|  |
| --- |
| 清华大学 电子工程系 计算机程序设计基础课程 |
| 犀利人事管理系统 |
| 小学期程序设计 实例一 |

|  |
| --- |
| 池雨泽  2013-6-18 |

# 犀利人事管理系统

索引

[犀利人事管理系统 1](#_Toc360964658)

[系统需求分析 1](#_Toc360964659)

[总体设计 2](#_Toc360964660)

[详细设计 3](#_Toc360964661)

[系统调试 6](#_Toc360964662)

[结果分析 6](#_Toc360964663)

[总结 7](#_Toc360964664)

[附录1：源程序清单 8](#_Toc360964665)

[附录2：评分表 31](#_Toc360964666)

## 系统需求分析

一个好的公司总是有很多员工需要管理。犀利哥的公司有许多员工，每个员工拥有不同的属性，总的来说分为以下几种：

1. 普通雇员；
2. 项目经理；
3. 销售经理；
4. 销售人员；
5. 技术人员。

公司可能扩大规模、增加员工数目，也可能奖励表现好的员工对其进行升职加薪等，或是对表现不好的员工进行处罚或者直接解雇。公司可能需要根据需要查询员工信息或进行整体上的统计。据此，提供功能如下：

1. 录入数据；可以选择录入员工的类别和相应的属性；
2. 修改数据：可以修改已经录入的员工属性；
3. 查询数据：可以查询已经录入的员工属性；
4. 删除数据：可以删除已经录入的员工资料；
5. 统计数据：可以统计已经录入的员工属性。

犀利哥可以通过以上功能录入信息、修改信息、查询信息、删除信息、整理统计出所要了解的信息，除了要实现上述的基本功能之外，本系统还应该在细节上下工夫，使用户使用方便，在使用的过程中保持一个愉快的心情。犀利人事管理系统有广大的用户群，这其中有犀利公司、有天仙夫妇名下其他公司，还可将之出售给其他公司的管理人员，也可以供给一般用户使用。总之该系统可以满足用户需求，实现对其旗下人员的人事管理工作。

## 总体设计

系统提供五大功能：录入、修改、查询、删除、统计。

在录入员工信息时根据系统提示选择员工类别逐一输入。每输入完一条信息，系统会提示是否继续输入，用户可以选择继续或返回主菜单。

在修改员工信息时，用户首先输入要修改的员工编号，系统会检索。如果不知道有哪些员工，还可以显示所有员工信息。如果系统中有该员工的相关信息，则系统首先会提示用户修改该员工的哪一部分信息，用户可自行选择。如果系统中没有该员工的相关信息，则系统会给相关提示之后返回主菜单，用户可以继续选择操作。

在查询员工信息时，用户可以输入要查询的员工编号，系统检索判断是否存在该员工的信息然后做出相应的提示。也可以直接选择显示全部员工信息。

在删除员工信息时，用户需要输入欲删除的员工编号，系统会检索，如果存在该员工则将之删除并给出相应提示。总之，用户可以方便的在系统提示下进行使用。

犀利人事管理系统中功能模块图：

图 1 犀利人事管理系统中模块之间的关系

## 详细设计

员工分为5类，其继承关系如下：

图 2 犀利人事管理系统中类的关系

犀利人事管理系统各功能模块的实现：

1. 录入模块
2. 修改模块
3. 查询模块
4. 删除模块
5. 统计模块

## 系统调试

系统调试遇到的第一个问题是数据文件不稳定。经过对系统结构特别是对数据文件读写过程的分析，确认问题出现在直接将对象用二进制写入文件这一方法上。由于使用了虚函数等与运行相关的多态，类的对象中有地址，一旦将这些地址写入文件，再读取时很有可能导致内存泄露而崩溃。为此，我修改了程序数据文件读写框架，为每个类都添加了从输入输出流以二进制形式读取写入数据的成员函数。这一过程中，我采用了在主函数中调用读取跟保存函数的方法，而这两个函数分别遍历程序数据库，实现对每一个对象的读取或写入。为了程序简洁清晰，后续开发中我将主函数单独放在一个文件中，其他实用函数单独形成文件，增加程序可读性。调试中遇到的其他问题还有命令行UI的问题，例如无法顺利退出某一层菜单，这是由于循环无法正确跳出导致的。在需要结束运行的地方适当地加break或return语句就可以解决。经过反复的调试，除了一些命令行本身的限制例如无法屏蔽输入法、无法同时监视Esc与键盘输入等用户体验的细节问题外，没有已知的Bug。通过这一工程的调试，我认识到类对象不能够轻易地直接二进制写入文件，通常要单独设置读写方法来保证稳定和安全，这为今后的开发提供了指导。

## 结果分析

在源代码的组织上，第一次将源代码与工程文件分开保存，这样既便于跨平台编译，又利于保证安全防止误操作。类的定义单独保存为类名为文件名的.h和.cpp中，且头文件使用编译头保护，既避免了重复编译又减少了编译的工作量。

在数据的组织上，在直接二进制保存类对象的思路彻底失败之后，立即改为为每个类设置IO函数，保证了数据结构的稳定读写。美中不足在于，代码无法动态确定指针类型，只能将void指针通过switch强制转换为各类指针进行各种操作。这严重增加了重复的操作。

在代码的细节上，由于本学期学习的是输入输出流，而且大部分代码是在学期中逐渐积攒起来的，因此输入输出均采用了C++风格的流输入输出，无视掉了Google C++风格指引的建议。对于异常处理，根据Google的建议我没有采用C++风格的try-throw-catch结构，而是对于可以预见的非致命错误进行提示后继续运行，对于不可预见的致命性错误使用cerr输出后暂停程序。调试中并没有因为异常处理带来额外的麻烦，大部分的提示也都没有出现。

文件编码仍然是个大问题。果断地使用了GBK……上一次使用了先将代码页调成65001然后使用UTF-8的方法，虽然移植性可能好一点，但是调代码页之后Windows的命令行变得更难看了……这次全部编码都默认地使用了CP936，方便Windows上的运行。移植性神马的反正……不用移植就先不管了……

编译环境的选择。使用的仍然是Code::Blocks+自带MinGW的组合。32位编译器，在Windows 7 x64上编译并测试运行通过。调用了非标准库函数和C++ 11的特性，使用VS特别是老的VS编译不保证通过。[保证通不过我会说吗]

## 总结

虽然不是第一次写命令行UI的程序，也不是第一次写一千行以上的程序，但这次的程序开发还是给我带来了很大的收获。最大的收获就是类对象的IO吧，不能直接写到文件里的，只能保存数据成员，特别是有虚函数这种情况。其他的就是一些细节上比较零碎的收获了，总之有了这次开发经验，今后对C++的驾驭应该会更加轻车熟路。感谢老师为大家提供了这样一个提高自己的机会！

