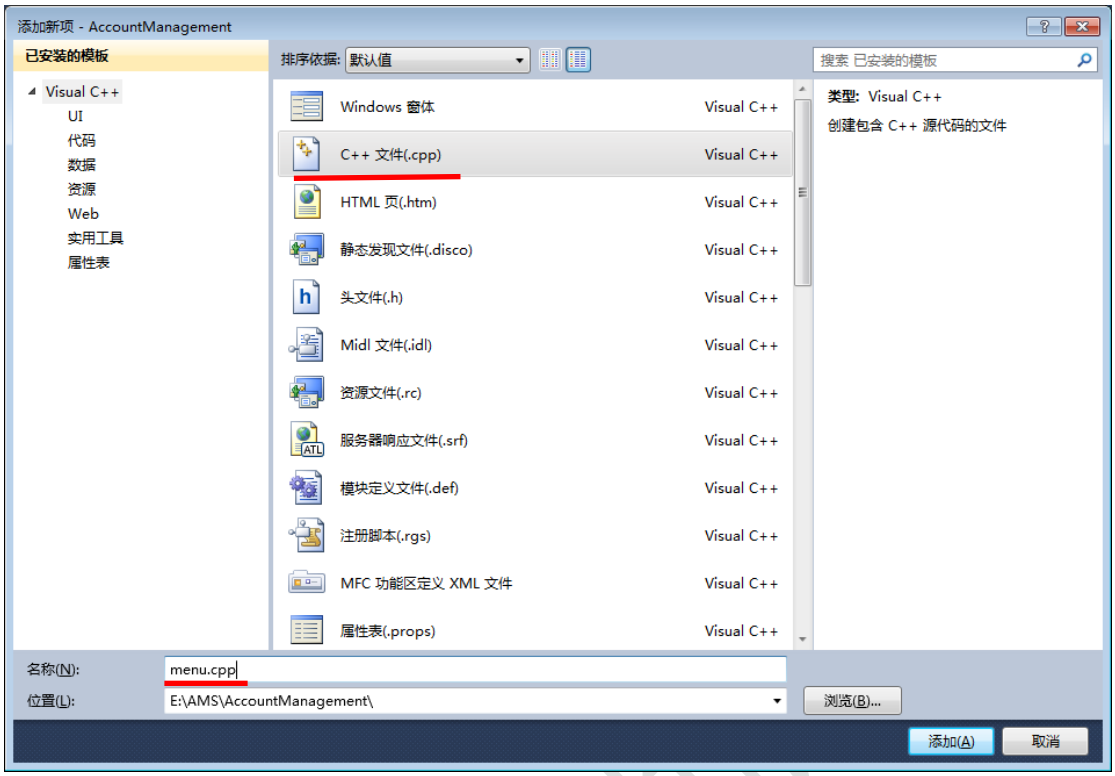
2） 主函数前面为什么要有函数声明？什么时候需要函数声明？什么时候不需要？

3. 重新编译连接，重新调试运行

八．优化程序结构

1. 添加 menu.cpp 文件，将与用户界面输入\输出相关的函数放在这里（目前只有 outputMenu 函数），即将前面 main.cpp 文件中 outputMenu 函数移过来





menu.cpp 中代码如下：

menu.h main.cpp menu.cpp

(全局范围)

