知识点列表

编号	名称	描述	级别
1	项目开发流程	从需求到设计再到编码和测试的瀑布式项目开发流程。	*
2	三层体系架构	由用户界面层、业务逻辑层和数据访问层组成的三层体系架构。	*
3	三层逻辑模型	由接口层、实现层和逻辑对象层组成的三层逻辑模型。	*
4	接口设计实现	基于继承与多态的抽象接口设计与实现。	*
5	用户界面设计	面向控制台应用的字符界面设计。	**
6	业务逻辑设计	连接界面与数据的业务逻辑流程。	**
7	数据存储设计	基于文件系统的数据存储与访问。	**
8	多源文件构建	基于makefile的多源文件系统构建技术。	**

注： “*”理解级别 “**”掌握级别 “***”应用级别

目录

1. 概述

本项目旨在通过一个简化的企业管理信息系统（Enterprise Management Information System，EMIS）项目，使学生在完成对 C/C++程序设计语言和基本数据结构与算法课程的学习后，综合运用所学到的语法和算法知识，构建一个接近实际应用场景的软件系统，以达到复习和巩固前期课程内容并为后续课程奠定基础的目的。

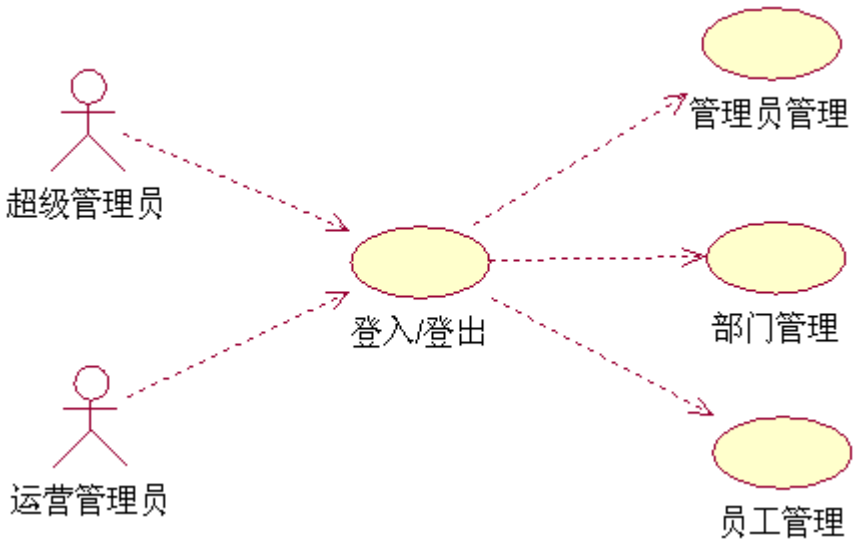
通过本项目的实施，学生可以初步了解包括需求分析、概要设计、详细设计、开发计划、编码测试等环节在内的软件项目开发流程，以及相关技术文档的撰写规范，为以后从事软件项目研发工作增加实践经验。

本案在系统设计方面有意识地采用多层体系架构的设计理念，旨在帮助学生逐步树立产品观念，从更高的角度，以更广的视野，综合考虑用户需求、技术路线和研发成本间的矛盾，深刻理解软件系统的可维护性、可扩展性对企业可持续性发展的重要意义。

2. 需求分析

2.1. 总体需求

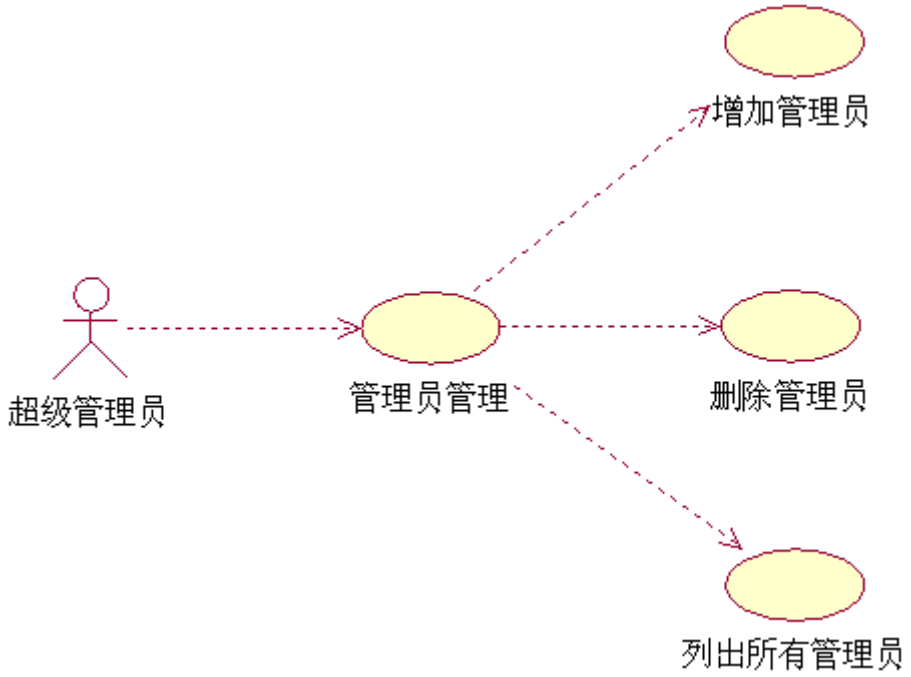
企业管理信息系统主要用于实现对企业基本信息的管理。具体包括对企业部门的管理、对企业员工的管理，以及对管理信息系统本身的管理。如图所示：



其中，对管理信息系统本身的管理主要是指对管理员的管理，这方面的需求可被归纳为管理需求，而对企业部门和员工的管理则被归纳为业务管理。

2.2. 管理需求

管理需求主要包括：增加管理员、删除管理员，以及列出所有管理员三项功能。如图所示：



用户进入系统，可见如下主菜单：

```
企业管理信息系统
------------------
[1] 增加管理员
[2] 删除管理员
[3] 列出所有管理员
[4] 运营管理
[0] 退出
------------------
请选择：
```