## 附录1：源程序清单

Code::Blocks工程文件:

|  |
| --- |
| Filename 犀利人事管理系统/project.codeblocks/personnelmanaging.cbp  1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  2 <CodeBlocks\_project\_file>  3 <FileVersion major="1" minor="6" />  4 <Project>  5 <Option title="犀利人事管理系统" />  6 <Option pch\_mode="2" />  7 <Option compiler="gcc" />  8 <Build>  9 <Target title="Debug">  10 <Option output="bin/Debug/犀利人事管理系统" prefix\_auto="1" extension\_auto="1" />  11 <Option object\_output="obj/Debug/" />  12 <Option type="1" />  13 <Option compiler="gcc" />  14 <Compiler>  15 <Add option="-std=c++11" />  16 <Add option="-Wall" />  17 <Add option="-g" />  18 </Compiler>  19 </Target>  20 <Target title="Release">  21 <Option output="bin/Release/犀利人事管理系统" prefix\_auto="1" extension\_auto="1" />  22 <Option object\_output="obj/Release/" />  23 <Option type="1" />  24 <Option compiler="gcc" />  25 <Compiler>  26 <Add option="-Os" />  27 <Add option="-O3" />  28 <Add option="-std=c++11" />  29 <Add option="-Wall" />  30 </Compiler>  31 <Linker>  32 <Add option="-s" />  33 </Linker>  34 </Target>  35 </Build>  36 <Compiler>  37 <Add option="-Wall" />  38 </Compiler>  39 <Unit filename="../codes/classes/employee.cpp" />  40 <Unit filename="../codes/classes/employee.h" />  41 <Unit filename="../codes/classes/manager.cpp" />  42 <Unit filename="../codes/classes/manager.h" />  43 <Unit filename="../codes/classes/salemanager.cpp" />  44 <Unit filename="../codes/classes/salemanager.h" />  45 <Unit filename="../codes/classes/salesman.cpp" />  46 <Unit filename="../codes/classes/salesman.h" />  47 <Unit filename="../codes/classes/technician.cpp" />  48 <Unit filename="../codes/classes/technician.h" />  49 <Unit filename="../codes/main.cpp" />  50 <Unit filename="../codes/utilities.cpp" />  51 <Unit filename="../codes/utilities.h" />  52 <Extensions>  53 <code\_completion />  54 <envvars />  55 <debugger />  56 <lib\_finder disable\_auto="1" />  57 </Extensions>  58 </Project>  59 </CodeBlocks\_project\_file> |

main.cpp:

|  |
| --- |
| Filename犀利人事管理系统/codes/main.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<cstdlib>  8 #include<cstring>  9  10 #include<iostream>  11 #include<vector>  12  13 #include<conio.h>  14  15 #include"classes\manager.h"  16 #include"classes\technician.h"  17 #include"classes\salesman.h"  18 #include"classes\salemanager.h"  19 #include"utilities.h"  20  21 using namespace std;  22  23 extern Data data;  24  25 int main()  26 {  27 system("title 犀利人事管理系统");  28 cout<<" 00000000000000000000000000 100 000\n 00000000000000000000000001 00000000 100\n 001 00 00000000000000 00\n\  29 001 00 00000 000 00\n 00000000000000000000000001 000 000 00\n 00 000 000 000 00\n 00 000 000 000 000 000 00\n 00 100000 000 00000001 000000000000000000 000 00\n 00 0 000 00 000000000000000000 000 00\n 00 0 000 0 000 000 00\n 00 00000000 000 000000 00000 000 00\n 00 001 000 0000000 000 00\n 000 000 000 0000000000 000 00\n 000 000 000 1001000 0000 000 00\n 000 00000000000000000000 000 000 1000 000 00\n 00 000 000 0000 000 1 000 00\n 000 00 000 0000 000 000 00\n 000 000 0000 000 00\n 00 0000000000000000000000000 00 000 00\n 000 000 000 00\n 0000 000 000 000\n 00 000 000 000000\n 000 000 0000\n";  30 if(LoadData())  31 {  32 Pause();  33 return -1;  34 }  35 Wait();  36 Wait();  37 for(;;)  38 {  39 system("cls");  40 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  41 cout<<"请选择欲使用的功能:\n";  42 cout<<" 1.数据录入\n";  43 cout<<" 2.数据修改\n";  44 cout<<" 3.数据查询\n";  45 cout<<" 4.数据删除\n";  46 cout<<" 5.数据统计\n";  47 cout<<" Q.退出\n";  48  49 fflush(stdin);  50 switch(getch())  51 {  52 case '1':  53 Input();  54 break;  55 case '2':  56 Modify();  57 break;  58 case '3':  59 Search();  60 break;  61 case '4':  62 Delete();  63 break;  64 case '5':  65 Statistic();  66 break;  67 case 'Q':case 'q':  68 SaveData();  69 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  70 {  71 switch(iter->type)  72 {  73 case kEmployee:  74 delete ((Employee\*)(iter->pointer));  75 break;  76 case kManager:  77 delete ((Manager\*)(iter->pointer));  78 break;  79 case kSalemanager:  80 delete ((Salemanager\*)(iter->pointer));  81 break;  82 case kSalesman:  83 delete ((Salesman\*)(iter->pointer));  84 break;  85 case kTechnician:  86 delete ((Technician\*)(iter->pointer));  87 break;  88 default:  89 cerr<<"数据结构错误。\n";  90 Pause();  91 break;  92 }  93 }  94 cout<<"感谢您的使用，再见。\n";  95 Wait();  96 system("cls");  97 return 0;  98 default:  99 cerr<<"对不起，输入错误；请重新输入。";  100 Wait();  101 break;  102 }  103 fflush(stdin);  104 }  105 SaveData();  106 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  107 {  108 switch(iter->type)  109 {  110 case kEmployee:  111 delete ((Employee\*)(iter->pointer));  112 break;  113 case kManager:  114 delete ((Manager\*)(iter->pointer));  115 break;  116 case kSalemanager:  117 delete ((Salemanager\*)(iter->pointer));  118 break;  119 case kSalesman:  120 delete ((Salesman\*)(iter->pointer));  121 break;  122 case kTechnician:  123 delete ((Technician\*)(iter->pointer));  124 break;  125 default:  126 cerr<<"数据结构错误。\n";  127 Pause();  128 break;  129 }  130 }  131 system("cls");  132 return 0;  133 } |

utilities.h:

|  |
| --- |
| Filename犀利人事管理系统/codes/utilities.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef UTILITIES\_H\_  8 #define UTILITIES\_H\_  9  10 #include<cstdio>  11 #include<cstdlib>  12 #include<ctime>  13  14 #include<iostream>  15 #include<vector>  16  17 #include<conio.h>  18  19 #include"classes\manager.h"  20 #include"classes\technician.h"  21 #include"classes\salesman.h"  22 #include"classes\salemanager.h"  23  24 int Input();  25 int Modify();  26 int Search();  27 int Statistic();  28 int Delete();  29  30 int LoadData(const char\* filename="save.dat");  31 int SaveData(const char\* filename="save.dat");  32  33 enum Type  34 {  35 kEmployee=1,  36 kManager=2,  37 kSalemanager=3,  38 kSalesman=4,  39 kTechnician=5  40 };  41  42 struct Object  43 {  44 Type type;  45 void\* pointer;  46 };  47  48 typedef std::vector<Object> Data;  49  50  51 inline void Pause()  52 {  53 fflush(stdin);  54 std::cin.clear();  55 std::cin.sync();  56 getch();  57 }  58  59 inline void Wait()  60 {  61 fflush(stdin);  62 std::cin.clear();  63 std::cin.sync();  64 time\_t start,current;  65 time(&start);  66 for(time(&current);current-start<1.0;time(&current));  67 }  68  69 #endif//UTILITIES\_H\_ |

utilities.cpp:

|  |
| --- |
| Filename犀利人事管理系统/codes/utilities.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<cstdlib>  8  9 #include<fstream>  10 #include<iostream>  11 #include<sstream>  12 #include<vector>  13  14 #include<conio.h>  15  16 #include"utilities.h"  17  18 using namespace std;  19  20 Data data;  21 Type read\_type;  22 Object object\_struct;  23  24 int Input()  25 {  26 for(;;)  27 {  28 cin.clear();  29 cin.sync();  30 system("cls");  31 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  32 cout<<"请选择欲录入的员工类别:\n";  33 cout<<" 1.职员 Employee\n";  34 cout<<" 2.经理 Manager\n";  35 cout<<" 3.销售经理 Salemanager\n";  36 cout<<" 4.销售员 Salesman\n";  37 cout<<" 5.技术员 Technician\n";  38 cout<<" B.返回主菜单\n";  39 char choice;  40 choice=getch();  41 fflush(stdin);  42 if(choice=='1')  43 {  44 Employee\* employee=new Employee;  45 if(employee==NULL)  46 {  47 cerr<<"内存分配错误。\n";  48 Pause();  49 }  50 else  51 {  52 cout<<"请分别输入员工等级和月薪:";  53 cin>>\*employee;  54 if(cin.fail())  55 {  56 cerr<<"对不起，输入错误；请重新输入。";  57 Wait();  58 }  59 else  60 {  61 object\_struct={kEmployee,employee};  62 data.push\_back(object\_struct);  63 cout<<"输入成功。\n";  64 Wait();  65 }  66 }  67 }  68 else if(choice=='2')  69 {  70 Manager\* manager=new Manager;  71 if(manager==NULL)  72 {  73 cerr<<"内存分配错误。\n";  74 Pause();  75 }  76 else  77 {  78 object\_struct={kManager,manager};  79 data.push\_back(object\_struct);  80 cout<<"输入成功。\n";  81 Wait();  82 }  83 }  84 else if(choice=='3')  85 {  86 Data salesmen;  87 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  88 {  89 if(iter->type==kSalesman)  90 {  91 salesmen.push\_back(\*iter);  92 }  93 }  94 if(salesmen.size()==0)  95 {  96 cout<<"现在没有销售员，不能录用销售经理。\n";  97 Wait();  98 }  99 else  100 {  101 system("cls");  102 cout<<"将列出现有的的销售员。\n";  103 Wait();  104 for(Data::iterator iter=salesmen.begin();iter!=salesmen.end();++iter)  105 {  106 cout<<\*((Salesman\*)(iter->pointer))<<endl;  107 }  108 cout<<"请在一行内输入欲录用销售经理所管辖的销售员工号:";  109 string temp\_str;  110 getline(cin,temp\_str);  111 vector<Salesman\*> salesmen\_pointer;  112 size\_t length\_salesmen=0;  113 for(istringstream istrin(temp\_str);;)  114 {  115 int num;  116 istrin>>num;  117 if(istrin.fail())  118 {  119 break;  120 }  121 for(Data::iterator iter=salesmen.begin();iter!=salesmen.end();++iter)  122 {  123 if(num==((Salesman\*)(iter->pointer))->individual\_emp\_no())  124 {  125 salesmen\_pointer.push\_back((Salesman\*)(iter->pointer));  126 length\_salesmen+=sizeof(\*((Salesman\*)(iter->pointer)));  127 }  128 }  129 }  130 if(length\_salesmen>0)  131 {  132 Salesman\* salesmen\_array=(Salesman\*)malloc(length\_salesmen);  133 size\_t position=0;  134 for(vector<Salesman\*>::iterator iter=salesmen\_pointer.begin();  135 iter!=salesmen\_pointer.end();++iter)  136 {  137 for(size\_t i=0;i<sizeof(\*\*iter);++i)  138 {  139 \*((char\*)salesmen\_array+position+i)=\*((char\*)(\*iter)+i);  140 }  141 position+=sizeof(\*\*iter);  142 }  143 Salemanager\* salesmanager=new Salemanager(salesmen\_array,salesmen\_pointer.size());  144 if(salesmanager==NULL)  145 {  146 cerr<<"内存分配错误。\n";  147 Pause();  148 }  149 else  150 {  151 object\_struct={kSalemanager,salesmanager};  152 data.push\_back(object\_struct);  153 cout<<"输入成功。\n";  154 Wait();  155 }  156 free(salesmen\_array);  157 }  158 else  159 {  160 cout<<"没有找到匹配的工号。\n";  161 Wait();  162 }  163 }  164 }  165 else if(choice=='4')  166 {  167 Salesman\* salesman=new Salesman;  168 if(salesman==NULL)  169 {  170 cerr<<"内存分配错误。\n";  171 Pause();  172 }  173 else  174 {  175 cout<<"请输入该销售员本月的销售额:";  176 cin>>\*salesman;  177 if(cin.fail())  178 {  179 cerr<<"对不起，输入错误；请重新输入。";  180 Wait();  181 }  182 else  183 {  184 object\_struct={kSalesman,salesman};  185 data.push\_back(object\_struct);  186 cout<<"输入成功。\n";  187 Wait();  188 }  189 }  190 }  191 else if(choice=='5')  192 {  193 Technician\* technician=new Technician;  194 if(technician==NULL)  195 {  196 cerr<<"内存分配错误。\n";  197 Pause();  198 }  199 else  200 {  201 cout<<"请分别该技术员本月的工作时长:";  202 cin>>\*technician;  203 if(cin.fail())  204 {  205 cerr<<"对不起，输入错误；请重新输入。";  206 Wait();  207 }  208 else  209 {  210 object\_struct={kTechnician,technician};  211 data.push\_back(object\_struct);  212 cout<<"输入成功。\n";  213 Wait();  214  215 }  216 }  217 }  218 else if(choice=='B'||choice=='b')  219 {  220 return 0;  221 }  222 else  223 {  224 cerr<<"对不起，输入错误；请重新输入。";  225 Wait();  226 }  227 }  228 return 0;  229 }  230 int Modify()  231 {  232 for(;;)  233 {  234 cin.clear();  235 cin.sync();  236 system("cls");  237 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  238 cout<<"请选择欲使用的功能;输入其他字符可以开始输入工号:\n";  239 cout<<" D.显示当前所有员工\n";  240 cout<<" B.返回主菜单\n";  241 char choice;  242 choice=getch();  243 if(choice=='D'||choice=='d')  244 {  245 system("cls");  246 if(data.empty())  247 {  248 cerr<<"数据空。\n";  249 }  250 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  251 {  252 switch(iter->type)  253 {  254 case kEmployee:  255 cout<<\*((Employee\*)iter->pointer)<<endl;  256 break;  257 case kManager:  258 cout<<\*((Manager\*)iter->pointer)<<endl;  259 break;  260 case kSalemanager:  261 cout<<\*((Salemanager\*)iter->pointer)<<endl;  262 break;  263 case kSalesman:  264 cout<<\*((Salesman\*)iter->pointer)<<endl;  265 break;  266 case kTechnician:  267 cout<<\*((Technician\*)iter->pointer)<<endl;  268 break;  269 default:  270 cerr<<"数据结构错误。\n";  271 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  272 }  273 }  274 Pause();  275 }  276 else if(choice=='B'||choice=='b')  277 {  278 return 0;  279 }  280 else if(choice>='0'&&choice<='9')  281 {  282 int num;  283 cin>>num;  284 if(cin.fail())  285 {  286 cerr<<"对不起，输入错误；请重新输入。";  287 Wait();  288 }  289 else  290 {  291 bool found=false;  292 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  293 {  294 char c;  295 switch(iter->type)  296 {  297 case kEmployee:  298 if(num==((Employee\*)(iter->pointer))->individual\_emp\_no())  299 {  300 if(found)  301 {  302 cerr<<"工号重复。\n";  303 }  304 system("cls");  305 cout<<\*(Employee\*)(iter->pointer)<<endl\  306 <<"请选择欲修改的内容:\n"\  307 <<" 1.等级\n 2.月薪\n";  308 c=getch();  309 switch(c)  310 {  311 case '1':  312 int grade;  313 cout<<"请输入修改后的等级:";  314 cin>>grade;  315 if(cin.fail())  316 {  317 cerr<<"输入错误，请重新输入。\n";  318 }  319 else  320 {  321 ((Employee\*)iter->pointer)->set\_grade(grade);  322 cout<<"修改成功。\n";  323 }  324 break;  325 case '2':  326 int pay;  327 cout<<"请输入修改后的月薪:";  328 cin>>pay;  329 if(cin.fail())  330 {  331 cerr<<"输入错误，请重新输入。\n";  332 }  333 else  334 {  335 ((Employee\*)iter->pointer)->promote(pay,0);  336 cout<<"修改成功。\n";  337 }  338 break;  339 default:  340 cerr<<"输入错误，请重新输入。\n";  341 }  342 found=true;  343 Wait();  344 }  345 break;  346 case kManager:  347 if(num==((Manager\*)(iter->pointer))->individual\_emp\_no())  348 {  349 if(found)  350 {  351 cerr<<"工号重复。\n";  352 }  353 system("cls");  354  355 int pay;  356 cout<<"请输入修改后的月薪:";  357 cin>>pay;  358 if(cin.fail())  359 {  360 cerr<<"输入错误，请重新输入。\n";  361 }  362 else  363 {  364 ((Manager\*)iter->pointer)->promote(pay,0);  365 cout<<"修改成功。\n";  366 }  367  368 found=true;  369 Wait();  370 }  371 break;  372 case kSalemanager:  373 if(num==((Salemanager\*)(iter->pointer))->individual\_emp\_no())  374 {  375 if(found)  376 {  377 cerr<<"工号重复。\n";  378 }  379 system("cls");  380 cout<<\*(Salemanager\*)(iter->pointer)<<endl\  381 <<"请选择欲修改的内容:\n"\  382 <<" 1.固定工资\n 2.提成\n";  383 c=getch();  384 switch(c)  385 {  386 case '1':  387 int pay;  388 cout<<"请输入修改后的固定工资:";  389 cin>>pay;  390 if(cin.fail())  391 {  392 cerr<<"输入错误，请重新输入。\n";  393 }  394 else  395 {  396 ((Salemanager\*)iter->pointer)->promote(pay,0);  397 cout<<"修改成功。\n";  398 }  399 break;  400 case '2':  401 double percentage;  402 cout<<"请输入修改后的提成:";  403 cin>>percentage;  404 if(cin.fail())  405 {  406 cerr<<"输入错误，请重新输入。\n";  407 }  408 else  409 {  410 ((Salemanager\*)iter->pointer)->promote(0,percentage);  411 cout<<"修改成功。\n";  412 }  413 break;  414 default:  415 cerr<<"输入错误，请重新输入。\n";  416 }  417 found=true;  418 Wait();  419 }  420 break;  421 case kSalesman:  422 if(num==((Salesman\*)(iter->pointer))->individual\_emp\_no())  423 {  424 if(found)  425 {  426 cerr<<"工号重复。\n";  427 }  428 system("cls");  429 cout<<\*(Salesman\*)(iter->pointer)<<endl\  430 <<"请选择欲修改的内容:\n"\  431 <<" 1.销售额\n 2.提成\n";  432 c=getch();  433 switch(c)  434 {  435 case '1':  436 int sales;  437 cout<<"请输入修改后的销售额:";  438 cin>>sales;  439 if(cin.fail())  440 {  441 cerr<<"输入错误，请重新输入。\n";  442 }  443 else  444 {  445 ((Salesman\*)iter->pointer)->set\_sales(sales);  446 cout<<"修改成功。\n";  447 }  448 break;  449 case '2':  450 double percentage;  451 cout<<"请输入修改后的提成:";  452 cin>>percentage;  453 if(cin.fail())  454 {  455 cerr<<"输入错误，请重新输入。\n";  456 }  457 else  458 {  459 ((Salesman\*)iter->pointer)->promote(0,percentage);  460 cout<<"修改成功。\n";  461 }  462 break;  463 default:  464 cerr<<"输入错误，请重新输入。\n";  465 }  466 found=true;  467 Wait();  468 }  469 break;  470 case kTechnician:  471 if(num==((Technician\*)(iter->pointer))->individual\_emp\_no())  472 {  473 if(found)  474 {  475 cerr<<"工号重复。\n";  476 }  477 system("cls");  478 cout<<\*(Technician\*)(iter->pointer)<<endl\  479 <<"请选择欲修改的内容:\n"\  480 <<" 1.工作时长\n 2.时薪\n";  481 c=getch();  482 switch(c)  483 {  484 case '1':  485 int hours;  486 cout<<"请输入修改后的销售额:";  487 cin>>hours;  488 if(cin.fail())  489 {  490 cerr<<"输入错误，请重新输入。