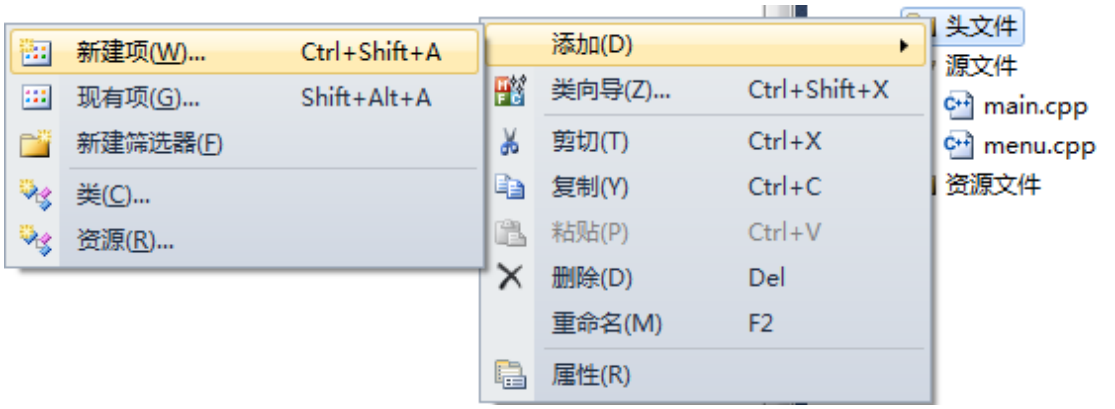
```cpp
#include <iostream>
using namespace std;

//[函数名]   outputMenu
//[功能]     输出系统菜单
//[参数]     void
//[返回值]   void
void outputMenu()
{
    //输出系统菜单
    cout << "--------计费系统菜单--------" << endl << endl;
    cout << "1.添加卡" << endl;
    cout << "2.查询卡" << endl;
    cout << "3.上机" << endl;
    cout << "4.下机" << endl;
    cout << "5.充值" << endl;
    cout << "6.退费" << endl;
    cout << "7.查询统计" << endl;
    cout << "8.注销卡" << endl;
    cout << "0.退出" << endl << endl;
    cout << "请选择菜单项编号（0~8）：";
}
```