2.2.1. 增加管理员

用户从主菜单选择[1]，进入增加管理员功能模块，根据屏幕提示依次输入管理员的用户名和密码。系统自动为其分配ID号，并向用户提供反馈信息。如图所示：

```
请选择：1

请输入管理员用户名：李建华
请输入管理员密码：123456
增加管理员成功!

企业管理信息系统
------------------
[1] 增加管理员
[2] 删除管理员
[3] 列出所有管理员
[4] 运营管理
[0] 退出
------------------
请选择：
```

之后返回主菜单，等待用户后续操作。

2.2.2. 删除管理员

用户从主菜单选择[2]，进入删除管理员功能模块，根据屏幕提示输入欲删除管理员的ID号。系统将该管理员删除，并向用户提供反馈信息。如图所示：

```
请选择：2

请输入管理员ID：1003
删除管理员成功!

企业管理信息系统
------------------
[1] 增加管理员
[2] 删除管理员
[3] 列出所有管理员
[4] 运营管理
[0] 退出
------------------
请选择：
```

之后返回主菜单，等待用户后续操作。

2.2.3. 列出所有管理员

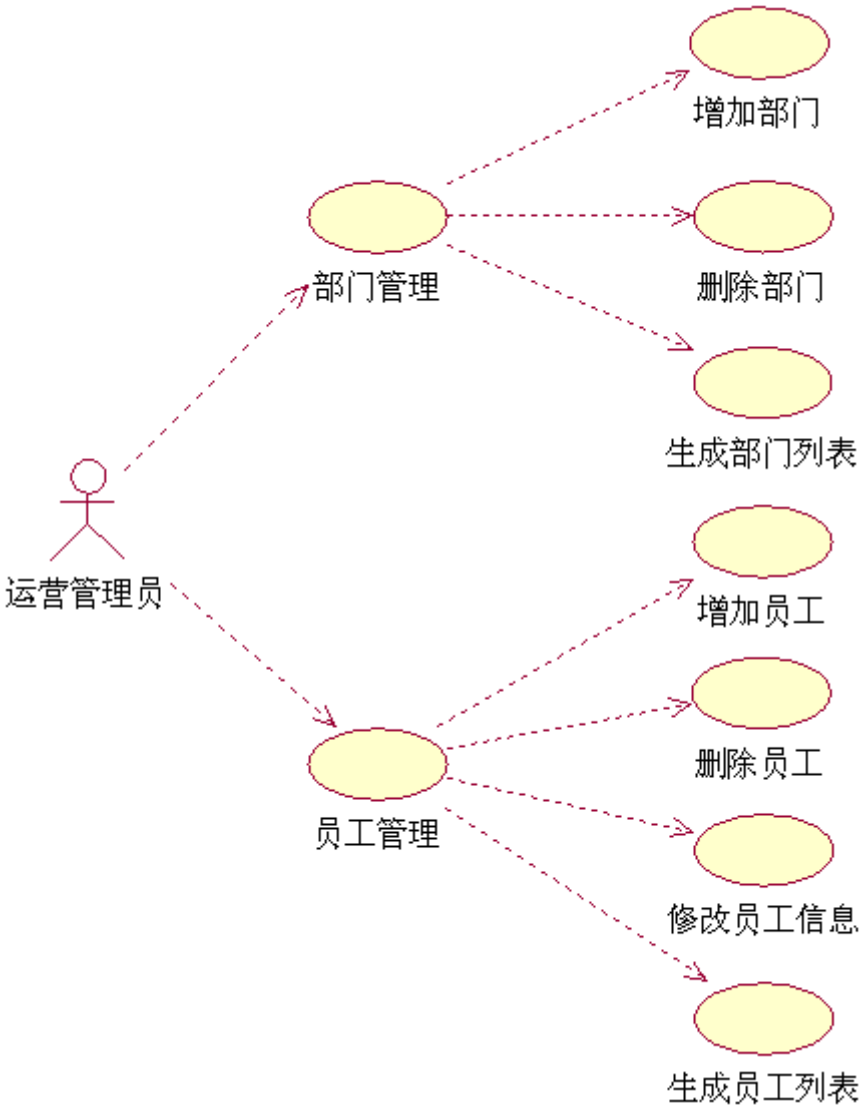
用户从主菜单选择[3]，进入列出所有管理员功能模块。系统以列表形式显示所有管理员的ID号、用户名和密码。如图所示：

```
请选择：3

管理员ID    管理员用户名          管理员密码
-----------------------------------------
1001        李建华                123456
1002        刘海涛                888888
1003        马文忠                000000

企业管理信息系统
------------------
[1] 增加管理员
[2] 删除管理员
[3] 列出所有管理员
[4] 运营管理
[0] 退出
------------------
请选择：
```

之后返回主菜单，等待用户后续操作。

2.3. 业务需求

业务需求主要包括：部门管理和员工管理两部分功能。其中部门管理包括：增加部门、删除部门和生成部门列表等三项功能。而员工管理则包括：增加员工、删除员工、修改员工信息，以及生成员工列表等四项功能。针对生成员工列表，又可进一步细化为生成全体员工列表和生成某一特定部门的员工列表两种情况。如图所示：



用户从主菜单选择[4]，可见如下运营管理子菜单：

```
请选择：4

运营管理子系统
--------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
--------------------
请选择：
```

2.3.1. 部门管理

2.3.1.1. 增加部门

用户从运营管理子菜单选择[1]，进入增加部门功能模块，根据屏幕提示输入部门名称。系统自动为其分配ID号，并向用户提供反馈信息。如图所示：

```
请选择：1

请输入部门名称：研发部
增加部门成功!

运营管理子系统
--------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
--------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.1.2. 删除部门

用户从运营管理子菜单选择[2]，进入删除部门功能模块，根据屏幕提示输入欲删除部门的ID号。系统将该部门删除，并向用户提供反馈信息。如图所示：

```
请选择：2

请输入部门ID：1007
删除部门成功!