\n";  491 }  492 else  493 {  494 ((Technician\*)iter->pointer)->set\_working\_hours(hours);  495 cout<<"修改成功。\n";  496 }  497 break;  498 case '2':  499 double pay;  500 cout<<"请输入修改后的时薪:";  501 cin>>pay;  502 if(cin.fail())  503 {  504 cerr<<"输入错误，请重新输入。\n";  505 }  506 else  507 {  508 ((Technician\*)iter->pointer)->promote(pay,0);  509 cout<<"修改成功。\n";  510 }  511 break;  512 default:  513 cerr<<"输入错误，请重新输入。\n";  514 }  515 found=true;  516 Wait();  517 }  518 break;  519 default:  520 cerr<<"数据结构错误。\n";  521 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  522 }  523 }  524 if(!found)  525 {  526 cerr<<"对不起，没有找到匹配的工号。";  527 Wait();  528 }  529 }  530 }  531 else  532 {  533 cerr<<"对不起，输入错误；请重新输入。";  534 Wait();  535 }  536 }  537 return 0;  538 }  539 int Search()  540 {  541 for(;;)  542 {  543 cin.clear();  544 cin.sync();  545 system("cls");  546 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  547 cout<<"请选择欲使用的功能;输入其他字符可以开始输入工号:\n";  548 cout<<" D.显示当前所有员工\n";  549 cout<<" B.返回主菜单\n";  550 char choice;  551 choice=getch();  552 if(choice=='D'||choice=='d')  553 {  554 system("cls");  555 if(data.empty())  556 {  557 cerr<<"数据空。\n";  558 }  559 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  560 {  561 switch(iter->type)  562 {  563 case kEmployee:  564 cout<<\*((Employee\*)iter->pointer)<<endl;  565 break;  566 case kManager:  567 cout<<\*((Manager\*)iter->pointer)<<endl;  568 break;  569 case kSalemanager:  570 cout<<\*((Salemanager\*)iter->pointer)<<endl;  571 break;  572 case kSalesman:  573 cout<<\*((Salesman\*)iter->pointer)<<endl;  574 break;  575 case kTechnician:  576 cout<<\*((Technician\*)iter->pointer)<<endl;  577 break;  578 default:  579 cerr<<"数据结构错误。\n";  580 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  581 }  582 }  583 Pause();  584 }  585 else if(choice=='B'||choice=='b')  586 {  587 return 0;  588 }  589 else if(choice>='0'&&choice<='9')  590 {  591 int num;  592 cin>>num;  593 if(cin.fail())  594 {  595 cerr<<"对不起，输入错误；请重新输入。";  596 Wait();  597 }  598 else  599 {  600 bool found=false;  601 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  602 {  603 switch(iter->type)  604 {  605 case kEmployee:  606 if(num==((Employee\*)(iter->pointer))->individual\_emp\_no())  607 {  608 if(found)  609 {  610 cerr<<"工号重复。\n";  611 }  612 cout<<\*(Employee\*)(iter->pointer)<<endl;  613 found=true;  614 Pause();  615 }  616 break;  617 case kManager:  618 if(num==((Manager\*)(iter->pointer))->individual\_emp\_no())  619 {  620 if(found)  621 {  622 cerr<<"工号重复。\n";  623 }  624 cout<<\*(Manager\*)(iter->pointer)<<endl;  625 found=true;  626 Pause();  627 }  628 break;  629 case kSalemanager:  630 if(num==((Salemanager\*)(iter->pointer))->individual\_emp\_no())  631 {  632 if(found)  633 {  634 cerr<<"工号重复。\n";  635 }  636 cout<<\*(Salemanager\*)(iter->pointer)<<endl;  637 found=true;  638 Pause();  639 }  640 break;  641 case kSalesman:  642 if(num==((Salesman\*)(iter->pointer))->individual\_emp\_no())  643 {  644 if(found)  645 {  646 cerr<<"工号重复。\n";  647 }  648 cout<<\*(Salesman\*)(iter->pointer)<<endl;  649 found=true;  650 Pause();  651 }  652 break;  653 case kTechnician:  654 if(num==((Technician\*)(iter->pointer))->individual\_emp\_no())  655 {  656 if(found)  657 {  658 cerr<<"工号重复。\n";  659 }  660 cout<<\*(Technician\*)(iter->pointer)<<endl;  661 found=true;  662 Pause();  663 }  664 break;  665 default:  666 cerr<<"数据结构错误。\n";  667 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  668 }  669 }  670 if(!found)  671 {  672 cerr<<"对不起，没有找到匹配的工号。";  673 Wait();  674 }  675 }  676 }  677 else  678 {  679 cerr<<"对不起，输入错误；请重新输入。";  680 Wait();  681 }  682 }  683 return 0;  684 }  685  686 int Statistic()  687 {  688 int employee\_counter=0,  689 manager\_counter=0,  690 salemanager\_counter=0,  691 salesman\_counter=0,  692 technician\_counter=0;  693 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  694 {  695 switch(iter->type)  696 {  697 case kEmployee:  698 ++employee\_counter;  699 break;  700 case kManager:  701 ++manager\_counter;  702 break;  703 case kSalemanager:  704 ++salemanager\_counter;  705 break;  706 case kSalesman:  707 ++salesman\_counter;  708 break;  709 case kTechnician:  710 ++technician\_counter;  711 break;  712 default:  713 cerr<<"数据结构错误。\n";  714 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  715 return -1;  716 }  717 }  718 cin.clear();  719 cin.sync();  720 system("cls");  721 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  722 cout<<"统计结果如下:\n"  723 <<" Employee: "<<employee\_counter<<endl  724 <<" Manager: "<<manager\_counter<<endl  725 <<" Salemanager: "<<salemanager\_counter<<endl  726 <<" Salesman: "<<salesman\_counter<<endl  727 <<" Technician: "<<technician\_counter<<endl  728 <<"总计: "<<employee\_counter+  729 manager\_counter+  730 salemanager\_counter+  731 salesman\_counter+  732 technician\_counter<<endl;  733 Pause();  734 return 0;  735 }  736  737 int Delete()  738 {  739 cin.clear();  740 cin.sync();  741 system("cls");  742 cout<<" #, #= #: # #\n # ## # # #\n # ## # ### # W##: ### Y=## ##B ### #\n # # B ,R # # # # # # # I# # # # # #\n :Y# ## ##### # # # # I; # # ##### #\n ## ## # # iB ; # # I; # # #\n # ## #,#V # #RB# #:#i I+ # # #,#X .#\n";  743 cout<<"请输入欲删除的员工编号:\n";  744 int num=0;  745 cin>>num;  746 if(cin.fail())  747 {  748 cerr<<"对不起，输入错误；请重新输入。";  749 Wait();  750 return -1;  751 }  752 int found=0;  753 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  754 {  755 int emp\_no=0;  756 switch(iter->type)  757 {  758 case kEmployee:  759 emp\_no=((Employee\*)(iter->pointer))->individual\_emp\_no();  760 break;  761 case kManager:  762 emp\_no=((Manager\*)(iter->pointer))->individual\_emp\_no();  763 break;  764 case kSalemanager:  765 emp\_no=((Salemanager\*)(iter->pointer))->individual\_emp\_no();  766 break;  767 case kSalesman:  768 emp\_no=((Salesman\*)(iter->pointer))->individual\_emp\_no();  769 break;  770 case kTechnician:  771 emp\_no=((Technician\*)(iter->pointer))->individual\_emp\_no();  772 break;  773 default:  774 cerr<<"数据结构错误。\n";  775 cerr<<"类型标识:"<<hex<<(iter->type)<<endl;  776 return -1;  777 }  778 if(emp\_no==num)  779 {  780 delete (Employee\*)(iter->pointer);  781 data.erase(iter);  782 ++found;  783 }  784 if(iter==data.end())  785 {  786 break;  787 }  788 }  789 if(found>0)  790 {  791 if(found>1)  792 {  793 cerr<<"工号重复。请检查数据结构。\n";  794 Wait();  795 return -3;  796 }  797 else  798 {  799 cout<<"删除成功。\n";  800 Wait();  801 return 0;  802 }  803 }  804 else  805 {  806 if(found==0)  807 {  808 cerr<<"未找到相应的数据。\n";  809 Wait();  810 return -1;  811 }  812 }  813 cerr<<"未知错误。\n";  814 Wait();  815 return -4;  816 }  817  818 int LoadData(const char\* filename)  819 {  820 ifstream fin(filename,ios::binary|ios::app|ios::in);  821 if(fin.fail())  822 {  823 cerr<<"文件打开失败。\n";  824 Wait();  825 return -1;  826 }  827 data.clear();  828 fin.seekg(0,ios::end);  829 streampos current=fin.tellg();  830 fin.seekg(0,ios::beg);  831 if(fin.tellg()!=current)  832 {  833 fin.read((char\*)(&(Employee::counter)),sizeof(Employee::counter));  834 if(!fin.eof())  835 {  836 void\* object=NULL;  837 for(;;)  838 {  839 fin.read((char\*)(&read\_type),sizeof(read\_type));  840 switch(read\_type)  841 {  842 case kEmployee:  843 object=new Employee(0,0,false);  844 ((Employee\*)(object))->read\_from(fin);  845 break;  846 case kManager:  847 object=new Manager(false);  848 ((Manager\*)(object))->read\_from(fin);  849 break;  850 case kSalemanager:  851 object=new Salemanager(NULL,0,false);  852 ((Salemanager\*)(object))->read\_from(fin);  853 break;  854 case kSalesman:  855 object=new Salesman(0,false);  856 ((Salesman\*)(object))->read\_from(fin);  857 break;  858 case kTechnician:  859 object=new Technician(0,false);  860 ((Technician\*)(object))->read\_from(fin);  861 break;  862 default:  863 cerr<<"数据结构错误。\n";  864 Pause();  865 break;  866 }  867 object\_struct={read\_type,object};  868 if(fin.eof())  869 {  870 break;  871 }  872 if(fin.fail())  873 {  874 cerr<<"读取文件错误。\n";  875 fin.close();  876 return -2;  877 }  878 data.push\_back(object\_struct);  879 }  880 }  881 cout<<"读取文件成功。\n";  882 }  883 else  884 {  885 cout<<"创建文件成功。\n";  886 }  887 fin.close();  888 return 0;  889 }  890 int SaveData(const char\* filename)  891 {  892 ofstream fout(filename,ios::binary|ios::out);  893 if(fout.fail())  894 {  895 cerr<<"文件打开失败。\n";  896 Wait();  897 return -1;  898 }  899 fout.write((char\*)(&(Employee::counter)),sizeof(Employee::counter));  900 for(Data::iterator iter=data.begin();iter!=data.end();++iter)  901 {  902 if(fout.fail())  903 {  904 cerr<<"文件写入失败。\n";  905 Wait();  906 return -2;  907 }  908 fout.write((char\*)(&(iter->type)),sizeof(iter->type));  909 switch(iter->type)  910 {  911 case kEmployee:  912 ((Employee\*)(iter->pointer))->write\_to(fout);  913 break;  914 case kManager:  915 ((Manager\*)(iter->pointer))->write\_to(fout);  916 break;  917 case kSalemanager:  918 ((Salemanager\*)(iter->pointer))->write\_to(fout);  919 break;  920 case kSalesman:  921 ((Salesman\*)(iter->pointer))->write\_to(fout);  922 break;  923 case kTechnician:  924 ((Technician\*)(iter->pointer))->write\_to(fout);  925 break;  926 default:  927 cerr<<"数据结构错误。\n";  928 Pause();  929 break;  930 }  931 }  932 cout<<"写入文件成功。\n";  933 fout.close();  934 return 0;  935 } |

