华中科技大学计算机科学与技术学院

C语言课程设计报告

题目：科研项目信息管理系统

专　　业：计算机科学与技术

班　　级：计算机科学与技术（校交）1601

学　　号：U201610136

姓　　名：朱晓光

成　　绩：

指导教师：甘早斌

完成日期：　2017 年　 8 月　 22 日

目录

该课程设计项目在 GitHub 上的地址：

https://github.com/smdsbz/curriculum-design-1

一、 系统需求分析

科研项目信息管理系统用于管理各院系科研项目相关信息，主要包括院系基本信息、科研团队基本信息和科研项目基本信息，方便有关部门对科研项目信息进行维护、查询、统计，提高工作效率。

科研项目信息管理系统要求实现一下几个方面的基本功能：

1. 基本信息的录入、修改和删除功能

科研信息管理系统的基本信息主要包括以下三类：

（1） 院系基本信息：院系名称、负责人姓名、联系电话等数据项。

（2） 科研团队基本信息：团队名称、负责人姓名、教师人数、研究生人数、团队所属院系等数据项。

（3） 科研项目基本信息：项目编号、项目类别、起始时间、项目经费、负责人姓名、项目所属团队等数据项。

系统应能实现以上三种基础数据信息的录入、修改和删除功能。在信息录入时，应简明的标识出所有用户应当提供的数据项，尽量一次输入就能够完成对象的添加；同时，在添加对象的时候应该还提供信息校验功能，保证添加的数据项合理、正确，能够形成正确的层级关系。

2. 基本信息的查询功能

系统应该实现对以上三种基础数据信息的查询功能，提供按多种条件分别进行查询的方式。具体包括：

（1） 按院系负责人姓名查询院系信息。

（2） 按院系名称的全部或一部分查询院系信息。

（3） 按团队名称的全部或一部分查询团队信息。

（4） 以教师人数为条件查询团队信息。

（5） 按项目编号查询项目信息。

（6） 按所属团队查询项目信息。

上述查询结果中，如果有多条信息被查中，则需提供逐条显示再进行选择的功能。

3. 数据统计功能

在以上三种基础数据信息的基础上，提供多方面的数据统计功能，具体包括：

（1） 统计各院系教师总数、研究生总数以及研究生与教师的人数比（保留两位小数），按学生教师人数比值降序排序后输出。

表 1.3.1 院系学生教师人数统计表

编号	院系名称	教师总数	学生总数	学/教比值
1	计算机学院	78	799	10.24
2	物理学院	45	456	10.13
...				

（2） 统计某年度各院系科研项目数、973 项目数、863 项目数以及科研总经费，按科研项目数降序排序后输出。

表 1.3.2 院系科研项目数统计表

编号	院系名称	项目总数	973 项目数	863 项目数	科研经费
1	计算机学院	3	2	0	15.84
2	物理学院	1	0	1	12.30
...					

（3） 统计历年来类别为国家自然科学基金的科研项目数最多的 10 个科研团队， 按项目数降序排序后输出科研团队名称、国家自然科学基金项目数以及项目经费总数。

表 1.3.3 国家自然科学基金项目团队统计表

编号	团队名称	NSFC 项目数	项目经费
1	火箭队	1	2.30
2	银河队	3	12.30
...			

（4） 统计科研项目数和教师人数的比值最高的 5 个科研团队，按比值（保留两位小数）降序排序后输出科研团队名称、教师人数、科研项目数、项目数与教师人数比值。

表 1.3.4 团队教师分配统计表

编号	团队名称	教师人数	项目总数	项目/教师比值
1	火箭队	2	2	1.00
2	银河队	4	1	4.00
...				

（5） （自设）统计项目平均经费最多的 5 个院系，按平均经费降序排序后输出院系名称、项目总数、总科研经费、项目平均经费。

表 1.3.5 院系平均项目经费统计表

编号	院系名称	项目总数	科研经费	项目平均经费
1	计算机学院	1	12.30	12.30
2	物理学院	3	15.84	5.28
...				

4. 数据存储功能

以上三种信息在程序运行时，以可以动态扩展的链表的形式存储在内存中；同时，在硬盘上以数据文件的形式实现长期保存。在程序退出时将内存中的数据与硬盘上的数据进行同步，从而保证数据的一致性以及安全性。

二、 总体设计

根据系统需求分析结果，将整个系统在宏观上分解成了以下几个功能模块，方便具体实现：

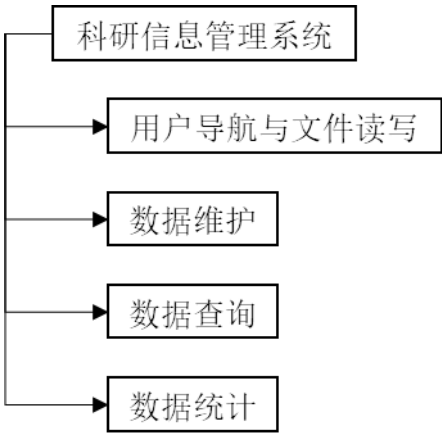

图 2.0 科研信息管理系统功能模块

1. 用户导航与文件读写

文件读写模块包括一系列与系统启动时环境建立与系统结束运行时数据保存的操作。它进一步划分为两个模块：数据加载与数据保存。

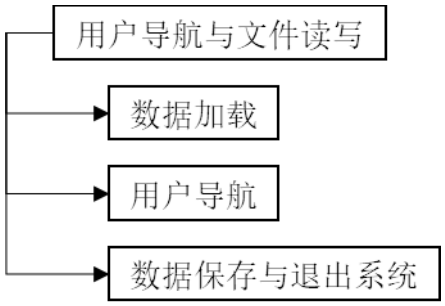

图 2.1 用户导航与文件读写模块的子模块划分

- 数据加载子模块：用于将存储在硬盘中的数据读取到内存中，并形成链表结构。在此阶段需要用户指明数据文件所在文件夹的路径，否则在程序根路径寻找数据文件，若未找到完整的数据文件则询问是否创建新的数据文件。
- 用户导航子模块：用于在程序运行的各个阶段向用户提示当前可用操作，使用户能够在系统的不同子模块中进行切换，并能够导航到任意对象。
- 数据保存与退出系统子模块：将内存中的数据同步到硬盘上的数据文件中（其路径在数据加载子模块中指明），释放程序动态申请的内存区域，结束系统运行。

2. 数据维护

数据维护模块负责对三种基础数据信息的录入、修改和删除功能，需要保证数据的准确性、完整性和有效性。该模块按信息种类进一步划分为院系信息维护、团队信息维护、项目信息维护三个子模块。

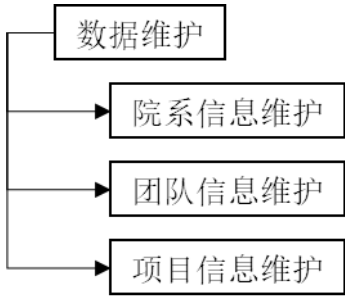

图 2.2 数据维护模块的子模块划分

- 院系信息维护子模块、团队信息维护子模块和项目信息维护子模块：三个模块分别负责用户对三种基本信息的录入、修改、删除等操作。要确保基础数据的正确性，确保用户在操作以上三种基础信息的时候不会破坏数据的层级结构。

3. 数据查询

数据查询模块提供对三种基础数据按多种条件进行查询的功能。该模块按信息种类分为院系信息查询、团队信息查询和项目信息查询。

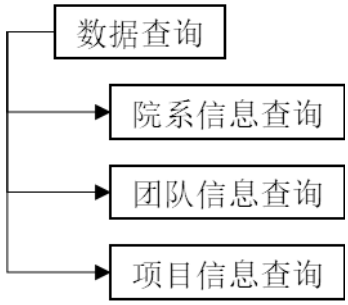

图 2.3 数据查询模块的子模块划分

- 院系信息查询子模块：用于提供两种方式查询院系基本信息——一种是按院系负责人姓名进行精确查询；另一种是按院系名称的全部或部分进行模糊查询。列出所有符合条件的查询结果后，用户能够选择其中一个或者退出查询。
- 团队信息查询子模块：用于提供两种方式查询团队基本信息——一种是按团队名称的全部或一部分进行模糊查询；另一种是以教师人数为条件进行模糊查询。

