西北大学

面向对象程序设计

上机实验报告

学号：2020115025

班级：计科2班

姓名：薄劲阳

日期：2022年6月5日

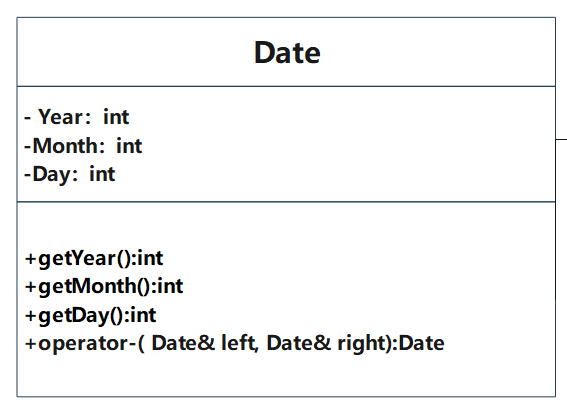
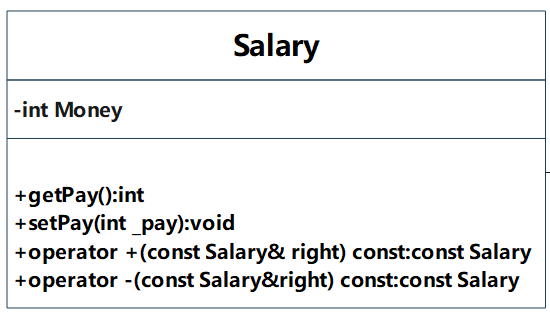
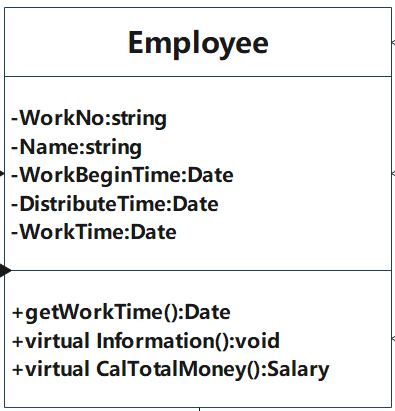
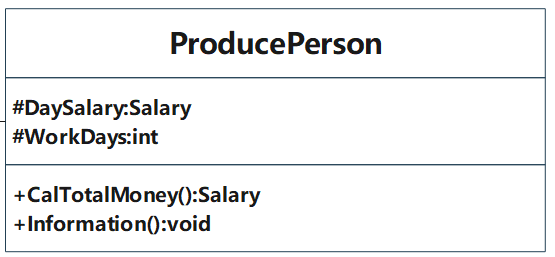
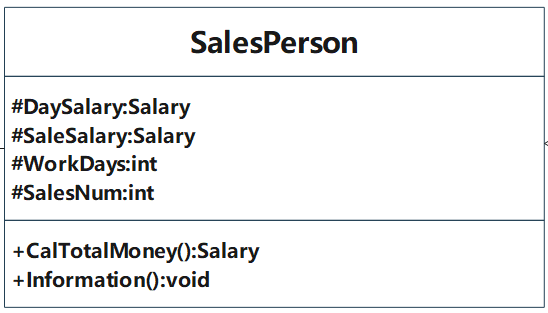
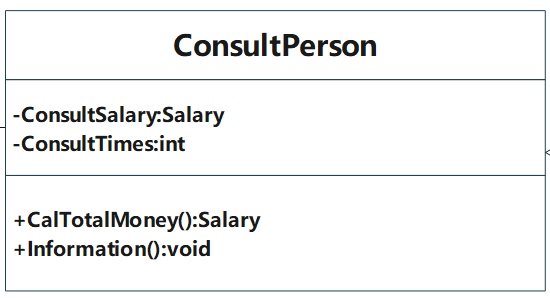
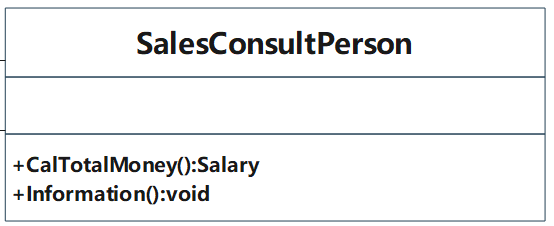
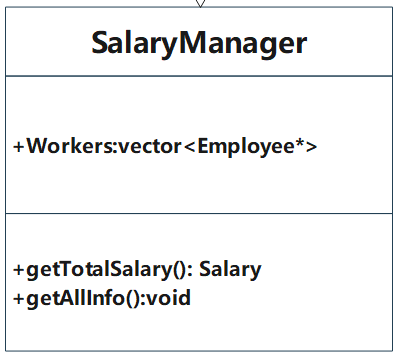
1. 问题分析