employee.h:

|  |
| --- |
| Filename犀利人事管理系统/codes/classes/employee.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef EMPLOYEE\_H\_  8 #define EMPLOYEE\_H\_  9  10 class Employee  11 {  12 public:  13 //使用等级和月薪初始化,如果add设为false则不增加计数  14 Employee(int grade=0,int pay=0,bool add=true):  15 individual\_emp\_no\_(counter),grade\_(grade),accum\_pay\_(pay)  16 {  17 if(add){++counter;}  18 ;  19 }  20 //使用虚析构函数避免使用基类指针数组时发生问题  21 //因并非每次析构都需要,遂取消了每次提示再见的功能  22 virtual ~Employee(){};  23  24 int grade(){return grade\_;}  25 void set\_grade(int grade){grade\_=grade;}  26 int individual\_emp\_no(){return individual\_emp\_no\_;}  27  28 //计算月薪  29 virtual int accum\_pay(){return accum\_pay\_;}  30 //设置底薪和提成  31 virtual void promote(int pay,double percentage){accum\_pay\_=pay;}  32 //用于二进制IO  33 virtual std::istream& read\_from(std::istream& in\_stream);  34 virtual std::ostream& write\_to (std::ostream& out\_stream);  35  36 //流IO和文件IO用到的友元  37 friend std::istream& operator>>(std::istream& in,Employee&);  38 friend std::ostream& operator<<(std::ostream& in,const Employee&);  39 friend int LoadData(const char\* filename);  40 friend int SaveData(const char\* filename);  41  42 protected:  43 int individual\_emp\_no\_;  44 int grade\_;  45 int accum\_pay\_;  46 private:  47 static int counter;  48 };  49 #endif//EMPLOYEE\_H\_ |