- 项目信息查询子模块：由于团队信息查询子模块可以定位到项目所属的团队，该模块只提供系统需求中另一种方式查询团队基本信息——按项目编号进行精确查询。

4. 数据统计

数据统计模块提供对三种基础数据进行多方面统计的功能。按统计条件，该模块划分为 5 个子模块。

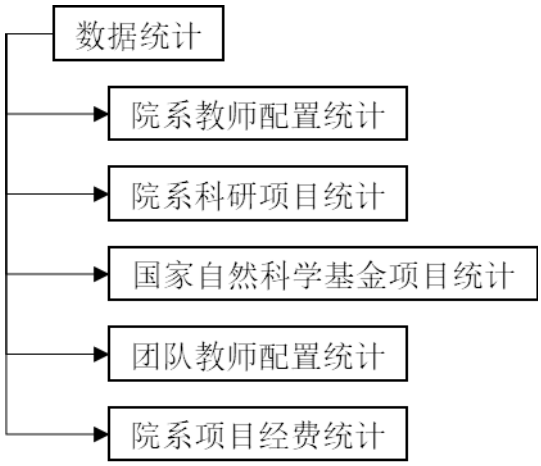

图 2.4 数据统计模块的子模块划分

- 院系教师配置统计子模块：用于统计各院系的教师总数与学生总数，并由此计算学生与老师人数比（保留两位小数），按人数比降序排序后输出统计信息。
- 院系科研项目统计子模块：用于统计某年度各院系科研项目总数、973 项目数、863 项目数以及科研总经费，按科研项目总数降序排序后输出统计信息。
- 国家自然科学基金项目统计子模块：用于统计各团队国家自然科学基金项目总数与该类项目经费，按项目数降序排序后输出排名前 10 团队的统计信息。
- 团队教师配置统计子模块：用于统计团队项目总数与教师总人数，并由此计算项目数与教师人数比值（保留两位小数），按比值降序排序后输出排名前 5 团队的统计信息。
- （自设）院系项目经费统计子模块：用于统计院系项目数与科研总经费，并由此计算项目的平均经费（保留两位小数），按比值降序排序后输出排名前 5 院系的统计信息。

三、 数据结构设计

按照任务要求，系统需要处理的基础信息有三种：院系基本信息、团队基本信息和项目基本信息。这三种信息之前存在这样的关联：院系下属有团队、团队开设项目。根据实现需要，每种基础数据结构有另外的包装结构，此外还增设了用户指针、头节点挂载点组教师人数查找条件等数据结构。下面分别展示每种数据结构的设计。

1. 院系数据结构

a） 数据文件存储结构

院系基础数据信息如下表所示。院系基础数据将以 DepartData 结构体的形式存储在“DEPART.DAT”文件中。

表 3.1.a 院系基本信息表

数据结构标识		DepartData	
数据项	**标识**	**类型**	**示例**
院系名称	name	char[20]	“Computer”
负责人	manager	char[12]	“Zhang3”
联系电话	mobile	char[15]	“13313371337”

b） 内存中存储结构

- 存储用：Depart 结构体用于链接在内存中的所有院系数据成链表。

表 3.1.b.1 院系节点存储用结构体

数据结构标识	Depart	
数据项	**标识**	**类型**
院系数据域	data	DepartData *
下一个院系节点	next	Depart *
所属团队链头节点	child_team_head	Team *
所属团队链尾节点	child_team_tail	Team *

- 搜索用：DepartWrapper 结构体用于返回只含符合搜索条件院系的链表。

表 3.1.b.2 院系搜索用结构体

数据结构标识	DepartWrapper	
数据项	**标识**	**类型**
院系节点	depart	Depart *
下一个节点	next	DepartWrapper *

- 统计用：DepartStatWrapper 结构体用于返回按要求进行统计、排序后的院系链表。

表 3.1.b.3.1 院系统计用数据结构体

数据结构标识	DepartStatData	
数据项	**标识**	**类型**
学生总人数	student_num	int
教师总人数	teacher_num	int
项目总数	project_total	int
973 项目总数	project_973	int
863 项目总数	project_863	int
科研总经费	funding	float

表 3.1.b.3.2 院系统计用节点结构体

数据结构标识	DepartStatWrapper	
数据项	**标识**	**类型**
院系节点	depart	Depart *
统计数据	stat	DepartStatData
下一节点	next	DepartStatWrapper *

2. 团队数据结构

a） 数据文件存储结构

团队基础数据信息如下表所示。团队基础数据将以 TeamData 结构体的形式存储在“TEAM.DAT”文件中。

表 3.2.a 团队基本信息表

数据结构标识	TeamData	
数据项	**标识**	**类型**
团队名称	name	char[30]
负责人	manager	char[12]
教师人数	teacher_num	int
学生人数	student_num	int
所属院系名称	faculty	char[20]

b） 内存中存储结构

- 存储用：Team 结构体用于链接在内存中的所有团队数据成链表。

表 3.2.b.1 团队节点存储用结构体

数据结构标识	Team	
数据项	**标识**	**类型**
团队数据域	data	TeamData *
下一节点	next	Team *
开设项目链头节点	child_project_head	Project *

开设项目链尾节点	child_project_tail	Project *
所属院系节点	parent_depart	Depart *

- 搜索用：TeamWrapper 结构体用于返回只含符合搜索条件团队的链表。

表 3.2.b.2 团队搜索用结构体

数据结构标识	TeamWrapper	
数据项	**标识**	**类型**
团队节点	team	Team *
下一节点	next	TeamWrapper *

- 统计用：TeamStatWrapper 结构体用于返回按要求进行统计、排序后的团队链表。

表 3.2.b.3.1 团队统计用数据结构体

数据结构标识	TeamStatData	
数据项	**标识**	**类型**
项目总数	project_total	int
国家自然科学基金项目数	project_NSFC	int
总经费	funding	float

表 3.2.b.3.2 团队统计用节点结构体

数据结构标识	TeamStatWrapper	
数据项	**标识**	**类型**
团队节点	team	Team *
统计数据	stat	TeamStatData
下一节点	next	TeamStatWrapper

3. 项目数据结构

a） 数据文件存储结构

项目基础数据信息如下标所示。项目基础数据将以 ProjectData 结构体的形式存储在“PROJECT.DAT”文件中。

表 3.3.a.1 项目基本信息表

数据结构标识	ProjectData	
数据项	**标识**	**类型**
项目编号	id	char[15]
项目类别	type	char
起始时间	start_date	char[8]
项目经费	funding	float
负责人	manager	char[12]
所属团队名称	team	char[30]

表 3.3.a.2 基本项目信息项目类别代码表

代码	对应项目类别
1	973 计划项目
2	国家自然科学基金项目
3	863 计划项目
4	国际合作项目
5	横向项目

b）内存中存储结构

- 存储用：Project 结构体用于链接在内存中的所有项目数据成链表。

表 3.3.b.1 项目节点存储用结构体

数据结构标识	Project	
数据项	**标识**	**类型**
项目数据域	data	ProjectData *
下一节点	next	Project *
开设团队节点	parent_team	Team *

- 搜索用：ProjectWrapper 结构体用处返回只含符合搜索条件项目的链表。

表 3.3.b.2 项目搜索用结构体

数据结构标识	ProjectWrapper	
数据项	**标识**	**类型**
项目节点	project	Project *
下一节点	next	ProjectWrapper

注：院系基本信息表、团队基本信息表与项目基本信息表在内存中构成类似于三向十字交叉链表的结构，如下图所示。

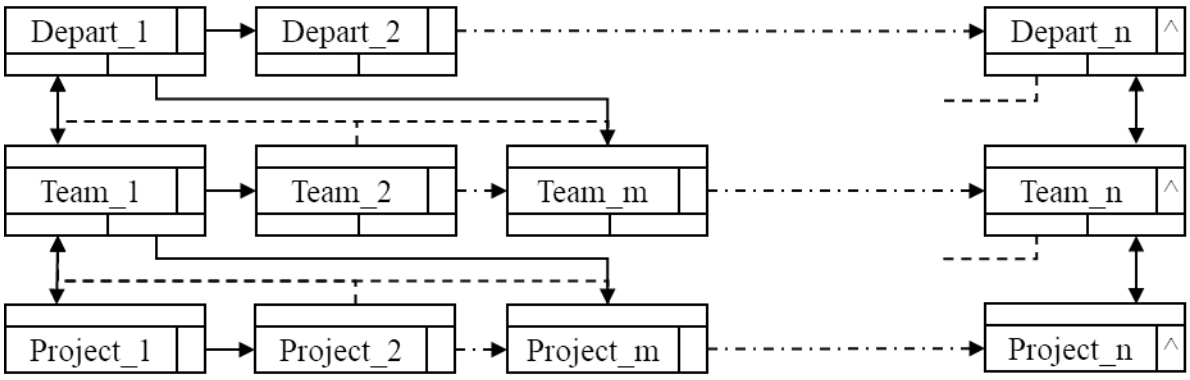

图 3.3 基础数据存储链表结构示意图

4. 用户指针数据结构

用户指针用于记录用户当前在链表中的位置，方便用户导航功能的实现。

表 3.4.1 用户指针结构体

数据结构标识	Cursor	
数据项	**标识**	**类型**
指向对象类型	type	int
指向对象地址	val	void *

表 3.4.2 用户指针结构体指向对象类型代码表

代码	**对应指向对象类型**
1	院系
2	团队
3	项目
0	置空

5. 头节点挂载点组数据结构

头节点挂载点组的设置旨在方便管理院系链表、团队链表和项目链表的头节点。将该数据在全局申明，可以大大减少向视图函数传递头节点指针的次数。

表 3.5 头节点挂载点组结构体

数据结构标识	MountPoint	
数据项	**标识**	**类型**
院系链表头节点	depart_head	Depart *
团队链表头节点	team_head	Team *
项目链表投节点	project_head	Project *