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.1.3. 列出部门

用户从运营管理子菜单选择[3]，进入列出部门功能模块。系统以列表形式显示所有部门的ID号、部门名称和员工人数。如图所示：

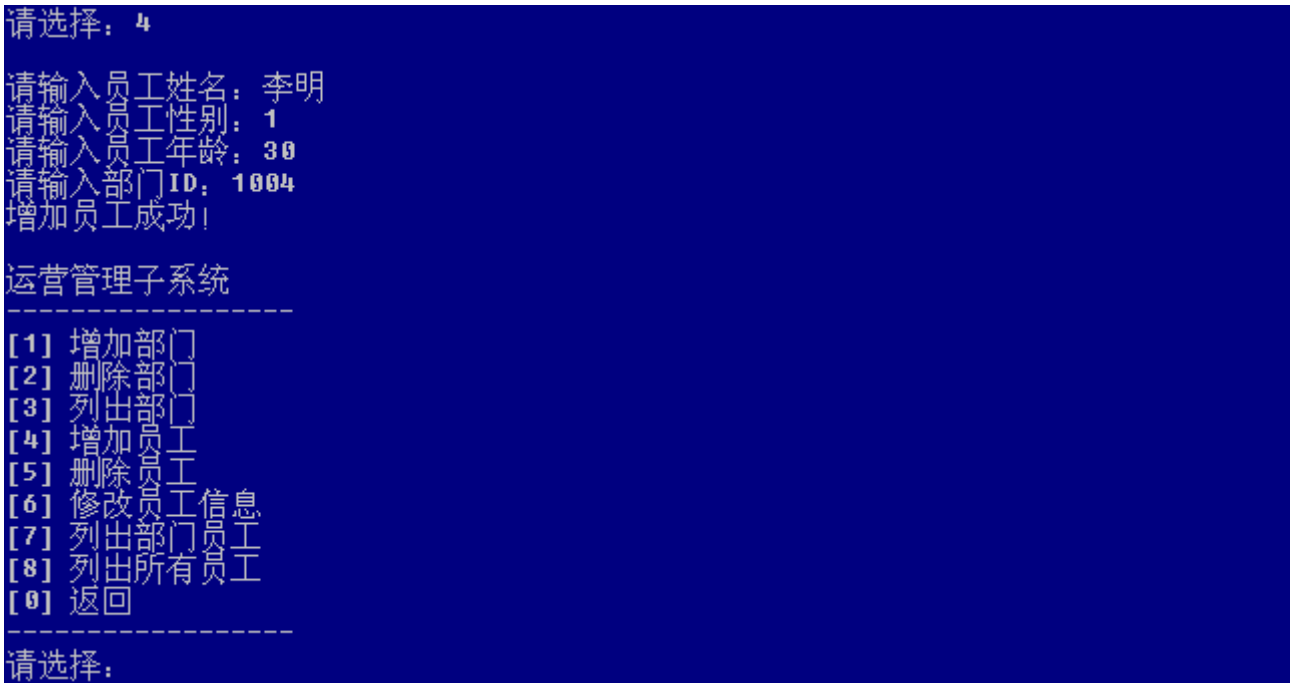
```
请选择：3

部门ID      部门名称                员工人数
----------------------------------------
1004        研发部                  0
1005        销售部                  0
1006        财务部                  0
1007        人力资源部              0

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.2. 员工管理

2.3.2.1. 增加员工

用户从运营管理子菜单选择[4]，进入增加员工功能模块，根据屏幕提示依次输入员工的姓名、性别、年龄，以及所属部门的ID号等信息。系统自动为其分配ID号，并向用户提供反馈信息。如图所示：



之后返回运营管理子菜单，等待用户后续操作。

2.3.2.2. 删除员工

用户从运营管理子菜单选择[5]，进入删除员工功能模块，根据屏幕提示输入欲删除员工的ID号。系统将该员工删除，并向用户提供反馈信息。如图所示：

```
请选择：5

请输入员工ID：1013
删除员工成功!

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.2.3. 修改员工信息

用户从运营管理子菜单选择[6]，进入修改员工信息功能模块，根据屏幕提示依次输入员工的ID号、姓名、性别、年龄等信息。系统更新与该员工有关的信息数据，并向用户提供反馈信息。如图所示：

```
请选择：6

请输入员工ID：1009
请输入员工姓名：刘宁胜
请输入员工性别：1
请输入员工年龄：36
修改员工信息成功!

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.2.4. 列出部门员工

用户从运营管理子菜单选择[7]，进入列出部门员工功能模块，根据屏幕提示输入部门的ID号。系统以列表形式显示该部门所有员工的ID号、姓名、性别和年龄。如图所示：

```
请选择：7

请输入部门ID：1004

员工ID      姓名                    性别      年龄
-----------------------------------------------
1008        李明                    1         30
1009        刘宁胜                  1         35
1010        陈飞                    1         35

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

2.3.2.5. 列出所有员工

用户从运营管理子菜单选择[8]，进入列出所有员工功能模块。系统以列表形式显示所有员工的ID号、姓名、性别和年龄。如图所示：

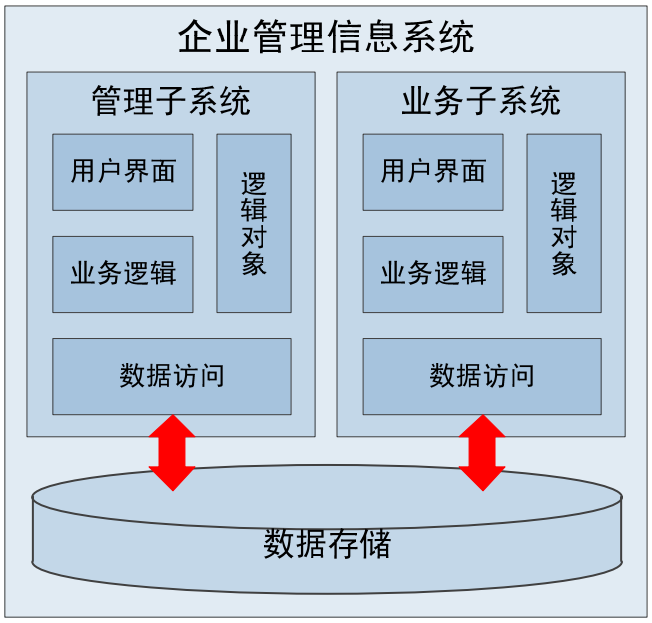
```
请选择：8

员工ID      姓名                    性别      年龄
-------------------------------------------------
1008        李明                    1         30
1009        刘宁胜                  1         35
1010        陈飞                    1         35
1012        张菁菁                  0         25
1011        孔祥戎                  0         28
1013        李小鹏                  1         27

运营管理子系统
------------------
[1] 增加部门
[2] 删除部门
[3] 列出部门
[4] 增加员工
[5] 删除员工
[6] 修改员工信息
[7] 列出部门员工
[8] 列出所有员工
[0] 返回
------------------
请选择：
```