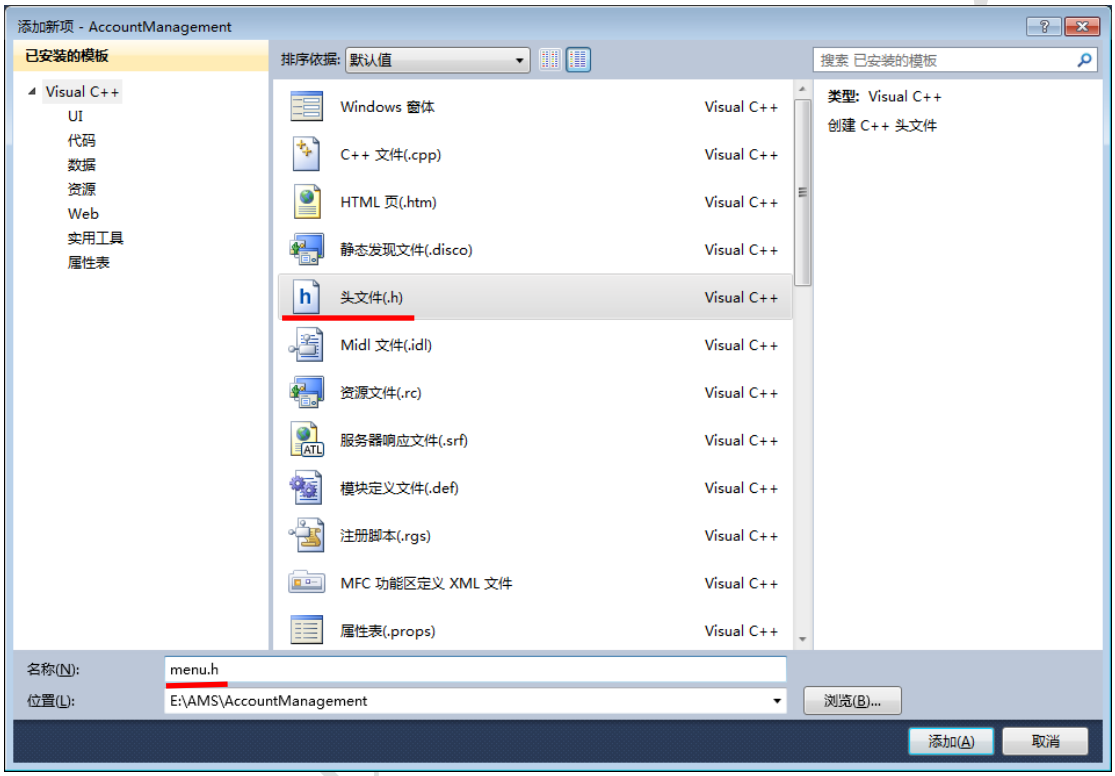
思考：

前面为什么添加 `#include <iostream>`

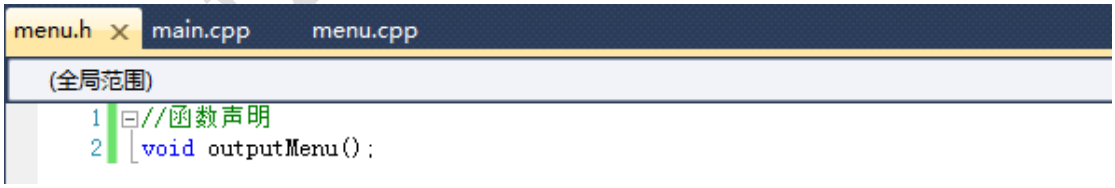
`using namespace std;`

2.添加 menu.h 文件，将 menu.cpp 中对应函数的声明放着这里（目前只有 outputMenu 函数）





menu.h 中代码如下：



3.主文件 main.cpp 中移走了前面的 outputMenu 函数声明，移走了后面的 outputMenu 函数，添加包含 menu.h 头文件。代码如下：

main.cpp menu.cpp

局范围)

```
#include <iostream>

#include"menu.h"

using namespace std;

//void void outputMenu();  //移到头文件

//[函数名]   main
//[功能]     程序入口函数
//[参数]     void
//[返回值]   0: 程序正常退出；非零:程序异常
int main()
{
    //输入的菜单项编号
    int  nSelection;

    cout << endl;
    cout << "★欢迎进入计费管理系统★" << endl;
    cout << endl;

   do
   {    //输出系统菜单
        outputMenu();

```

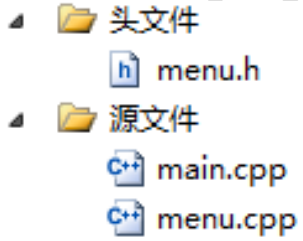
4 重新编译连接，重新调试运行

思考：1).cpp 文件和.h 文件的作用?

2) .cpp 文件和.h 文件是不是一对一的？

3) 什么时候要包含.h 文件？

优化后整个项目包括三个文件（一个头文件，两个 CPP 文件）



menu.cpp

```
#include <iostream>
using namespace std;

//[函数名]   outputMenu
//[功能]     输出系统菜单
//[参数]     void
//[返回值]   void
void outputMenu()
{
    //输出系统菜单
    cout << "--------计费系统菜单--------" << endl;
    cout << "1.添加卡" << endl;
    cout << "2.查询卡" << endl;
    cout << "3.上机" << endl;
    cout << "4.下机" << endl;
    cout << "5.充值" << endl;
    cout << "6.退费" << endl;
    cout << "7.查询统计" << endl;
    cout << "8.注销卡" << endl;
    cout << "0.退出" << endl << endl;
    cout << "请选择菜单项编号（0~8）：";
}
```

menu.h

```
//函数声明
void outputMenu();
```

main.cpp

```
#include <iostream>

#include"menu.h"

using namespace std;

//void void outputMenu();  //移到头文件

//[函数名]   main
//[功能]     程序入口函数
//[参数]     void
//[返回值]   0: 程序正常退出；非零:程序异常
int main()
{
    //输入的菜单项编号
    int  nSelection;

    cout << endl;
    cout << "★欢迎进入计费管理系统★" << endl;
    cout << endl;

   do
   {
       //输出系统菜单
       outputMenu();

       nSelection=-1;
       // 输入菜单项编号
       cin >> nSelection;
       cin.clear();
```

九．总结

本次任务建立的层次结构和调用关系