6. 教师人数查找条件数据结构

教师人数查找条件结构体的设置旨在向负责查询团队的函数传递查找条件的时候，减少参数个数，抽象查找条件并同一查询函数调用格式。

表 3.6 教师人数查找条件结构体

数据结构标识	Where	
数据项	**标识**	**类型**
查询方向	direction	char[3]
查询阈值	value	int

（其中查询方向可能的取值有“<”、“<=”、“=”、“>”、“>=”）

四、 详细设计

基于以上对任务需求的分析以及设计，下面展示主体功能实现的算法。

下图展示主程序运行流程。

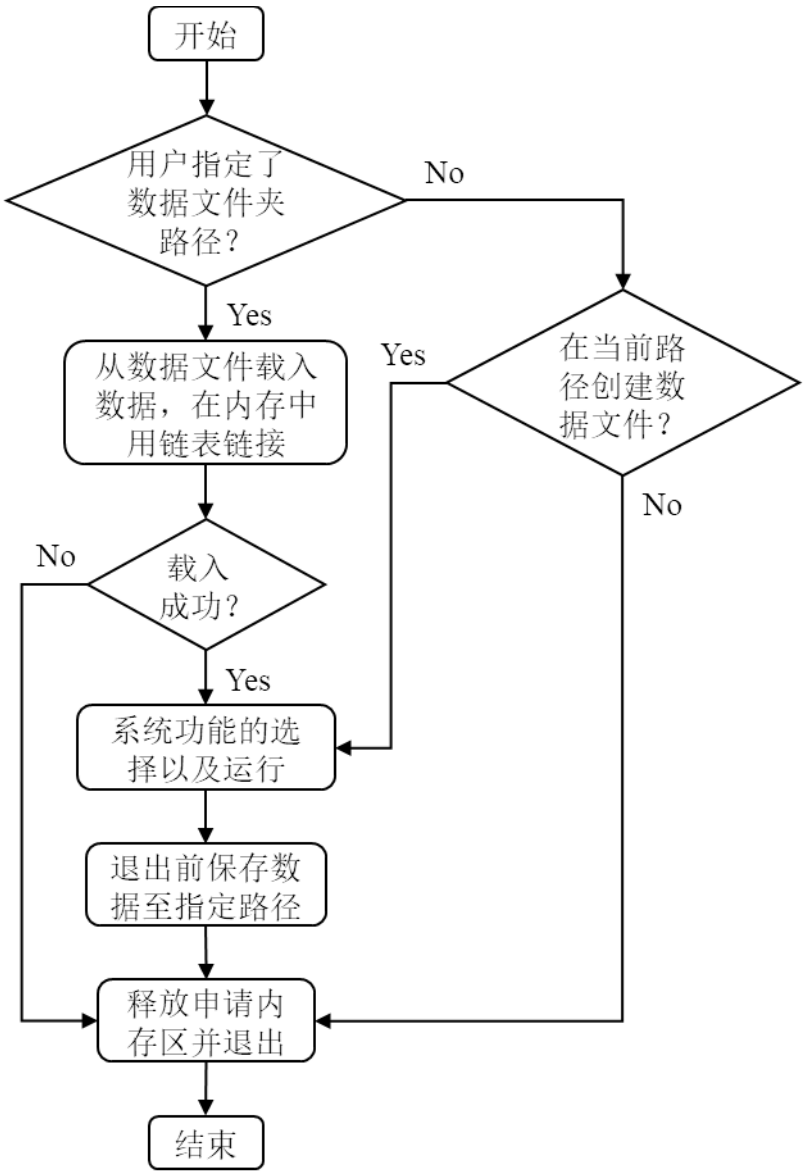

图 4 主程序运行流程图

下面详细展示载入数据部分以及系统功能的选择以及运行部分的设计。

1. 用户导航与文件读写

a） 数据加载

数据加载功能又可以分为三个子过程进行：加载院系数据、加载团队数据和加载项目

数据。加载三类数据的同时也要正确构建起数据之间的层级结构。

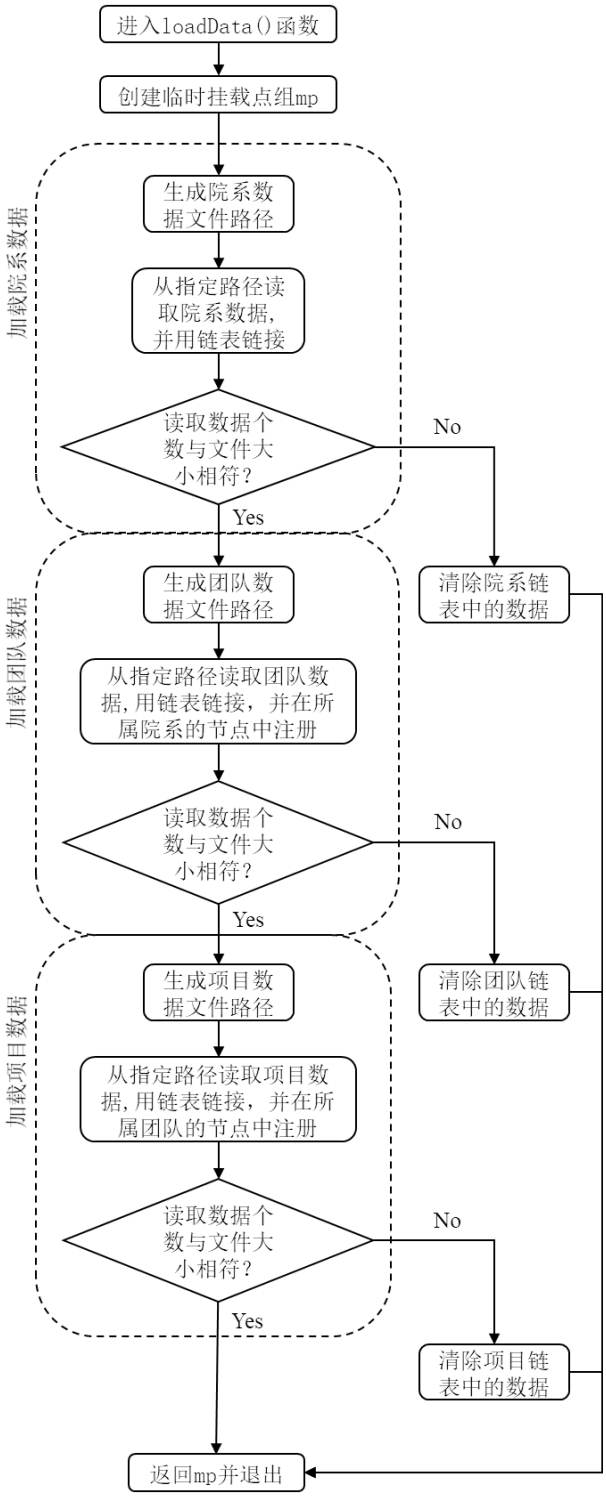

图 4.1.a 数据加载子模块流程图

注：由于在存储的时候是有序的，所以加载时不需要考虑新数据插入的位置，按照“先进后出”的顺序读取即可。

注： loadData() 函数中关于用链表链接数据的部分均由另定义的函数（appendDepart()、appendTeam()和 appendProject()）实现。

b）用户导航

该软件采用了字符界面，即程序将会列出用户当前可用的操作，用户将输入数字来选择列出操作中的某一项来执行。由于这里没有窗口的概念，一个逻辑清晰的用户导航是必须的。下图展示了该软件的用户导航逻辑。