之后返回运营管理子菜单，等待用户后续操作。

3. 概要设计

3.1. 总体架构

根据前述需求分析，本案在逻辑上可被划分为管理子系统和业务子系统两大模块，分别用于实现对管理员的管理和对部门及员工的管理功能。此外还需提供必要的数据存储策略，以实现对所有数据的持久化。系统总体架构如图所示：



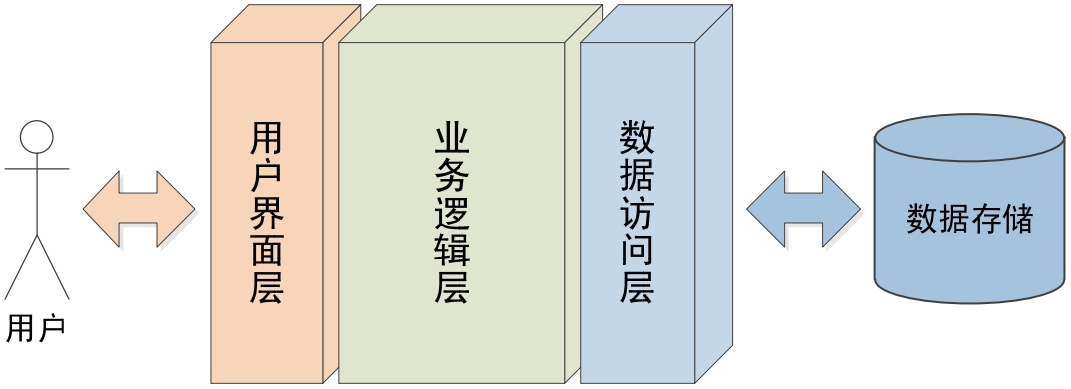
其中：

- 管理子系统：实现对管理员的管理功能。具体包括增加管理员、删除管理员、列出所有管理员。
  - 用户界面：显示主菜单、接受用户输入、向用户显示提示信息、处理结果和必要的反馈。
  - 业务逻辑：具体实现主菜单的各个功能项，以逻辑对象为载体，在用户界面和数据访问之间传递有关管理员的信息数据。
  - 数据访问：实现逻辑对象与数据存储之间的序列化与反序列化。
  - 逻辑对象：实现管理员对象的逻辑模型。
- 业务子系统：实现对部门及员工的管理功能。具体包括增加部门、删除部门、列出部门、增加员工、删除员工、修改员工信息、列出部门员工、列出所有员工。
  - 用户界面：显示运营管理子菜单、接受用户输入、向用户显示提示信息、处理结果和必要的反馈。
  - 业务逻辑：具体实现运营管理子菜单的各个功能项，以逻辑对象为载体，在用户界面和数据访问之间传递有关部门及员工的信息数据。
  - 数据访问：实现逻辑对象与数据存储之间的序列化与反序列化。
  - 逻辑对象：实现部门及员工对象的逻辑模型。
- 数据存储：实现整个管理信息系统的数据持久化。

3.2. 技术方案

3.2.1. 体系架构

本案在水平方向上采用三层体系架构。如图所示：

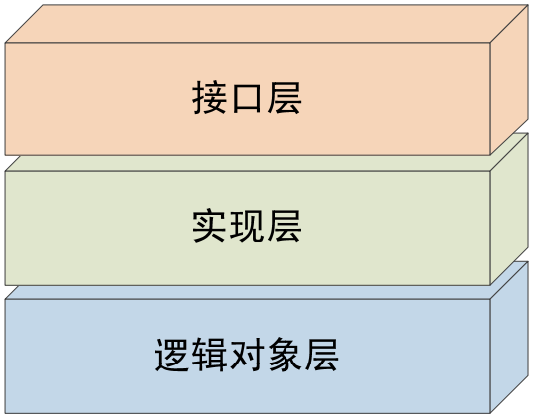


其中：

- 用户界面层：处理与最终用户的交互，既负责从用户处收集信息，也负责向用户展现结果、给出提示或反馈。
- 业务逻辑层：针对用户界面层所体现的功能项，以数据访问层为基础，实现与业务逻辑相关的算法和流程。
- 数据访问层：实现对数据存储介质的访问，为业务逻辑层提供数据源，并接受其处理结果。

3.2.2. 逻辑模型

本案在垂直方向上采用三层逻辑模型。如图所示：



其中：

- 接口层：定义各功能模块的抽象接口，降低模块间的耦合性，提高代码复用率，降低维护成本。
- 实现层：对抽象接口的具体实现。本案用户界面层的接口实现拟采用控制台方式，而数据访问层的接口实现则采用文件系统方式。
- 逻辑对象层：以逻辑模型的方式对系统中的相关数据加以组织，并构成从用户界面到业务逻辑再到数据访问各层之间的信息载体。本案的逻辑对象包括：管理员、部门和员工。