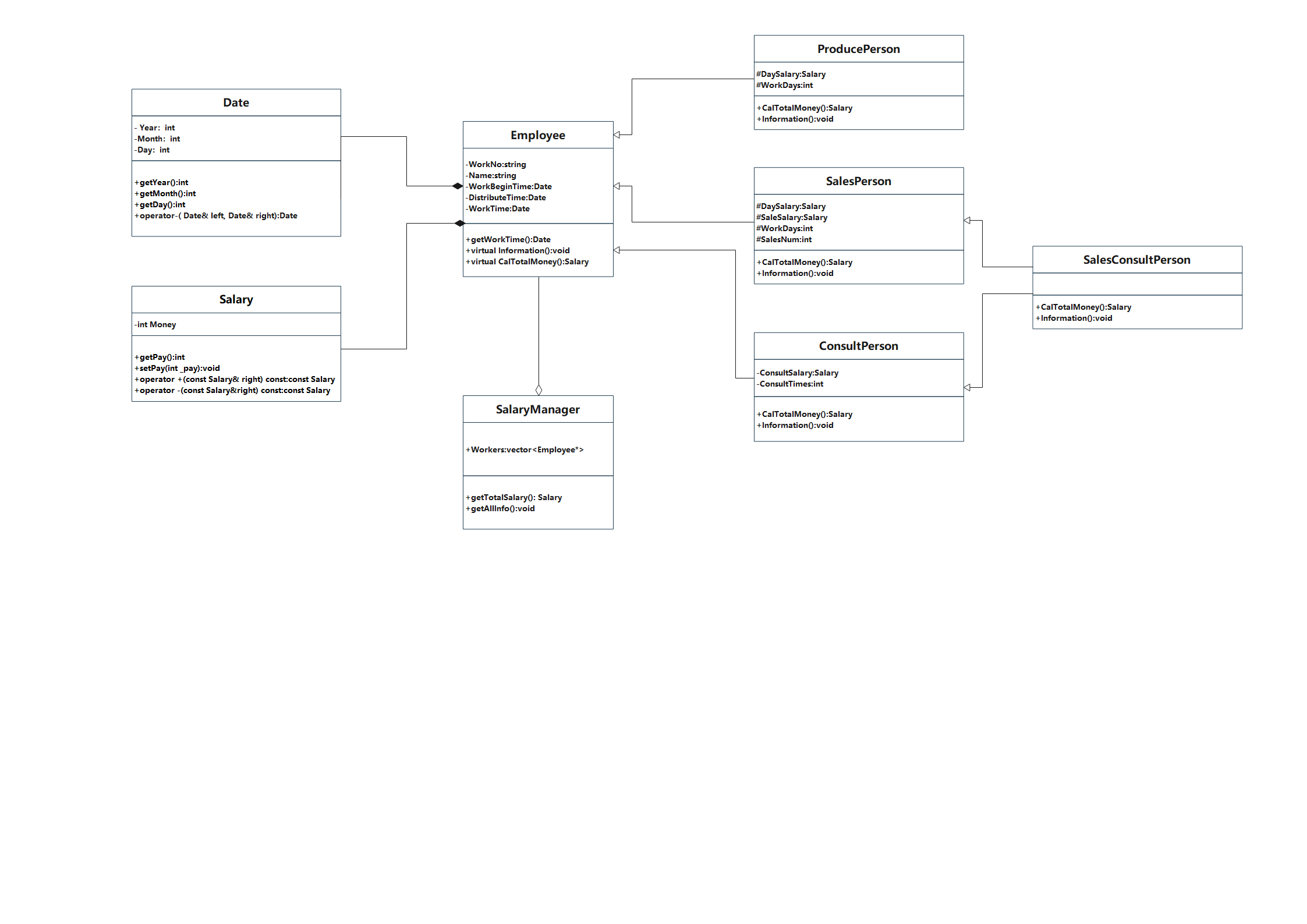
* 问题实例化

工资管理系统中，工资的分发与工作时间，薪资标准、工种有关，因此，可以在此基础上构建类。

* + 日期
  + 薪资
  + 员工（产生五种子类）
    - 流水线员工
    - 销售员工
    - 咨询员工
    - 销售咨询员工（销售员工与咨询员工的子类）
    - 工资管理员