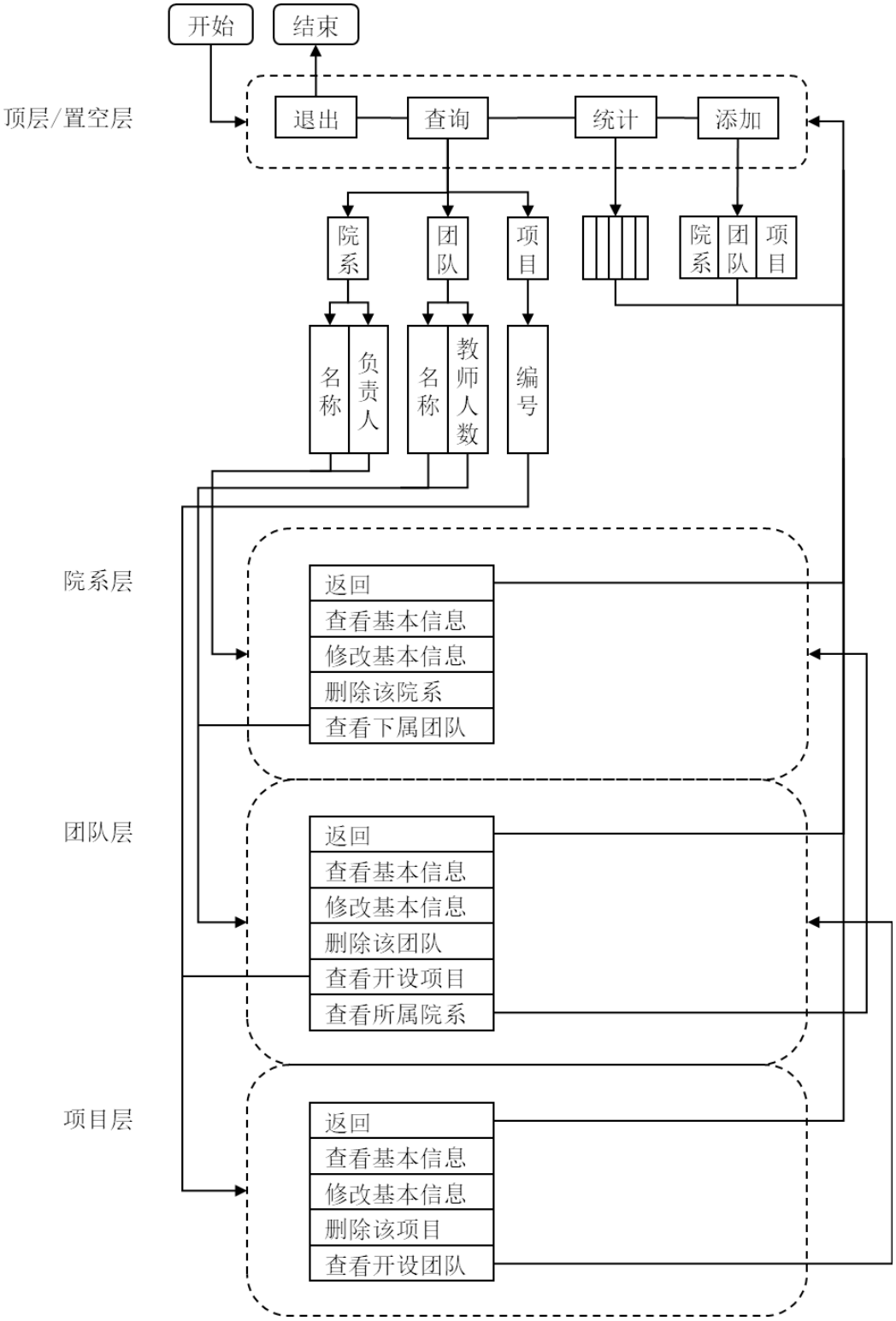

图 4.1.b 用户导航逻辑示意图

c） 数据保存与退出系统

- 数据保存：这里的逻辑与数据加载部分类似。

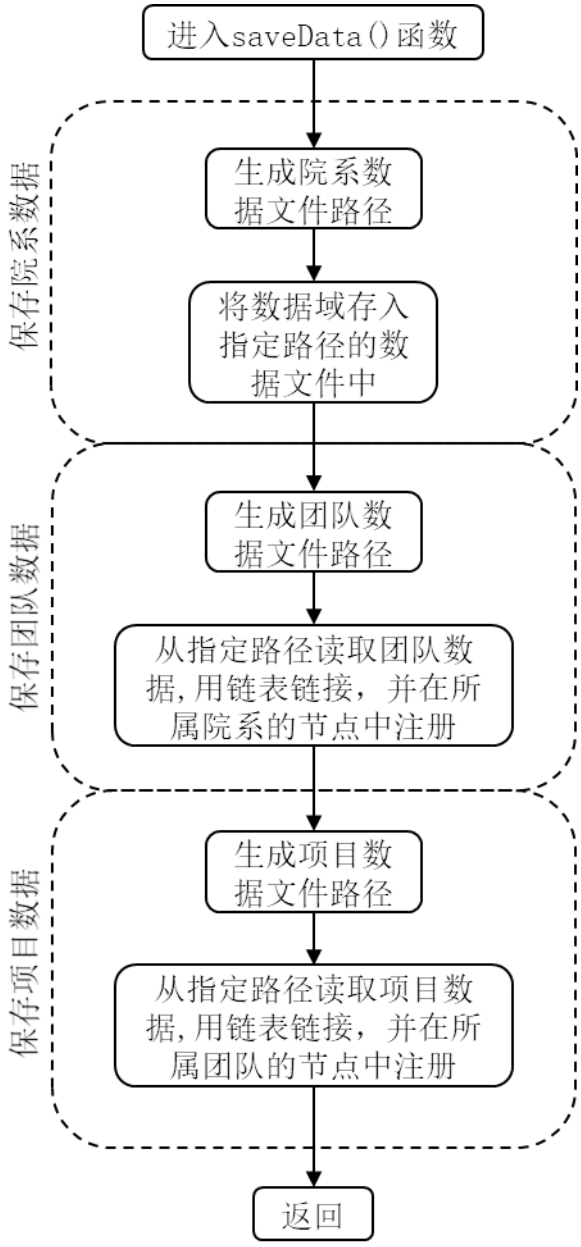

图 4.1.c 数据保存流程图

注：向数据文件写数据用“wb”模式，因此此操作将覆盖之前的记录或者新建数据文件。

- 退出系统：退出系统部分主要负责释放程序运行中所有申请的内存空间，该部分集成到用户导航的流程中实现，故在此不给出流程示意图。

2. 数据维护

由于对于三种基础数据进行维护的操作的重复度很高，在此只给出最具代表性、最复杂的对团队基础数据维护的详细流程。

a） 院系信息维护 **c） 项目信息维护** 略

b） 团队信息维护

（1） 团队信息添加：为了不破坏已有链表结构的完整性，在新添加团队节点的时候不能只是简单地在团队链表末尾添加节点，而是要找到所属院系节点下的团队节点区块，插入到该区块的尾部。