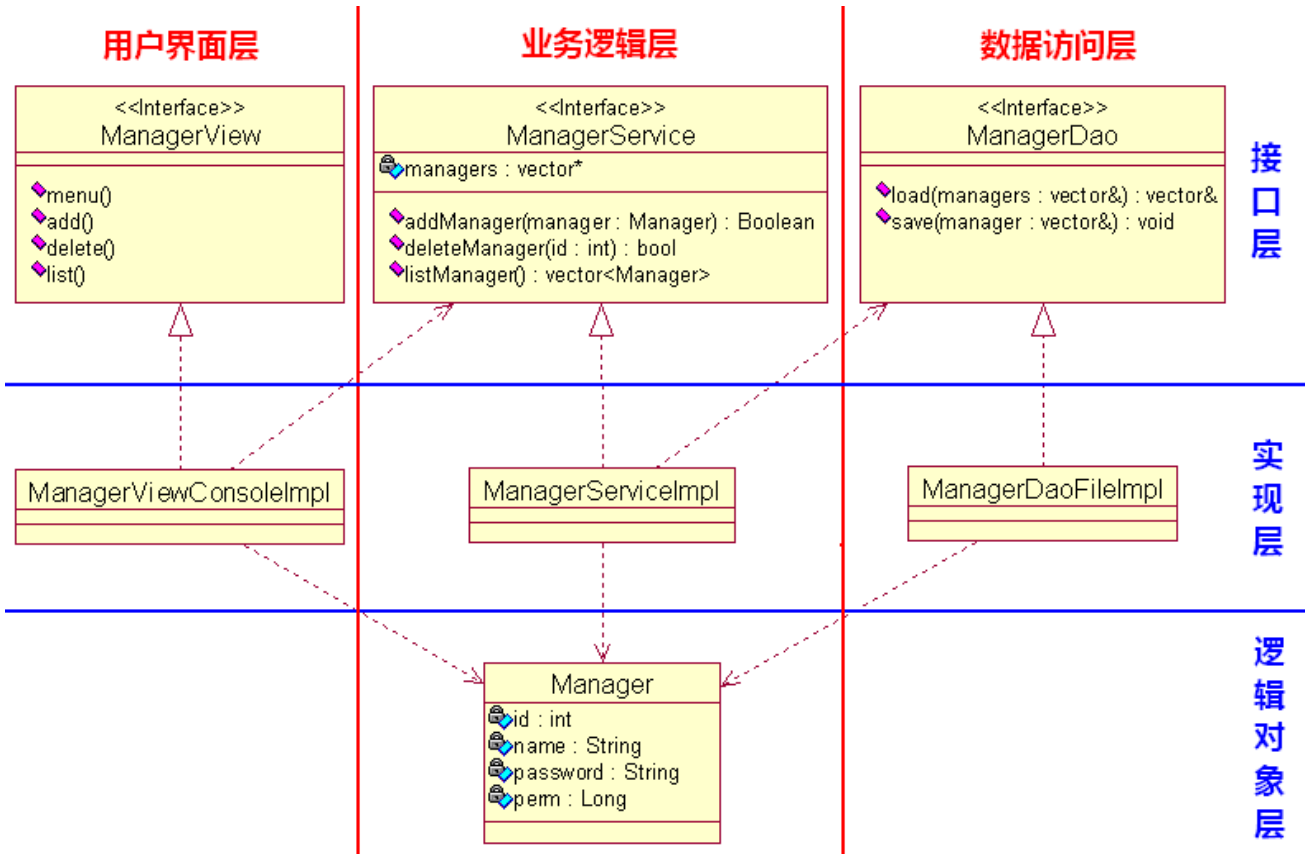
3.2.3. 平台约束

硬件环境	32 位 Intel x86 及其兼容处理器的个人计算机
操作系统	Fedora Linux R16
开发工具	GCC 4.6.2，C/C++标准库（STD）及标准模板库（STL）
应用类型	命令行应用程序
用户界面	非全屏模式的控制台字符界面
数据存储	二进制及纯文本文件
平台中立	不要求
交叉编译	不要求

4. 详细设计

4.1. 管理子系统

本案管理子系统由用户界面、业务逻辑、数据访问和逻辑对象四部分组成。其中用户界面、业务逻辑和数据访问又分别包括接口和实现两部分。如图所示：



4.1.1. 用户界面

4.1.1.1. 接口类：ManagerView

作为管理子系统用户界面层的接口类，ManagerView 类被定义为纯抽象类，由 4 个纯虚函数组成。

menu()	显示主菜单
add()	处理增加管理员菜单项
del()	处理删除管理员菜单项
list()	处理列出所有管理员菜单项

注释：斜体代表纯虚函数。

4.1.1.2. 实现类：ManagerViewConsoleImpl

作为管理子系统用户界面层的实现类，ManagerViewConsoleImpl 从纯抽象类 ManagerView 继承，并对基类中的 4 个纯虚函数提供覆盖版本。

menu()	通过控制台显示主菜单。在一个无限循环中不停显示菜单，接受用户选择，并根据用户所选菜单项调用其它三个

	接口函数。当用户选择进入运营管理子菜单时，应创建ServiceViewConsoleImpl对象，并调用其menu()接口。当用户选择退出时，终止循环，返回main()函数
add()	通过控制台处理增加管理员菜单项。根据用户输入的管理员用户名和密码创建Manager对象，调用ManagerService::addManager()接口函数增加管理员
del()	通过控制台处理删除管理员菜单项。根据用户输入的管理员ID号，调用ManagerService::deleteManager()接口函数删除管理员
list()	通过控制台处理列出所有管理员菜单项。调用ManagerService::listManager()接口函数获得管理员容器，遍历该容器并列表显示
ManagerService* service	业务逻辑对象。构造函数中动态创建ManagerServiceImpl对象

4.1.2. 业务逻辑

4.1.2.1. 接口类：ManagerService

作为管理子系统业务逻辑层的接口类，ManagerService 类被定义为纯抽象类，由 3 个纯虚函数组成。

addManager()	增加管理员
deleteManager()	删除管理员
listManager()	列出所有管理员

4.1.2.2. 实现类：ManagerServiceImpl

作为管理子系统业务逻辑层的实现类，ManagerServiceImpl 从纯抽象类 ManagerService 继承，并对基类中的 3 个纯虚函数提供覆盖版本。

addManager()	增加管理员。将从参数传入的Manager对象加入managers容器
deleteManager()	删除管理员。从managers容器中删除符合特定ID号的Manager对象
listManager()	列出所有管理员。返回managers容器
ManagerDao* dao	数据访问对象。构造函数中动态创建

	ManagerDaoFileImpl对象
`vector<Manager> managers`	管理员对象容器

4.1.3. 数据访问

4.1.3.1. 接口类：ManagerDao

作为管理子系统数据访问层的接口类，ManagerDao 类被定义为纯抽象类，由 2 个纯虚函数组成。

`load()`	从数据存储读取管理员信息
`save()`	将管理员信息写入数据存储

4.1.3.2. 实现类：ManagerDaoFileImpl

作为管理子系统数据访问层的实现类，ManagerDaoFileImpl 从纯抽象类 ManagerDao 继承，并对基类中的 2 个纯虚函数提供覆盖版本。

`load()`	从文件读取管理员信息。以二进制方式整块读取Manager对象，加入从参数传入的管理员容器
`save()`	将管理员信息写入文件。遍历从参数传入的管理员容器，以二进制方式整块写入每一个Manager对象