employee.cpp:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/employee.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<iostream>  8 #include"employee.h"  9 using namespace std;  10 int Employee::counter(2013001);  11  12 istream& Employee::read\_from(std::istream& in\_stream)  13 {  14 return in\_stream  15 .read((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  16 .read((char\*)(&grade\_),sizeof(grade\_))  17 .read((char\*)(&accum\_pay\_),sizeof(accum\_pay\_));  18 }  19  20 ostream& Employee::write\_to(std::ostream& out\_stream)  21 {  22 return out\_stream  23 .write((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  24 .write((char\*)(&grade\_),sizeof(grade\_))  25 .write((char\*)(&accum\_pay\_),sizeof(accum\_pay\_));  26 }  27  28 istream& operator>>(istream& in,Employee& employee)  29 {  30 in>>employee.grade\_>>employee.accum\_pay\_;  31 return in;  32 }  33  34 ostream& operator<<(ostream& out,const Employee& employee)  35 {  36 out<<"个人编号:"<<employee.individual\_emp\_no\_\  37 <<"\n级别:"<<employee.grade\_\  38 <<"\n月薪:"<<employee.accum\_pay\_<<endl;  39 return out;  40 } |

manager.h:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/manager.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef MANAGER\_H\_  8 #define MANAGER\_H\_  9  10 #include"employee.h"  11  12 class Manager:virtual public Employee  13 {  14 public:  15 //取消人数限制提示  16 Manager(bool add=true);  17 ~Manager(){}  18  19 //功能与基类相同,不予覆盖  20 //virtual int accum\_pay(){return accum\_pay\_;}  21 //virtual void promote(int pay,double percentage){accum\_pay\_=pay;}  22 //同名覆盖  23 virtual std::istream& read\_from(std::istream& in\_stream);  24 virtual std::ostream& write\_to (std::ostream& out\_stream);  25  26 //经理不能使用流输入,原因在于无法设置月薪,欲改变月薪只能使用promote  27 friend std::ostream& operator<<(std::ostream& out,const Manager&);  28  29 protected:  30 static const unsigned char grade\_[5];  31 };  32  33 #endif//MANAGER\_H\_ |

manager.cpp:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/manager.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<iostream>  8 #include"manager.h"  9  10 using namespace std;  11 const unsigned char Manager::grade\_[5]={0xBE,0xAD,0xC0,0xED,'\0'}; //经理inGBK  12 const int kDefaultPay(12000);  13  14 Manager::Manager(bool add):Employee(-1,kDefaultPay,add){}  15  16 istream& Manager::read\_from(std::istream& in\_stream)  17 {  18 return in\_stream  19 .read((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  20 .read((char\*)(&accum\_pay\_),sizeof(accum\_pay\_));  21 }  22  23 ostream& Manager::write\_to(std::ostream& out\_stream)  24 {  25 return out\_stream  26 .write((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  27 .write((char\*)(&accum\_pay\_),sizeof(accum\_pay\_));  28 }  29  30 ostream& operator<<(ostream& out,const Manager& manager)  31 {  32 out<<"个人编号:"<<manager.individual\_emp\_no\_\  33 <<"\n职位:"<<Manager::grade\_\  34 <<"\n月薪:"<<manager.accum\_pay\_<<endl;  35 return out;  36 } |

salemanager.h:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/salemanager.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef SALEMANAGER\_H\_  8 #define SALEMANAGER\_H\_  9  10 #include"manager.h"  11 #include"salesman.h"  12  13 class Salemanager:virtual public Manager,virtual public Salesman  14 {  15 public:  16 //取消人数限制提示  17 Salemanager(Salesman managedmen[],int n,bool add=true);  18 ~Salemanager(){}  19  20 //同名覆盖  21 virtual int accum\_pay();  22 virtual void promote(int pay,double percentage);  23 virtual std::istream& read\_from(std::istream& in\_stream);  24 virtual std::ostream& write\_to (std::ostream& out\_stream);  25  26 //销售经理不设输入流  27 friend std::ostream& operator<<(std::ostream& in,const Salemanager&);  28  29 protected:  30 static const unsigned char grade\_[9];  31 int kPay;  32 double kPercentage;  33 };  34 #endif//SALEMANAGER\_H\_ |

salemanager.cpp:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/salemanager.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<iostream>  8 #include<vector>  9 #include"salemanager.h"  10  11 using namespace std;  12  13 const unsigned char Salemanager::grade\_[9]=  14 {0xCF,0xFA,0xCA,0xDB,0xBE,0xAD,0xC0,0xED,'\0'};  15 const int kDefaultPay(8000);  16 const double kDefaultPercentage(0.04);  17  18 Salemanager::Salemanager(Salesman managedmen[],int n,bool add):  19 Employee(-1,0,add),Manager(false),  20 kPay(kDefaultPay),kPercentage(kDefaultPercentage)  21 {  22 if(!add)  23 {  24 return;  25 }  26 sales\_=0;  27 for(int i=0;i<n;++i)  28 {  29 sales\_+=managedmen[i].sales();  30 }  31 accum\_pay\_=accum\_pay();  32 }  33  34 int Salemanager::accum\_pay()  35 {  36 return kPay+kPercentage\*sales\_;  37 }  38  39 void Salemanager::promote(int pay,double percentage)  40 {  41 kPay=(pay==0?kPay:pay);  42 kPercentage=(percentage==0?kPercentage:percentage);  43 accum\_pay\_=accum\_pay();  44 }  45  46 istream& Salemanager::read\_from(std::istream& in\_stream)  47 {  48 return in\_stream  49 .read((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  50 .read((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  51 .read((char\*)(&kPay),sizeof(kPay))  52 .read((char\*)(&kPercentage),sizeof(kPercentage));  53 }  54  55 ostream& Salemanager::write\_to(std::ostream& out\_stream)  56 {  57 return out\_stream  58 .write((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  59 .write((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  60 .write((char\*)(&kPay),sizeof(kPay))  61 .write((char\*)(&kPercentage),sizeof(kPercentage));  62 }  63  64 ostream& operator<<(std::ostream& out,const Salemanager& salemanager)  65 {  66 out<<"个人编号:"<<salemanager.individual\_emp\_no\_\  67 <<"\n职位:"<<Salemanager::grade\_\  68 <<"\n月薪:"<<salemanager.accum\_pay\_<<endl;  69 return out;  70 } |