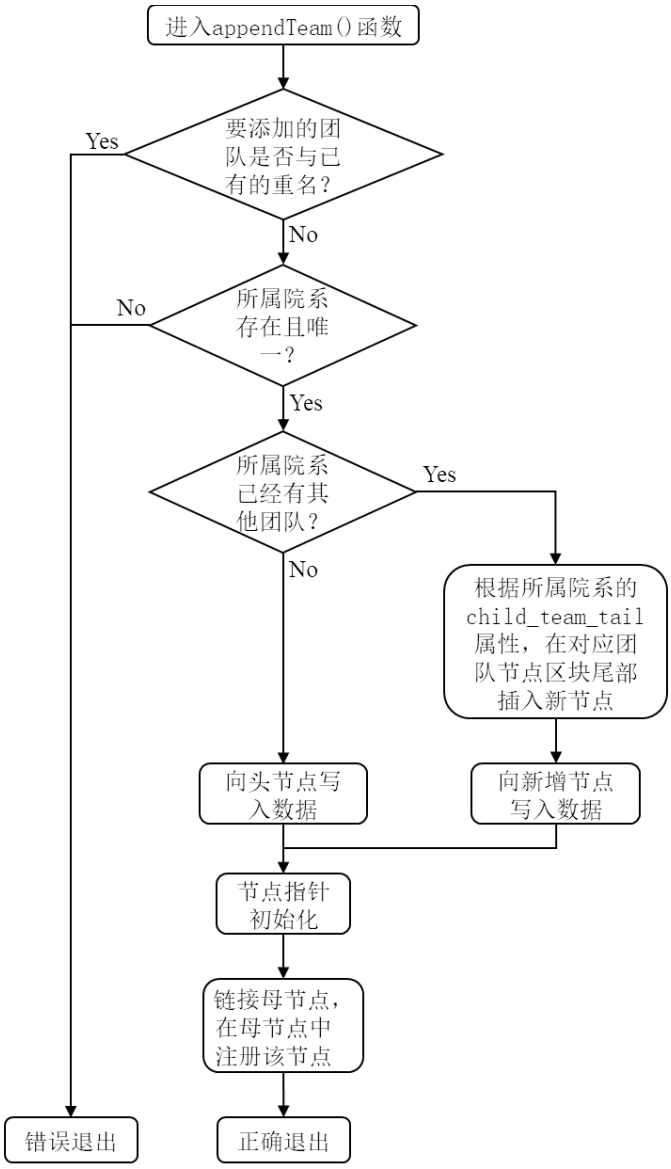

图 4.2.b.1 团队信息添加流程图

（2） 团队信息修改：为了规范用户行为，防止类似于直接对团队所属院系名

称进行修改，而未意识到链表结构准确性会被破坏的违规操作出现，此功能集成到用户导航部分实现。

（3） 团队信息删除：删除节点的时候要注意不能够破坏链表结构的完整性。

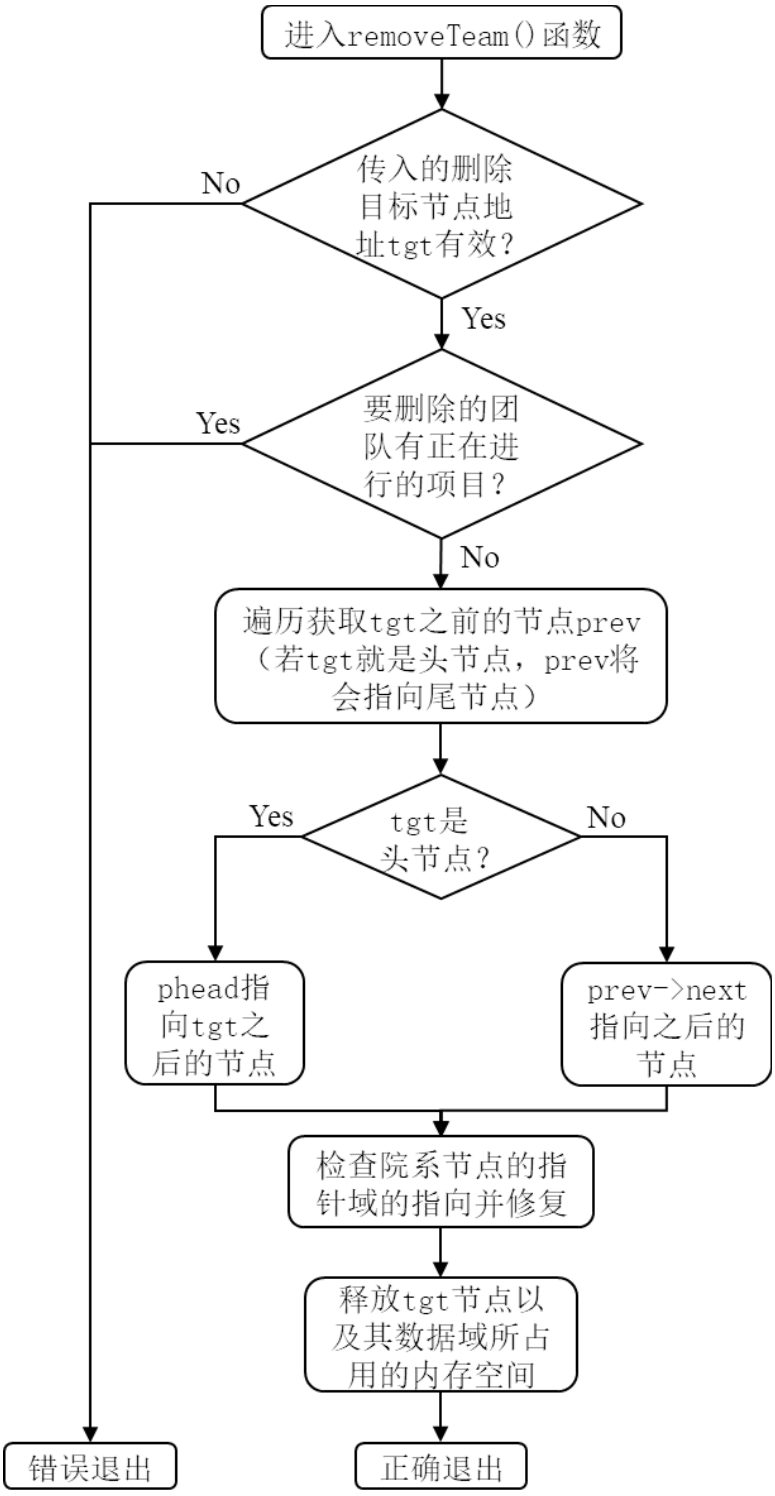

图 4.2.b.3 团队信息删除流程图

3. 数据查询

由于数据查询的实现过程基本上只有条件判断的部分不同，故在此只给出按教师人数查找团队的详细流程。

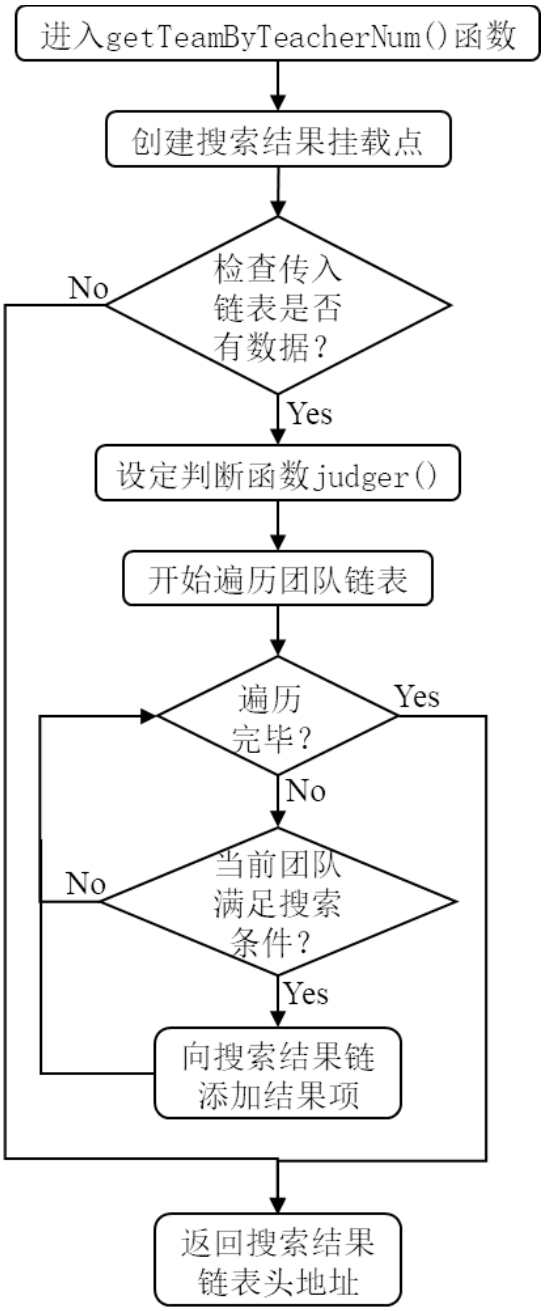

图 4.3 按教师人数查找团队流程图

注：这里由于查询方向比较多（大于、小于、等于等等），故设定了一个判断函数 judger()以减少代码量，其本质为预置好的 5 个判断函数中一个的拷贝。其他数据查询函数由于查询方向单一，不需要设定判断函数这一过程。

4. 数据统计

数据统计实现方式在数据查询的基础上，多出了向包装结构体写入统计数据和对搜索结果链表进行排序的过程。这里只给出统计院系项目平均经费功能的详细流程。

- 第一步：统计所有院系的相关信息，包括项目总数与科研总经费。

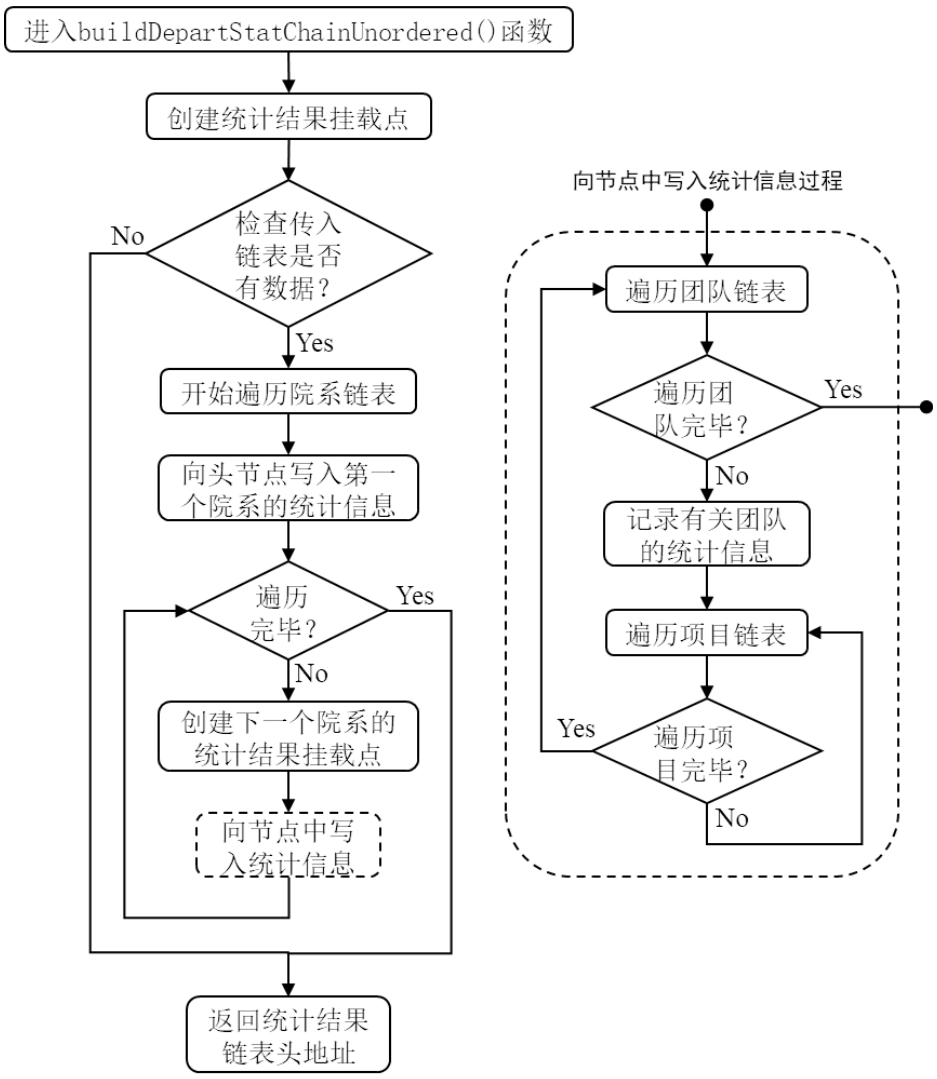

图 4.4.1 建立所有院系统计信息表流程图

注：若只是统计项目总数和科研经费，则可以不需要遍历团队链表，直接遍历项目链表即可。但是考虑到其他统计功能可能需要用到有关团队的统计信息，为了减少可能的代码量，在编写函数的时候还是遍历了团队链表。