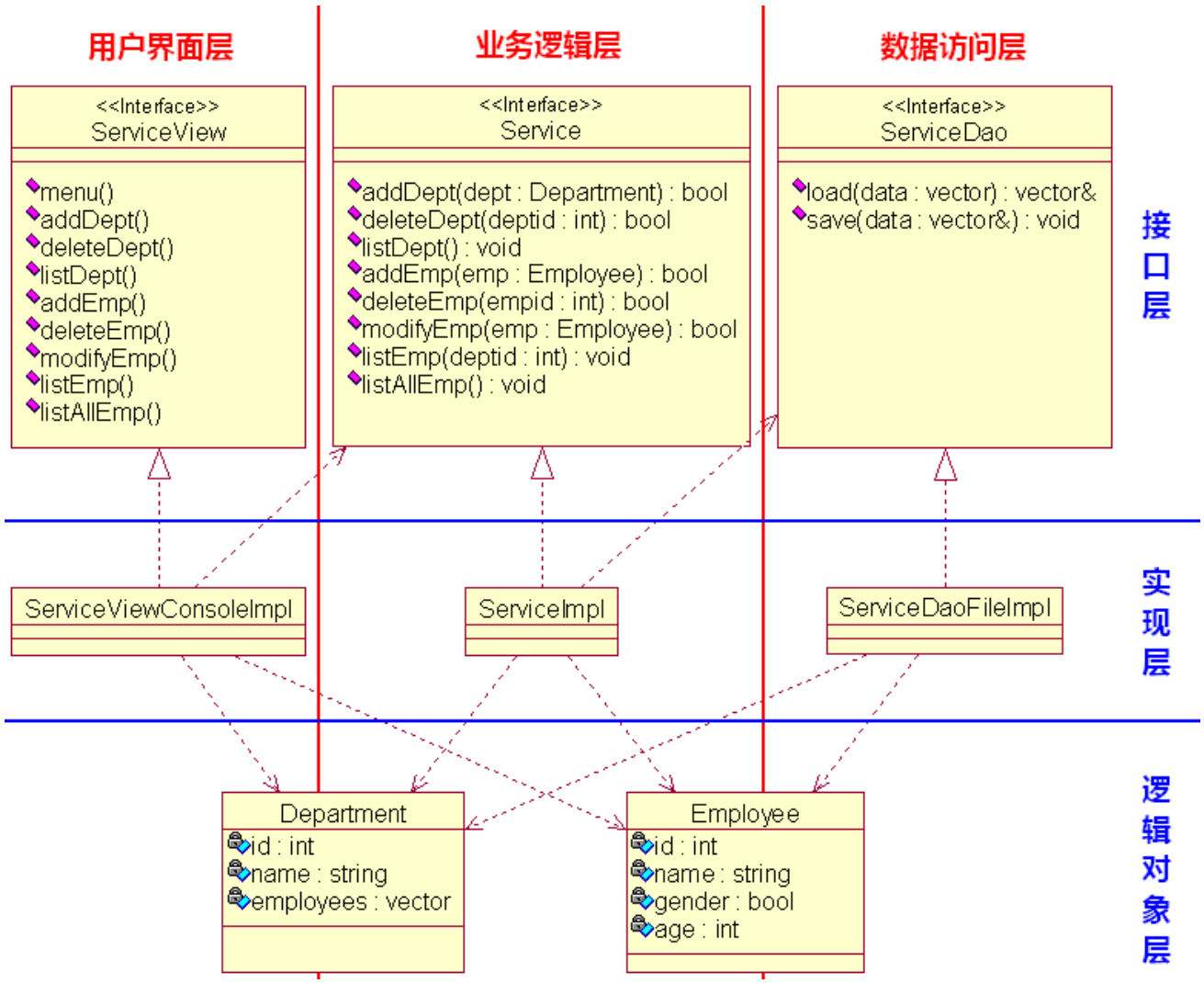
4.1.4. 逻辑对象

4.1.4.1. 管理员类：Manager

`int id`	ID号
`char name[20]`	用户名
`char password[20]`	密码

4.2. 业务子系统

本案业务子系统由用户界面、业务逻辑、数据访问和逻辑对象四部分组成。其中用户界面、业务逻辑和数据访问又分别包括接口和实现两部分。如图所示：



4.2.1. 用户界面

4.2.1.1. 接口类：ServiceView

作为业务子系统用户界面层的接口类，ServiceView 类被定义为纯抽象类，由 9 个纯虚函数组成。

menu()	显示运营管理子菜单
addDept()	处理增加部门菜单项
deleteDept()	处理删除部门菜单项
listDept()	处理列出部门菜单项
addEmp()	处理增加员工菜单项
deleteEmp()	处理删除员工菜单项
modifyEmp()	处理修改员工信息菜单项
listEmp()	处理列出部门员工菜单项

listAllEmp()	处理列出部门员工菜单项

4.2.1.2. 实现类：ServiceViewConsoleImpl

作为业务子系统用户界面层的实现类，ServiceViewConsoleImpl 从纯抽象类 ServiceView 继承，并对基类中的 9 个纯虚函数提供覆盖版本。

menu()	通过控制台显示运营管理子菜单。在一个无限循环中不停显示菜单，接受用户选择，并根据用户所选菜单项调用其它八个接口函数。当用户选择返回时，终止循环，返回ManagerViewConsoleImpl的menu()函数
addDept()	通过控制台处理增加部门菜单项。根据用户输入的部门名称创建Department对象，调用Service::addDept()接口函数增加部门
deleteDept()	通过控制台处理删除部门菜单项。根据用户输入的部门ID号，调用Service::deleteDept()接口函数删除部门
listDept()	通过控制台处理列出部门菜单项。调用Service::listDept()接口函数获得部门容器，遍历该容器并列表显示
addEmp()	通过控制台处理增加员工菜单项。根据用户输入的员工姓名、性别和年龄创建Employee对象。再根据用户输入的部门ID号，调用Service::addEmp()接口函数增加员工
deleteEmp()	通过控制台处理删除员工菜单项。根据用户输入的员工ID号，调用Service::deleteEmp()接口函数删除部门
modifyEmp()	通过控制台处理修改员工信息菜单项。根据用户输入的员工姓名、性别和年龄创建Employee对象，调用Service::modifyEmp()接口函数修改员工信息
listEmp()	通过控制台处理列出部门员工菜单项。根据用户输入的部门ID号，调用Service::listEmp()接口函数列出部门员工
listAllEmp()	通过控制台处理列出所有员工菜单项。调用Service::listAllEmp()接口函数列出所有员工
Service* m_pService	业务逻辑对象。构造函数中动态创建ServiceImpl对象