salesman.h:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/salesman.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef SALESMAN\_H\_  8 #define SALESMAN\_H\_  9  10 #include"employee.h"  11  12 class Salesman:virtual public Employee  13 {  14 public:  15 Salesman(int sales=0,bool add=true);  16 ~Salesman(){}  17  18 virtual int accum\_pay();  19 virtual void promote(int pay,double percentage);  20 virtual std::istream& read\_from(std::istream& in\_stream);  21 virtual std::ostream& write\_to (std::ostream& out\_stream);  22  23 int sales(){return sales\_;}  24 void set\_sales(int sales)  25 {  26 sales\_=sales;  27 accum\_pay\_=accum\_pay();  28 }  29  30 friend std::istream& operator>>(std::istream& in,Salesman&);  31 friend std::ostream& operator<<(std::ostream& in,const Salesman&);  32  33 protected:  34 static const unsigned char grade\_[7];  35 int sales\_;  36 double kPercentage;  37 };  38  39 #endif//SALESMAN\_H\_ |

salesman.cpp:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/salesman.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<iostream>  8 #include"salesman.h"  9 using namespace std;  10 const unsigned char Salesman::grade\_[7]={0xCF,0xFA,0xCA,0xDB,0xD4,0xB1,'\0'};  11 const double kDefaultPercentage(0.05);  12  13 Salesman::Salesman(int sales,bool add):  14 Employee(-1,kDefaultPercentage\*sales,add),  15 sales\_(sales),kPercentage(kDefaultPercentage){}  16  17 int Salesman::accum\_pay()  18 {  19 return kPercentage\*sales\_;  20 }  21 void Salesman::promote(int pay,double percentage)  22 {  23 kPercentage=percentage;  24 accum\_pay\_=accum\_pay();  25 }  26 istream& Salesman::read\_from(std::istream& in\_stream)  27 {  28 return in\_stream  29 .read((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  30 .read((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  31 .read((char\*)(&sales\_),sizeof(sales\_))  32 .read((char\*)(&kPercentage),sizeof(kPercentage));  33 }  34 ostream& Salesman::write\_to(std::ostream& out\_stream)  35 {  36 return out\_stream  37 .write((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  38 .write((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  39 .write((char\*)(&sales\_),sizeof(sales\_))  40 .write((char\*)(&kPercentage),sizeof(kPercentage));  41 }  42  43 istream& operator>>(istream& in,Salesman& salesman)  44 {  45 in>>salesman.sales\_;  46 if(!in.fail())  47 {  48 salesman.accum\_pay\_=salesman.accum\_pay();  49 }  50 return in;  51 }  52  53 ostream& operator<<(ostream& out,const Salesman& salesman)  54 {  55 out<<"个人编号:"<<salesman.individual\_emp\_no\_\  56 <<"\n职位:"<<Salesman::grade\_\  57 <<"\n月销售额:"<<salesman.sales\_\  58 <<"\n月薪:"<<salesman.accum\_pay\_<<endl;  59 return out;  60 } |

technician.h:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/technician.h  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #ifndef TECHNICIAN\_H\_  8 #define TECHNICIAN\_H\_  9  10 #include"employee.h"  11  12 class Technician:virtual public Employee  13 {  14 public:  15 //去掉人数限制提示  16 Technician(int working\_hours=0,bool add=true);  17 ~Technician(){}  18  19 virtual int accum\_pay();  20 virtual void promote(int pay,double percentage);  21 virtual std::istream& read\_from(std::istream& in\_stream);  22 virtual std::ostream& write\_to (std::ostream& out\_stream);  23  24 int working\_hours(){return working\_hours\_;}  25 void set\_working\_hours(int working\_hours)  26 {  27 working\_hours\_=working\_hours;  28 accum\_pay\_=kHourPay\*working\_hours;  29 }  30  31 friend std::istream& operator>>(std::istream& in,Technician&);  32 friend std::ostream& operator<<(std::ostream& in,const Technician&);  33  34 protected:  35 static const unsigned char grade\_[7];  36 int working\_hours\_;  37 int kHourPay;  38 };  39  40 #endif//TECHNICIAN\_H\_ |

technician.cpp:

|  |
| --- |
| Filename 犀利人事管理系统/codes/classes/technician.cpp  1 /\*  2 \* File Encoding: CP936/GBK  3 \* Chi Blaok @ EE THU, All Rights Reserved.  4 \*  5 \*/  6  7 #include<iostream>  8 #include"technician.h"  9 using namespace std;  10 const unsigned char Technician::grade\_[7]={0xBC,0xBC,0xCA,0xF5,0xD4,0xB1,'\0'};  11 const int kDefaultHourPay(260);  12  13 Technician::Technician(int working\_hours,bool add):Employee(-1,kDefaultHourPay\*working\_hours,add),working\_hours\_(working\_hours),kHourPay(kDefaultHourPay){}  14  15 int Technician::accum\_pay()  16 {  17 return kHourPay\*working\_hours\_;  18 }  19 void Technician::promote(int pay,double percentage)  20 {  21 kHourPay=pay;  22 accum\_pay\_=accum\_pay();  23 }  24 istream& Technician::read\_from(std::istream& in\_stream)  25 {  26 return in\_stream  27 .read((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  28 .read((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  29 .read((char\*)(&working\_hours\_),sizeof(working\_hours\_))  30 .read((char\*)(&kHourPay),sizeof(kHourPay));  31 }  32 ostream& Technician::write\_to(std::ostream& out\_stream)  33 {  34 return out\_stream  35 .write((char\*)(&individual\_emp\_no\_),sizeof(individual\_emp\_no\_))  36  37 .write((char\*)(&accum\_pay\_),sizeof(accum\_pay\_))  38 .write((char\*)(&working\_hours\_),sizeof(working\_hours\_))  39 .write((char\*)(&kHourPay),sizeof(kHourPay));  40 }  41  42 istream& operator>>(istream& in,Technician& technician)  43 {  44 in>>technician.working\_hours\_;  45 if(!in.fail())  46 {  47 technician.accum\_pay\_=technician.accum\_pay();  48 }  49 return in;  50 }  51  52 ostream& operator<<(ostream& out,const Technician& technician)  53 {  54 out<<"个人编号:"<<technician.individual\_emp\_no\_\  55 <<"\n职位:"<<Technician::grade\_\  56 <<"\n月工作小时数:"<<technician.working\_hours\_\  57 <<"\n月薪:"<<technician.accum\_pay\_<<endl;  58 return out;  59 } |

## 附录2：评分表

课程名称： 面向对象程序设计

|  |  |  |
| --- | --- | --- |
| **项 目** | **评 价** | |
| 设计方案的合理性与创新性 | **3×2** |  |
| 设计与调试结果 | **4×2** |  |
| 设计说明书的质量 | **1×2** |  |
| 程序基本要求涵盖情况 | **4×2** |  |
| 程序代码编写素养情况 | **2×2** |  |
| 课程设计周表现情况 | **1×2** |  |
| 综合成绩 | **15×2** |  |

教师签名：\_\_\_\_\_\_\_\_\_\_\_

日 期：\_\_\_\_\_\_\_\_\_\_\_