- 第二步：对上一步中得到的院系统计结果链按科研总经费与项目总数的比值降序排序。这里采用冒泡排序，交换节点的时候交换院系统计节点指向的院系与相应的统计信息（depart 指针与 stat 数据项）

五、 系统实现

现在展示程序所有部分的具体实现。

1. 源程序文件结构

```
./PROGRAM_ROOT
    +-- utils/
    |   +-- data/                   // 测试用数据文件夹
    |   |   +-- DEPART.DAT
    |   |   +-- TEAM.DAT
    |   |   +-- PROJECT.DAT
    |   +-- __init__.h              // 所有数据处理函数的申明
    |   +-- data_structure.h        // 数据结构
    |   +-- faculty_functions.h     // 院系函数
    |   +-- faculty_functions.c
    |   +-- team_functions.h        // 团队函数
    |   +-- team_functions.c
    |   +-- project_functions.h     // 项目函数
    |   +-- project_functions.c
    |   +-- io_functions.h          // 数据载入、存储函数
    |   +-- io_functions.c
    |   +-- makefile                // 单元测试用
    +-- doc_strings.h               // 用户帮助提示
    +-- views.h                     // 视图函数申明
    +-- main.c                      // 编译起点
    +-- makefile                    // 完整编译用
    +-- py-extension/               // 更新后前端
```

2. 源程序清单

（见附录 I）

六、 运行测试与结果分析（命令行版）

1. 数据维护

a） 数据添加

- 在“Electron”团队（物理学院）下添加编号为“123321”的项目。所有记录的列表中将多出“123321”一项。

```
Computer
    Rocket
        123456
        123234
    MilkyWay
        123345
Physics
    Electron
        123567
Chemistry

Operations Available:
     0) Quit
     1) List out all records
     2) Query utilities
     3) Statistics utilities
     4) Add a new record

> 4
Add object:
     0) Go back
     1) Department
     2) Team
     3) Project

add > 3
add/project::id > 123321
add/project::type
    1 - 973
    2 - NSFC
    3 - 863
    4 - International
    5 - Transverse
> 2
add/project::start_date > 1970/01
add/project::funding > 12.3
add/project::manager > M8
add/project::team > Electron
Operations Available:
     0) Quit
     1) List out all records
     2) Query utilities
     3) Statistics utilities
     4) Add a new record

> 1
Computer
    Rocket
        123456
        123234
    MilkyWay
        123345
Physics
    Electron
        123567
        123321
Chemistry
```

图 6.1.a.1 数据添加功能展示

- 若输入了错误的团队名称，则会出现提示。

```
add/project::team > Elect
Team named Elect not found, do you mean Electron instead?
```

图 6.1.a.2 数据添加错误处理展示

b） 数据修改

- 修改项目“123321”的经费为 2.3 万元。项目的经费将从 1.2 万元变成 2.3 万元。

```
project/123321 > 1
        ATTR          VALUE
      ----------    ----------
      ID            123321
      Type          NSFC
      StartDate     1970/01
      Funding       1.200000 *10000 CNY
      Manager       MS
      Team          Electron

Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) Focus on parent team

project/123321 > 2
Modify Attribute:
     0) Go back
     1) Type
     2) Funding
     3) Manager

project/123321 > 2
project/123321::funding > 2.3
Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) Focus on parent team

project/123321 > 1
        ATTR          VALUE
      ----------    ----------
      ID            123321
      Type          NSFC
      StartDate     1970/01
      Funding       2.300000 *10000 CNY
      Manager       MS
      Team          Electron
```

图 6.1.b 数据修改功能展示

c） 数据删除

- 删除项目“123321”。所有记录的列表中将不再出现名为“123321”的项目。

```
Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) Focus on parent team

project/123321 > 3
Operations Available:
     0) Quit
     1) List out all records
     2) Query utilities
     3) Statistics utilities
     4) Add a new record

> 1
Computer
    Rocket
        123456
        123234
    MilkyWay
        123345
Physics
    Electron
        123567
Chemistry
```

图 6.1.c.1 数据删除功能展示

● 删除非底层节点时，系统会阻止用户删除含有子节点的节点。

```
team/Electron > 3
The team you are deleting has affiliated projects!
Failed to remove record!
Operations Available:
        0) Unfocus
        1) List info
        2) Modify
        3) Delete
        4) List projects
        5) Focus on parent department

team/Electron >
```

图 6.1.c.2 数据删除保险展示

2. 数据查询

鉴于报告篇幅考虑，这里只展示两种数据查询方式，欢迎尝试其他 3 种查询方式 :-)

a） 按名称的全部或部分查询院系

```
Operations Available:
        0) Quit
        1) List out all records
        2) Query utilities
        3) Statistics utilities
        4) Add a new record

> 2
Query Objects:
        0) Go back
        1) Departments
        2) Teams
        3) Projects

query > 1
Query Methods:
        0) Go back
        1) By name
        2) By manager

query/depart > 1
query/depart::name > Comp
            Name          |      Manager      |      Telephone
      ------------------- | ----------------- | -----------------
    1   Computer          |   Zhang3          |   12233334444

Set cursor to department: 1
Operations Available:
        0) Unfocus
        1) List info
        2) Modify
        3) Delete
        4) List teams

depart/Computer >
```

图 6.2.a 按名称查询院系功能展示

b） 按教师人数查询团队

```
query > 2
Query Methods:
     0) Go back
     1) By name
     2) By teacher number

query/team > 2
query/team::condition > <= 5
       Name          |    Manager     |    Faculty
  ---------------------------------------------------
   1  Rocket          |  Hanzo         |  Computer
   2  MilkyWay        |  Genji         |  Computer
   3  Electron        |  L8            |  Physics

Set cursor to team: 2
Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) List projects
     5) Focus on parent department

team/MilkyWay > 1
       ATTR          |    VALUE
  ---------------------------------
       Name          |  MilkyWay
       Manager       |  Genji
       Faculty       |  Computer
       TeacherNum    |  4
       StudentNum    |  4
```

图 6.2.b 按教师人数查询团队功能展示

c） 按所属团队查询项目

- 在前面的例子中已经定位到“MilkyWay”团队，现在导航至其项目“123345”。

```
Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) List projects
     5) Focus on parent department

team/MilkyWay > 4
       ID            |    Manager     |    Type
  ---------------------------------------------------
   1  123345          |  Zhao6         |  NSFC

Set cursor to project: 1
Operations Available:
     0) Unfocus
     1) List info
     2) Modify
     3) Delete
     4) Focus on parent team

project/123345 >
```

图 6.2.c 从团队导航至项目功能展示

注：各层节点均有向上层、向下层导航操作可选，这使得用户查询数据的方式非常灵活。

3. 数据统计

与上面的展示相同，这里只展示 1 种数据统计功能。同样地，欢迎尝试其他 4 种 :-)

a） 院系人数统计

```
Statistics Tools Available:
    0) Go back
    1) Human Resource
    2) Project Overview
    3) Top 10 NSFC Teams
    4) Top 5 Project/Teacher Ratio Teams
    5) Top 5 Average Project Funding Departments

stat > 1
       Name          Teachers      Students      S/T Ratio
  1  Physics          5             6             1.20
  2  Computer         6             7             1.17
  3  Chemistry        0             0             ——
```

图 6.3.a 院系人数统计功能展示

注：第三项统计结果避免了除数为零可能导致的问题。

七、 总结

本次课设应该是第二个自己一个人实现的比较大的项目了。第一个项目是用 Python-Flask 框架实现的一个网页应用（https://github.com/smdsbz/HR-build，已迁移），所以先入为主的面向对象和动态语言的设计哲学对本次课设多少有些影响。可以看到在代码实现中有许多重复，甚至完全相同的部分，仅仅只是因为对象类型不同就必须重新写一个函数。说实话其实现在写完了也没有很习惯用 C 编写带有对象概念的应用，开学了一定要借鉴一下同学的代码学习学习。

此外，由于之前一直都用 Python，被它方便的包引用功能带坏了，也没有用过任何 IDE。在一开始就用了多文件结构，并没有考虑到如何管理工作空间、如何编译等等问题，等到第二个模块写完了要测试的时候才意识到这是个大问题，而之前的课程内容中并没太多与多文件编译相关的内容。于是上网搜索了很多资料，中间也跟“多次 include 函数申明”、“typedef 能不能用 extern 申明”、“Windows 下怎么 make”之类的问题纠缠了很久，姑且是了解了怎么写 makefile，能够把整个项目跑起来了，不过也不知道自己的解决方案是不是正确规范的。