1. 问题建模和流程设计

* UML建模
  + Date类
    - 
  + Salary类
    - 
  + Employee类
    - 
    - ProducePerson类
      * 
    - SalesPerson类
      * 
    - ConsultPerson类
      * 
    - SalesConsultPerson类
      * 
    - SalaryManager类
      * 
  + 类与类之间的关系



三、程序实现和测试

四、实验源码

* **h文件**
  + **Basic.h（Salary、Date类）**

1. //基础类的声明
2. #ifndef BASIC\_H\_INCLUDE
3. #define BASIC\_H\_INCLUDE
4. #include<iostream>
5. #include<string>
7. **using** **namespace** std;
9. //Date类
10. **class** Date {
11. **int** Year;
12. **int** Month;
13. **int** Day;
14. **public**:
15. Date();//默认构造函数
16. Date(**int** \_year,**int** \_month,**int** \_day);//隐式类型转换构造函数
17. Date(**const** Date& a);//拷贝构造函数
18. //访问器和修改器
19. **int** getYear();
20. **int** getMonth();
21. **int** getDay();
22. **void** setYear(**int** \_year);
23. **void** setMonth(**int** \_month);
24. **void** setDay(**int** \_day);
26. **friend** **const**  Date operator-( Date& left, Date& right);//-双目运算符重载——非成员函数
28. };

31. **class** Salary {
32. **int** Pay;
33. **public**:
34. Salary();//默认构造函数
35. Salary(**int** Pay);//隐式类型转换构造函数
36. Salary(**const** Salary &a);//拷贝构造函数
37. **int** getPay();
38. **void** setPay(**int** Pay);
39. //运算符重载
40. **const** Salary operator +(**const** Salary& right) **const**;
41. **const** Salary operator -(**const** Salary&right) **const**;
42. };

45. #endif
    * **Employee.h（Employee类）**
46. //工种超类的声明
47. #ifndef PEOPLE\_H\_INCLUDE
48. #define PEOPLE\_H\_INCLUDE
49. #include<iostream>
50. #include<string>
51. #include"Basic.h"
53. **using** **namespace** std;
55. **class** Employee {
56. **private**:
57. string WorkNo;//工号
58. string Name;
59. Date WorkBeginTime;//开始工作时间
60. Date DistributeTime;//结发工资时间
61. Date WorkTime;//实际工作时长
62. **public**:
63. //成员函数
64. //构造函数
65. Employee();//默认构造函数
66. Employee(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime); //隐式类型转换构造函数
67. Employee(**const** Employee&a);//拷贝构造函数
68. //访问器和修改器
69. string getWorkNo();
70. **void** setWorkNo(string WorkNo);
71. string getName();
72. **void** setName(string Name);
73. Date getWorkBeginTime();
74. **void** setWorkBeginTime(Date WorkBeginTime);
75. Date getDistributeTime();
76. **void** setDistributeTime(Date DistributeTime);
77. Date getWorkTime();

80. //虚函数——希望子类覆盖的函数
81. **virtual** **void** Information();//显示信息——const成员函数，对对象不修改
82. **virtual** Salary CalTotalMoney();//计算总工资
83. };