4.2.2. 业务逻辑

4.2.2.1. 接口类：Service

作为业务子系统业务逻辑层的接口类，Service 类被定义为纯抽象类，由 8 个纯虚函数组成。

addDept()	增加部门
deleteDept()	删除部门
listDept()	列出部门
addEmp()	增加员工
deleteEmp()	删除员工
modifyEmp()	修改员工信息
listEmp()	列出部门员工
listAllEmp()	列出所有员工

4.2.2.2. 实现类：ServiceImpl

作为业务子系统业务逻辑层的实现类，ServiceImpl 从纯抽象类 Service 继承，并对基类中的 8 个纯虚函数提供覆盖版本。

addDept()	增加部门。将从参数传入的Department对象加入m_vecDepts容器
deleteDept()	删除部门。从m_vecDepts容器中删除符合特定ID号的Department对象
listDept()	列出部门。返回m_vecDepts容器
addEmp()	增加员工。根据部门ID号找到该员工所隶属的部门，将参数Employee对象加入该部门的m_vecEmps容器
deleteEmp()	删除员工。依次调用每个部门的deleteEmp()接口函数
modifyEmp()	修改员工信息。依次调用每个部门的modifyEmp()接口函数
listEmp()	列出部门员工。根据部门ID号找到相应的部门，返回该部门的m_vecEmps容器
listAllEmp()	列出所有员工。依次调用每个部门listEmp()接口函数，将处理结果汇总到一个容器中返回
ServiceDao* m_pDao	数据访问对象。构造函数中动态创建ServiceDaoFileImpl对象

`vector<Department> m_vecDepts`	部门对象容器

4.2.3. 数据访问

4.2.3.1. 接口类：ServiceDao

作为业务子系统数据访问层的接口类，ServiceDao 类被定义为纯抽象类，由 2 个纯虚函数组成。

`load()`	从数据存储读取部门及员工信息
`save()`	将部门及员工信息写入数据存储

4.2.3.2. 实现类：ServiceDaoFileImpl

作为业务子系统数据访问层的实现类，ServiceDaoFileImpl 从纯抽象类 ServiceDao 继承，并对基类中的 2 个纯虚函数提供覆盖版本。

`load()`	从文件读取部门及员工信息。以文本方式读取全部部门和员工信息，加入从参数传入的部门容器
`save()`	将部门及员工信息写入文件。遍历从参数传入的部门容器，以文本方式写入每一个部门及其所有员工的信息

4.2.4. 逻辑对象

4.2.4.1. 部门类：Department

`deleteEmp()`	删除本部门的员工。根据从参数传入的员工ID号，从m_vecEmps容器中删除相应员工，没找到则返回false
`listEmp()`	列出本部门的员工。遍历m_vecEmps容器，将本部门的每位员工逐一加入从参数传入的员工容器
`modifyEmp()`	修改本部门的员工信息。根据从参数传入的员工对象的ID号属性，从m_vecEmps容器中找到相应员工，更新其信息，没找到则返回false
`int m_nId`	ID号
`string m_strName`	名称
`vector<Employee> m_vecEmps`	员工对象容器

4.2.4.2. 员工类：Employee

`int m_nId`	ID号
`string m_strName`	姓名
`bool m_bGender`	性别，true表示男性，false表示女性
`int m_nAge`	年龄

4.3. 基础设施及辅助工具

`generator_id()`	生成唯一ID号。从id.dat配置文件中读取上次生成的ID号，将其加1后重新写入，同时返回新生成的ID号

4.4. 配置文件

`id.dat`	唯一ID号配置文件。保存最后一次生成的ID号，每次新生成的ID号即在此基础上加1

4.5. 数据存储

`managers.dat`	管理员信息数据库。以二进制形式保存全部Manager对象
`services.dat`	部门及员工信息数据库。形如： …… **1004 研发部 3** 1008 李明 1 30 1009 刘宁胜 1 36 1010 陈飞 1 35 **1005 销售部 1** 1012 张菁菁 0 25 **1006 财务部 1** 1011 孔祥戎 0 28 …… 1. 其中以加粗字体显示的是部门记录，包括三个字段：部门ID号、部门名称和该部门的员工人数。程序可以以部门员工人数字段的值作为后续读取员工记录的循环控制上限； 2. 每个部门记录下面紧跟着隶属于该部门的员工记录，包

	括四个字段：员工ID号、员工姓名、员工性别（1表示男性，0表示女性）和员工年龄； 3. 为了便于程序通过标准I/O流进行格式化访问，各字段之间以空格分隔。

5. 文件组织

5.1. 代码文件

5.1.1. 管理子系统

managerview.h	定义ManagerView抽象基类
managerview_console_impl.h	声明ManagerViewConsoleImpl类
managerview_console_impl.cpp	实现ManagerViewConsoleImpl类
managerservice.h	定义ManagerService抽象基类
managerservice_impl.h	声明ManagerServiceImpl类
managerservice_impl.cpp	实现ManagerServiceImpl类
managerdao.h	定义ManagerDao抽象基类
managerdao_file_impl.h	声明ManagerDaoFileImpl类
managerdao_file_impl.cpp	实现ManagerDaoFileImpl类
manager.h	声明Manager类
manager.cpp	实现Manager类