开始编写数据统计模块时已经是 8 月 16 号了，由于本人时间观念有点过强，写这个模块的时候有点赶，一心只想着赶紧写完（事实是只花了不到两天就收尾了）。本着“够用就行”的想法写代码，诞生了我认为是整个项目中第二失败的设计——三个 StatWrapper 结构体。把所有统计数据全部塞到内存里面去，这样做虽然减少了代码编写量（排序和输出的时候不用每个对象都计算一遍），但是确实是很耗内存，但最重要的一点是这样的设计可以说基本没有任何扩展性——如果来了一份新的 PRD，定义了一项新的功能，先不说可能带来的内存消耗，你需要在所有相关的统计函数中逐一添加处理新需求的代码。虽然要加的地方只有两个，但是看到越来越庞大的 buildXxxStatWrapperUnordered()函数，可以预见当数据和需求有相当的量的时候，这将是拖垮整个系统的设计。

一个活生生的例子就是整个项目第一失败的设计——buildTeamStatWrapperUnordered()函数的第三个参数 NSFC_flag。添加这个参数是因为我之前漏看了数据统计功能需求中“某年度”的要求，后来写这篇报告的时候才发现要分年度统计。回头再看 build 函数的时候才无奈地发现整个数据统计模块已经被写死了。于是只好再加一个参数，并且还重用了

本来用于统计所有科研项目总经费的 funding 数据项，进一步限制了统计功能的扩展性（比如又有一个新需求是统计 NSFC 项目经费在总经费中占比，那就又要添加一个数据统计项了）。

这里还要另外提一下用户界面的事情。PRD 中要求的是“至少要有类似于课程设计书上的图形界面”，我翻了一下 conio.h 的官方文档，再看了下课设书的代码，立马就意识到了这可能会是一项十分复杂的工程，想要按时完成项目估计不大可能。而且 MFC 貌似是一个比较杂乱的库，更不用说写出来的程序无法直接“一次编写，到处编译”，移植到*nix 平台上。于是就先写了现在这个实现了类似于应用内导航的字符界面。虽然不合要求，但是至少能用。希望能用剩下来的几天研究一下 GTK，写一个真正的界面出来。

（Update：Win 下 GTK 部署失败，*nix 中有部分基础函数参数列表与 Win 不同）

（Sep. 1st Update：现在已经有基于网页的图形界面了）

最后谈一下自己对整个课设编写过程的感想。

我写这个课设的时间，由于各种其它活动、项目，实际上十分地分散（如图）。

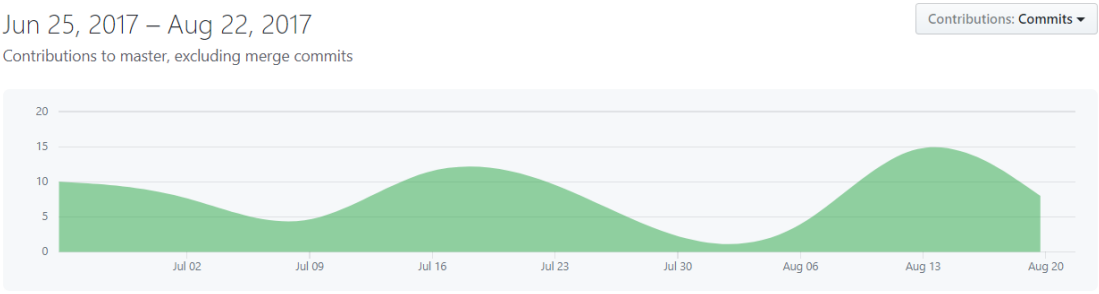

图 7 项目进度提交频率统计（图片来自 GitHub）

所以中间经常性地有被突然打断两三天，然后又要马上回忆起之前的项目进度、所有设计细节的情况出现。第一次被打断然后重新捡起来的时候很困难（中间隔了一周），然后就知道了在头文件注释和行内注释上下功夫的必要。多亏了这些的注释，我暂停两周后回来继续写的时候才能有如此高的效率（图中 8 月 12 号开始的最后一个提交高峰）。

最后的最后我还是想自己夸一句：我觉得这个项目中链表的设计十分赏心悦目 :-)

9 月 1 日更新

在 8 月 22 日完成报告的第一版后，我就回去继续学习 Python 了，无意间看到了一章讲如何编写 Python 的 C 扩展，突发奇想——如果自己能够学会这一章的话，就可以用之前学过的 Python-Flask 框架给课设写一个网页前端来充当图形界面，摆脱命令行交互模式了。于是经过了三天的努力，写代码、查文档（书上的内容很浅而且只支持到 Python2，因此必须去 Python 官网查 API）、debug，终于有了现在的界面。

由于一心想要坚持本次课设的初衷，所有涉及数据的操作都是由 C 扩展完成，Python 层只是负责前端渲染，没有指针、对象的概念，所以操作肯定没有第一版命令行交互要灵活，不过应该也已经符合 PRD 要求了。

由于更新的前端涉及到了其他语言（Python、HTML、JS、CSS 等），此报告书中不会展示具体设计以及代码实现。有兴趣可以查看该部分在 GitHub 上托管的代码（https://github.com/smdsbz/curriculum-design-1/tree/master/py-extension）或者咨询本人。

八、 参考文献

李开主编. C 语言实验与课程设计. 科学出版社, 2011.3.

曹计昌主编. C 语言与程序设计. 电子工业出版社, 2013.1.

Wesley J. Chun. Core Python Applications Programming (3rd Edition). Prentice Hall, 2012.5.

Python Software Foundation, Python/C Reference Manual (Python 3.6.2), 2017.9.

Materialize, Documentation (materializecss.com), 2017.8.

九、 附录

I. 源程序清单

注：由于 Office 软件原因，空格不等宽。可以访问该课设项目在 GitHub 上的 repo（）以获取更好的阅读体验。

./makefile

```
#   $PROGRAM_ROOT/makefile
#   编译所有文件
#   USAGE:
#       Win (with MinGW):   PS > mingw32-make
#                           PS > mingw32-make run
#       *nix:   $ make && ./a.exe ./utils/data

main:
    gcc -o a.exe ./utils/*?.c ./*?.c

# `chcp 65001`为Win命令，调整控制台活动页为UTF-8
run:
    chcp 65001
    ./a.exe .\\utils\\data

clean:
    rm a.exe
```