86. #endif
    * **Employees.h（ProducePerson、SalesPerson、ConsultPerson、SalesConsultPerson、SalaryManager类）**
87. //子类的声明
88. #ifndef EMPLOYS\_H\_INCLUDE
89. #define EMPLOYS\_H\_INCLUDE
90. #include"Employee.h"
91. #include<vector>
93. **using** **namespace** std;
95. //生产线员工
96. **class** ProducePerson : **public** Employee {
97. **protected**:
98. Salary DaySalary = 500;//流水线员工每天月薪500元——调用隐式类型转换构造函数
99. **int** WorkDays;//工作总日期
100. **public**:
101. //构造函数
102. ProducePerson();//默认构造函数
103. ProducePerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime);
104. //覆盖父类的函数（虚函数）
105. Salary CalTotalMoney() override;
106. **void** Information() override;
107. };
108. //销售员工
109. **class** SalesPerson : **virtual** **public** Employee {
110. **protected**:
111. Salary DaySalary = 400;//销售员工日薪为400元
112. Salary SaleSalary = 50;//每销售一件提成50元
113. **int** WorkDays;//工作总日期
114. **int** SalesNum;//销售产品件数
115. **public**:
116. //构造函数
117. SalesPerson();
118. SalesPerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime, **int** SalesNum);
119. //虚函数-覆盖父类的函数
120. Salary CalTotalMoney() override;//薪资计算-覆盖基类的函数
121. **void** Information() override;//信息显示
122. };
123. //顾问员工-虚基类继承Employee 因为后面的 SalesConsultPerson要继承，如果不设置为虚基类，会导致二义性
124. **class** ConsultPerson : **virtual** **public** Employee {
125. **protected**:
126. Salary ConsultSalary = 300;//每次咨询费300元
127. **int** ConsultTimes;//咨询次数
128. **public**:
129. ConsultPerson();
130. ConsultPerson(string WorkNo, string Name, **int** ConsultTimes);
131. //覆盖父类的函数
132. Salary CalTotalMoney() override;
133. **void** Information() override;
134. };
136. //销售顾问
137. **class** SalesConsultPerson :**public** SalesPerson,**public** ConsultPerson {
138. **public**:
139. SalesConsultPerson();
140. SalesConsultPerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime, **int** SalesNum,**int** ConsultTimes);
141. Salary CalTotalMoney() override;
142. **void** Information() override;
143. };

146. //工资管理员
147. **class** SalaryManager {
148. **public**:
149. //成员变量
150. vector<Employee\*> Workers;//存储所有员工——动态数组（向上类型转换）
151. //成员函数
152. SalaryManager(){};
153. Salary getTotalSalary();//获取所有员工的总工资
154. **void** getAllInfo();//所有信息
156. };