5.1.2. 业务子系统

serviceview.h	定义ServiceView抽象基类
serviceview_console_impl.h	声明ServiceViewConsoleImpl类
serviceview_console_impl.cpp	实现ServiceViewConsoleImpl类
service.h	定义Service抽象基类
service_impl.h	声明ServiceImpl类
service_impl.cpp	实现ServiceImpl类
servicedao.h	定义ServiceDao抽象基类
servicedao_file_impl.h	声明ServiceDaoFileImpl类
servicedao_file_impl.cpp	实现ServiceDaoFileImpl类
department.h	声明Department类

department.cpp	实现Department类
employee.h	声明Employee类
employee.cpp	实现Employee类

5.1.3. 基础设施及辅助工具

main.cpp	定义main()函数
emis.h	声明全局变量
emis.cpp	定义全局变量
tools.h	声明工具函数
tools.cpp	定义工具函数

5.2. 脚本文件

makefile	项目制作脚本

6. 开发计划

鉴于本案的功能性及模块化特征，拟采用分步实施，迭代渐进的开发方式。在全体项目参与者对需求和设计全面了解的前提下，首先完成管理子系统用户界面部分的开发，包括抽象基类ManagerView及其控制台实现类ManagerViewConsoleImpl。在其测试验证通过后，加入逻辑对象Manager类、业务逻辑模块ManagerService抽象基类和ManagerServiceImpl实现类，同时完成辅助工具之唯一ID号生成函数的实现。此时应能满足针对管理子系统的非持久化功能验证。随后加入数据访问模块ManagerDao抽象基类和ManagerDaoFileImpl实现类，并进行针对管理子系统的功能性测试。

如果以上两步均能顺利完成，可考虑启动业务子系统用户界面部分的开发，包括抽象基类ServiceView及其控制台实现类ServiceViewConsoleImpl，同时将其调用接口以子菜单的形式加入管理子系统的主菜单中。在其测试验证通过后，加入逻辑对象Department和Employee类、业务逻辑模块Service抽象基类和ServiceImpl实现类。此时应能满足针对业务子系统的非持久化功能验证。随后加入数据访问模块ServiceDao抽象基类和ServiceDaoFileImpl实现类，并进行针对业务子系统的功能性测试。

至此，本案开发过程结束，项目进入测试阶段。提请测试部门拟定测试计划，并在收到研发部门测试通知书后两个工作日内启动测试进程。

以下开发计划按1人12小时（两个工作日）的工作量计算，具体实施过程可根据项目推进情况灵活调整，但最好能保证一个子系统的完整实现。

1	项目启动		编写需求分析、概要设计和详细设计，制定开发计划，完成评审	1.5h
2	管理子系统	用户界面	定义ManagerView抽象基类和ManagerViewConsoleImpl实现类，定义main()函数，编写makefile，构建并测试	1.5h
3		业务逻辑	定义Manager类、ManagerService抽象基类和ManagerServiceImpl实现类、唯一ID号生成函数，连接用户界面，修改makefile，构建并测试	1.5h
4		数据访问	定义ManagerDao抽象基类和ManagerDaoFileImpl实现类，连接业务逻辑，修改makefile，构建并测试	1.5h
5	业务子系统	用户界面	定义ServiceView抽象基类和ServiceViewConsoleImpl实现类，加入主菜单，修改makefile，构建并测试	1.5h
6		业务逻辑	定义Department和Employee类、Service抽象基类和ServiceImpl实现类，连接用户界面，修改makefile，构建并测试	1.5h
7		数据访问	定义ServiceDao抽象类和ServiceDaoFileImpl实现类，连接业务逻辑，修改makefile，构建并测试	1.5h
8	集成测试		测试验证全部功能性需求	1.5h
9	提交测试		发布测试通知，进入测试阶段	—

7. 风险评估

风险预测	危害程度	发生概率	规避策略
环境异常	中	中	MIS 协助解决
人事变动	中	小	项目经理负责
需求变更	低	中	产品经理负责
不可抗力	高	小	—

8. 项目总结

本项目的研发旨在提供一种，建立在有限技术基础之上的，适度规模的，相对较接近于真实场景的，具有一定可扩展性的典型开发案例。以下几个方面共同构成本项目的研发重点：

- 相对完整的项目开发流程：
  本案基本采用传统意义上的瀑布式开发流程，从需求分析入手，到概要设计、详细设计，再到编码和测试。整个过程环环相扣，每一个步骤都严格依赖于前一个步骤的成果和结论。但在编码阶段，本案又结合具体项目的具体特点，引入了某些现代敏捷开发的思想和理念，以一种小幅渐进的方式，提高了可交付物的产出效率。

- 多层次的体系与模型设计：
  本案一方面在体系架构上采用了由用户界面、业务逻辑和数据访问共同组成的三层结构。同时又

在每一层的内部自上而下地进行了从接口到实现再到逻辑对象的三层划分。高聚低耦乃是本项目的开发宗旨之一。

- 多源文件系统的构建：
  本案最终由多个.h和.cpp文件组成，通过makefile可以对工程的构建方式进行定制。这也是现实世界中常见的工程构建方式之一。一方面通过文件将逻辑层面彼此独立的类和函数组合进行物理层面的划分，另一方面又将类及函数的声明和实现分别放到.h和.cpp中，同时保持文件名的相关性。这样既有利于团队间的分工协作，又突出了相对稳定的接口定义在大型项目开发过程中的重要性。

- 基于控制台的字符界面和基于文件的数据存储：
  基于控制台的字符界面和基于文件的数据存储，虽然难度不大，但仍有许多细节需要考虑，比如，用户输入的数据不正确如何处理，数据文件损坏或尚未创建如何处理等。在项目开发的过程中越涉及到底层的细节，越需要谨慎对待。

- 借助逻辑对象实现业务逻辑的算法与控制：
  在三层体系架构中，业务逻辑层往往最复杂且不易调试。因其与具体业务相关，借助C++语言的建模能力往往能达到事半功倍的效果。

附录 A：makefile 样例

```
CC = g++

EXEC = emis

OBJS = main.o emis.o tools.o \
      manager.o employee.o department.o \
      managerview_console_impl.o \
      managerservice_impl.o \
      managerdao_file_impl.o \
      serviceview_console_impl.o \
      service_impl.o \
      servicedao_file_impl.o

all: $(EXEC)

$(EXEC): $(OBJS)
      $(CC) -o $@ $^

clean:
      rm $(EXEC) $(OBJS)
```

附录 B：参考实现

ftp://ftp.tarena.com.cn