159. #endif

* **cpp文件**
  + **main.cpp**

1. #include <iostream>
2. #include<string>
3. #include"Basic.h"
4. #include"Employee.h"
5. #include"Employees.h"
7. **using** **namespace** std;
9. **int** main(**int** argc, **char**\*\* argv) {
10. **const** Date timeBegin(2021, 2, 10); //上次发工资时间
11. **const** Date timeEnd1(2021, 4, 10); //本次发工资时间
12. //生产线员工
13. ProducePerson Worker1("001", "张三", timeBegin, timeEnd1);
14. //销售员工
15. SalesPerson Worker2("002", "李四", timeBegin, timeEnd1, 50);
16. //顾问类员工
17. ConsultPerson Worker3("003", "王五", 40);
18. //销售顾问类员工
19. SalesConsultPerson Worker4("004","赵六",timeBegin, timeEnd1, 50, 40);
20. //工资管理员
21. SalaryManager Boss;
22. Boss.Workers = {&Worker1,&Worker2,&Worker3,&Worker4};//数组赋值
23. Boss.getAllInfo();
24. cout<<"所有员工总工资为:"<<Boss.getTotalSalary().getPay()<<endl;
25. system("pause");
27. **return** 0;
28. }
    * **Basic.cpp**
29. #include"Basic.h"
31. **using** **namespace** std;
33. //Date类
34. //默认构造函数
35. Date::Date() {
36. Year = 0;
37. Month = 0;
38. Day = 0;
39. }
40. //隐式类型转换构造函数
41. Date::Date(**int** \_year, **int** \_month, **int** \_day): Year(\_year), Month(\_month), Day(\_day) {
42. }
43. //拷贝构造函数-浅拷贝
44. Date::Date(**const** Date& a): Year(a.Year), Month(a.Month), Day(a.Day) { //初始化列表初始化
45. }
47. //访问器
48. **int** Date::getYear() {
49. **return** Year;
50. }
51. **int** Date::getMonth() {
52. **return** Month;
53. }
54. **int** Date::getDay() {
55. **return** Day;
56. }
57. //修改器
58. **void** Date::setYear(**int** Year) {
59. **if** (Year <= 0 || Year > 2022) {
60. cout << "年份输入有误，自动进行默认初始化" << endl;
61. **return**;
62. } **else** {
63. **this**->Year = Year;
64. cout << "年份修改成功" << endl;
65. **return**;
66. }
68. }
69. **void** Date::setMonth(**int** Month) {
70. **if** (Month >= 1 && Month <= 12) {
71. **this**->Month = Month;
72. cout << "月份修改成功" << endl;
73. **return**;
74. } **else** {
75. cout << "月份输入有误，自动进行默认初始化" << endl;
76. **return**;
77. }
78. }
79. **void** Date::setDay(**int** Day) {
80. **if** (Day >= 1 && Day <= 30) { //这里默认每个月均为30天
81. **this**->Day = Day;
82. cout << "日期修改成功" << endl;
83. **return**;
84. } **else** {
85. cout << "日期输入有误，自动进行默认初始化" << endl;
86. **return**;
87. }
88. }
89. //计算差值——运算符重载(二元运算符) 返回值为Date类 Date类中的成员函数
90. **const** Date operator-( Date& left, Date& right) {
91. **int** YearGap = left.getYear() - right.getYear();
92. **int** MonthGap, DayGap;
93. **if** (YearGap == 0) { //如果在同一年，月份直接作差就好
94. MonthGap = left.getMonth() - right.getMonth();
95. } **else** { //如果不在同一年，需向借一年份，同时减少一年份
96. //如果最后结束时间的月数比开始工作的晚，直接作差
97. **if** (left.getMonth() >= right.getMonth()) {
98. MonthGap = left.getMonth() - right.getMonth(); //大减小
99. } **else** {
100. MonthGap = left.getMonth() + 12 - right.getMonth(); //从年份借一年
101. YearGap -= 1; //年份差减一年
102. }
103. }
104. //工资计算以月份为单位,只有日期差为负值的时候才会影响月份差
105. DayGap = left.getDay() - right.getDay();
106. **if** (DayGap < 0) {
107. MonthGap -= 1; //MonthGap减1 因为没有达到原来的MonthGap
108. DayGap = right.getDay() - left.getDay();
109. }
111. **return** Date(YearGap, MonthGap, DayGap);
112. }
114. //Salary类
115. //默认构造函数——无参
116. Salary::Salary() {
117. Pay = 0;//默认为0元
118. }
119. //隐式类型转换构造函数
120. Salary::Salary(**int** Pay) {
121. **this**->Pay = Pay;//用this来区分是成员变量
122. }
123. //拷贝构造函数(用Salary类进行初始化)
124. Salary::Salary(**const** Salary &a) {
125. Pay = a.Pay;
126. }
128. //访问器
129. **int** Salary:: getPay() {
130. **return** Pay;
131. }
132. //修改器
133. **void** Salary:: setPay(**int** Pay) {
134. **if** (Pay <= 200) {
135. cout << "每月工资金额不得少于两百元，已自动默认为2000元每月" << endl;
136. **return**;
137. }
138. cout << "每月工资修改成功" << endl;
139. **this**->Pay = Pay;
140. **return**;
141. }
142. //运算符重载(二元运算符、成员函数)
143. //+运算符重载——返回值为const Salary类型
144. **const** Salary Salary::operator +(**const** Salary& right) **const** {
145. **return** Salary(Pay + right.Pay);
146. }
147. //-运算符重载
148. **const** Salary Salary::operator -(**const** Salary&right) **const** {
149. **return** Salary(Pay - right.Pay);
150. }
     * **Employee.cpp**
151. //超类成员函数定义
152. #include"Employee.h"
154. **using** **namespace** std;

157. //默认构造函数(采用初始化列表初始化)——期中类的直接初始化会调用相应的构造函数
158. Employee::Employee(): WorkNo("NULL"), Name("John Doe"), WorkBeginTime(), DistributeTime() {
159. }
160. //隐式类型转换构造函数
161. Employee::Employee(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime) {
162. **this**->WorkNo = WorkNo;
163. **this**->Name = Name;
164. **this**->WorkBeginTime = WorkBeginTime; //调用Date类的拷贝构造函数
165. **this**->DistributeTime = DistributeTime;
166. //实际工作时长等于工资发放时间减开始工作时间
167. **this**->WorkTime = DistributeTime - WorkBeginTime; //调用Date类的-号运算符重载函数
168. //  this->TotalMoney = CalTotalMoney(); //总工资由函数计算得到，调用Salary类的拷贝构造函数
169. }
170. //拷贝构造函数
171. Employee::Employee(**const** Employee&a) {
172. WorkNo = a.WorkNo;
173. Name = a.Name;
174. WorkBeginTime = a.WorkBeginTime;
175. DistributeTime = a.DistributeTime;
176. WorkTime = a.WorkTime;
177. }
179. //访问器
180. string Employee::getWorkNo()  {
181. **return** WorkNo;
182. }
184. string Employee::getName()  {
185. **return** Name;
186. }
188. Date Employee::getWorkBeginTime() {
189. **return** WorkBeginTime;
190. }
191. Date Employee::getDistributeTime() {
192. **return** DistributeTime;
193. }
194. Date Employee::getWorkTime() { //获取实际工作时长
195. **return** WorkTime;
196. }

199. **void** Employee::setWorkNo(string WorkNo) {
200. **this**->WorkNo = WorkNo;
201. cout << "工号修改成功" << endl;
202. **return**;
203. }
204. **void** Employee::setName(string Name) {
205. **this**->Name = Name;
206. cout << "姓名修改成功" << endl;
207. **return**;
208. }
210. **void** Employee::setWorkBeginTime(Date WorkBeginTime) {
211. **this**->WorkBeginTime = WorkBeginTime;
212. cout << "开始工作时间修改成功" << endl;
213. **return**;
214. }
215. **void** Employee::setDistributeTime(Date DistributeTime) {
216. **this**->DistributeTime = DistributeTime;
217. cout << "工作发放时间修改成功" << endl;
218. **return**;
219. }
221. //父类中没有实际用处，需要在子类中重写覆盖
223. //员工信息显示(虚函数-不同子类打印的信息不同)
224. **void** Employee::Information()  {
225. **return**;
226. }
228. //计算所得工资(虚函数-不同子类打印的信息不同)
229. Salary Employee:: CalTotalMoney() {
230. **return** 0;
231. }
     * **Employees.cpp**
232. //子类的声明
233. #ifndef EMPLOYS\_H\_INCLUDE
234. #define EMPLOYS\_H\_INCLUDE
235. #include"Employee.h"
236. #include<vector>
238. **using** **namespace** std;
240. //生产线员工
241. **class** ProducePerson : **public** Employee {
242. **protected**:
243. Salary DaySalary = 500;//流水线员工每天月薪500元——调用隐式类型转换构造函数
244. **int** WorkDays;//工作总日期
245. **public**:
246. //构造函数
247. ProducePerson();//默认构造函数
248. ProducePerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime);
249. //覆盖父类的函数（虚函数）
250. Salary CalTotalMoney() override;
251. **void** Information() override;
252. };
253. //销售员工
254. **class** SalesPerson : **virtual** **public** Employee {
255. **protected**:
256. Salary DaySalary = 400;//销售员工日薪为400元
257. Salary SaleSalary = 50;//每销售一件提成50元
258. **int** WorkDays;//工作总日期
259. **int** SalesNum;//销售产品件数
260. **public**:
261. //构造函数
262. SalesPerson();
263. SalesPerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime, **int** SalesNum);
264. //虚函数-覆盖父类的函数
265. Salary CalTotalMoney() override;//薪资计算-覆盖基类的函数
266. **void** Information() override;//信息显示
267. };
268. //顾问员工-虚基类继承Employee 因为后面的 SalesConsultPerson要继承，如果不设置为虚基类，会导致二义性
269. **class** ConsultPerson : **virtual** **public** Employee {
270. **protected**:
271. Salary ConsultSalary = 300;//每次咨询费300元
272. **int** ConsultTimes;//咨询次数
273. **public**:
274. ConsultPerson();
275. ConsultPerson(string WorkNo, string Name, **int** ConsultTimes);
276. //覆盖父类的函数
277. Salary CalTotalMoney() override;
278. **void** Information() override;
279. };
281. //销售顾问
282. **class** SalesConsultPerson :**public** SalesPerson,**public** ConsultPerson {
283. **public**:
284. SalesConsultPerson();
285. SalesConsultPerson(string WorkNo, string Name, Date WorkBeginTime, Date DistributeTime, **int** SalesNum,**int** ConsultTimes);
286. Salary CalTotalMoney() override;
287. **void** Information() override;
288. };

291. //工资管理员
292. **class** SalaryManager {
293. **public**:
294. //成员变量
295. vector<Employee\*> Workers;//存储所有员工——动态数组（向上类型转换）
296. //成员函数
297. SalaryManager(){};
298. Salary getTotalSalary();//获取所有员工的总工资
299. **void** getAllInfo();//所有信息
301. };

304. #endif