./main.c

```
/*  $PROG_ROOT/main.c
 *  主函数及编译起始点，这里定义了所有的视图函数
 *  NOTE: 所有视图函数均不接受参数，也没有返回值
 */

#include <stdio.h>
#include <stdlib.h>

#include "./utils/__init__.h"    // 数据处理相关
#include "./views.h"             // 视图函数的申明
#include "./doc_strings.h"       // 用户导航帮助说明

// 初始化环境全局变量
MountPoint mp;                // 头节点挂载点组
char TGT_PATH[100] = "./";    // 数据文件夹路径buffer
Cursor cursor = {0, NULL};    // 用户指针


/********** Data **********/

    /**** Listing ****/

/* 列出所有节点 */
void listAllNodes(void) {
    // 判断是否有数据，防止跨界访问
    if (mp.depart_head == NULL) {
        puts("No data!");
        return;
    }
```

```
    Depart *d; Team *t; Project *p;
    // 遍历院系
    for (d = mp.depart_head; d != NULL; d = d->next) {
        printf("%s\n", d->data->name);
        if (d->child_team_head) {   // 判断该院系是否有团队
            // 遍历团队
            for (t = d->child_team_head;
                    t != d->child_team_tail->next;
                    t = t->next) {
                printf("    %s\n", t->data->name);
                if (t->child_project_head) {    // 同上
                    // 遍历项目
                    for (p = t->child_project_head;
                            p != t->child_project_tail->next;
                            p = p->next) {
                        printf("        %s\n", p->data->id);
                    }
                }
            }
        }
    }
    putchar('\n');
}

/* 列出院系搜索结果中所有内容 */
void listDepartWrapper(DepartWrapper *head) {
    // 传入参数检查
    if (head == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    if (head->depart == NULL) {
        puts("No record found!");
        return;
    }
    // 输出表头
    puts("\
        Name            |    Manager    |    Telephone\n\
     -------------------|---------------|---------------");
    int num = 1;    // 计数器
    // 遍历搜索结果
    for (; head; head = head->next, ++num) {
        printf("%4d  %-18s|  %-13s|  %s\n",
               num, head->depart->data->name,
               head->depart->data->manager,
               head->depart->data->mobile);
    }
    putchar('\n');
}

/* 列出院系人数统计结果 */
void listDepartHRStat(void) {
    // 获取所有院系的统计结果
    DepartStatWrapper *rst = \
        buildDepartStatChainUnordered(mp.depart_head, NULL, 0);
    if (rst == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    // 将统计结果按学生/教师人数比排序
    orderDepartStatWrapperBySTRatio(rst);
    puts("\
        Name            |    Teachers   |    Students   |    S/T Ratio\n\
     -------------------|---------------|---------------|---------------");
    DepartStatWrapper *head = rst; int counter = 1;
```

```
    for (; head; head = head->next, ++counter) {
        printf("%4d  %-18s|  %-13d|  %-13d|  ",
               counter, head->depart->data->name, head->stat.teacher_num,
               head->stat.student_num);
        if (head->stat.teacher_num) {   // 除数可能是零
            printf("%.2f\n", head->stat.st_ratio);
        } else { puts("---"); }
    }
    // 释放统计结果所占用的内存空间
    cleanupDepartStatWrapper(rst);
    putchar('\n');
}

/* 列出院系项目统计结果 */
void listDepartProjectStat(void) {
    // 获取目标查询年份
    printf("stat/project_overview::year (0 for all) > ");
    int year; scanf("%d", &year);
    // 下面的步骤于上一个函数大致相同，略去说明
    DepartStatWrapper *rst = \
        buildDepartStatChainUnordered(mp.depart_head, NULL, year);
    if (rst == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    orderDepartStatWrapperByProjectTotal(rst);
    puts("\
        Name            |    Projects   |    973 Proj.  |    863 Proj.  |
Funding\n\

------------------|---------------|---------------|---------------|--------
-------");
    DepartStatWrapper *head = rst; int counter = 1;
    for (; head; head = head->next, ++counter) {
        printf("%4d  %-18s|  %-13d|  %-13d|  %-13d|  %.2f\n",
               counter, head->depart->data->name,
               head->stat.project_total, head->stat.project_973,
               head->stat.project_863, head->stat.funding);
    }
    cleanupDepartStatWrapper(rst);
    putchar('\n');
}

/* 列出院系项目平均经费统计结果 */
void listDepartFundingStat(void) {
    DepartStatWrapper *rst = \
        buildDepartStatChainUnordered(mp.depart_head, NULL, 0);
    if (rst == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    orderDepartStatWrapperByAvgFunding(rst);
    puts("\
        Name            |    Projects   |    Funding    |    Fund/Proj.\n\
     ------------------|---------------|---------------|---------------");
    DepartStatWrapper *head = rst; int counter = 1;
    for (; head && counter <= 5; head = head->next, ++counter) {
        printf("%4d  %-18s|  %-13d|  %-13.2f|  ",
               counter, head->depart->data->name,
               head->stat.project_total,
               head->stat.funding);
        if (head->stat.project_total) {     // 除数可能为零
            printf("%.2f\n", head->stat.avg_funding);
        } else {
            puts("---");
        }
```

```
    }
    cleanupDepartStatWrapper(rst);
    putchar('\n');
}

/* 列出团队搜索结果中所有内容 */
void listTeamWrapper(TeamWrapper *head) {
    if (head == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    if (head->team == NULL) {
        puts("No record found!");
        return;
    }
    puts("\
        Name             |    Manager    |    Faculty\n\
     -------------------|---------------|---------------");
    int num = 1;
    for (; head; head = head->next, ++num) {
        printf("%4d  %-18s|  %-13s|  %s\n",
               num, head->team->data->name,
               head->team->data->manager,
               head->team->data->faculty);
    }
    putchar('\n');
}

/* 列出NSFC项目最多的10个团队 */
void listTeamNSFCProjectStat(void) {
    TeamStatWrapper *rst = \
        buildTeamStatChainUnordered(mp.team_head, NULL, 1);
    if (rst == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    orderTeamStatWrapperByNSFCProject(rst);
    puts("\
        Name             |    NSFC Proj. |    Funding\n\
     -------------------|---------------|---------------");
    TeamStatWrapper *head = rst; int counter = 1;
    // 较之前的遍历过程，多了列举项目数量的限制
    for (; head && counter <= 10; head = head->next, ++counter) {
        printf("%4d  %-18s|  %-13d|  %.2f\n",
               counter, head->team->data->name,
               head->stat.project_NSFC,
               head->stat.funding);
    }
    cleanupTeamStatWrapper(rst);
    putchar('\n');
}

/* 列出项目数/教师比值前5团队 */
void listTeamProjectStat(void) {
    TeamStatWrapper *rst = \
        buildTeamStatChainUnordered(mp.team_head, NULL, 0);
    if (rst == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    orderTeamStatWrapperByPTRatio(rst);
    puts("\
        Name             |    Teachers   |    Projects   |    P/T Ratio\n\
     -------------------|---------------|---------------|---------------");
    TeamStatWrapper *head = rst; int counter = 1;
    for (; head && counter <= 5; head = head->next, ++counter) {
```

```
        printf("%4d  %-18s|  %-13d|  %-13d|  ",
               counter, head->team->data->name,
               head->team->data->teacher_num,
               head->stat.project_total);
        if (head->team->data->teacher_num) {
            printf("%.2f\n", head->stat.pt_ratio);
        } else { puts("---"); }
    }
    cleanupTeamStatWrapper(rst);
    putchar('\n');
}

/* 列出项目搜索结果的全部内容 */
void listProjectWrapper(ProjectWrapper *head) {
    if (head == NULL) {
        puts("RUNTIME ERROR!");
        exit(-1);
    }
    if (head->project == NULL) {
        puts("No record found!");
        return;
    }
    puts("\
        ID              |    Manager    |    Type\n\
     -------------------|---------------|---------------");
    int num = 1;
    char type_str[40];
    for (; head; head = head->next, ++num) {
        printf("%4d  %-18s|  %-13s|  %s\n",
               num, head->project->data->id,
               head->project->data->manager,
               parseTypeCodeToString(type_str,
                                     head->project->data->type));
    }
    putchar('\n');
}

/* 将数据中存储的项目类型代码，转换为对应类别的名称 */
char *parseTypeCodeToString(char *to, const char from) {
    switch (from) {
        case '1': { strcpy(to, "973 Program"); break; }
        case '2': { strcpy(to, "NSFC"); break; }
        case '3': { strcpy(to, "863 Program"); break; }
        case '4': { strcpy(to, "International Cooperation"); break; }
        case '5': { strcpy(to, "Transverse"); break; }
        default: { strcpy(to, "---"); break; }
    }
    return to;
}

    /**** Querying ****/

        /** Selectors **/

/* 查询对象选择器 */
void selectQueryObjects(void) {
    while (1) {
        // 显示帮助说明
        puts(DOC_QUERY_OBJ);
        int oper_code = 0;  // 记录用户选择的操作
        printf("query > "); scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: {   // 查询院系
                selectQueryDepartMethod();
                return;
            }
```

```
            case 2: {   // 查询团队
                selectQueryTeamMethod();
                return;
            }
            case 3: {   // 查询项目
                // NOTE: 根据团队查找项目整合到团队节点操作中去
                //       故此处直接进入按 ID 查找项目的函数
                queryProjectById();
                return;
            }
            case 0: { return; }
            default: { break; } // 输入无效代码则要求用户重新输入
        }
    }   // input loop
}

/* 查询院系方法选择器 */
void selectQueryDepartMethod(void) {
    while (1) {
        puts(DOC_QUERY_DEPART_METHOD);
        int oper_code = 0;
        printf("query/depart > "); scanf("%d", &oper_code);
        switch(oper_code) {
            case 1: {   // 按名称
                queryDepartByName();
                return;
            }
            case 2: {   // 按负责人
                queryDepartByManager();
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }   // input loop
}

/* 查询团队方法选择器 */
void selectQueryTeamMethod(void) {
    while (1) {
        puts(DOC_QUERY_TEAM_METHOD);
        int oper_code = 0;
        printf("query/team > "); scanf("%d", &oper_code);
        switch(oper_code) {
            case 1: {   // 名称
                queryTeamByName();
                return;
            }
            case 2: {   // 教师数量
                queryTeamByTeacherNum();
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }   // input loop
}

        /** Process **/

/* 按名称查找院系 */
void queryDepartByName(void) {
    // 用户输入目标院系名称
    char depart_name[20];
    printf("query/depart::name > "); scanf("%s", depart_name);
```

```
    // 获取查询结果
    DepartWrapper *depart_wrapper = \
        getDepartByName(mp.depart_head, NULL, depart_name);
    // 列出查询结果
    listDepartWrapper(depart_wrapper);
    // ADD: 用户可以从搜索结果中选择院系，从而进行下一步针对某个院系的操作
    if (depart_wrapper->depart != NULL) {
        int oper_code;
        printf("Set cursor to department: "); scanf("%d", &oper_code);
        DepartWrapper *cur = depart_wrapper;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        // 用户可以通过输入无效值来退出该步骤
        // NOTE: 输入负数仍会将用户指针指向第一个搜索结果
        //       （反正不会溢出，懒得改了。。
        if (cur == NULL || oper_code == 0) {
            if (cur == NULL) { puts("Out of range!"); }
            cleanupDepartWrapper(depart_wrapper);
            return;
        }
        // 给用户指针赋值
        cursor.type = 1; cursor.val = (void *)cur->depart;
    }
    cleanupDepartWrapper(depart_wrapper);
}

/* 按负责人查找院系 */
void queryDepartByManager(void) {
    char depart_manager[20];
    printf("query/depart::manager > "); scanf("%s", depart_manager);
    DepartWrapper *depart_wrapper = \
        getDepartByManager(mp.depart_head, NULL, depart_manager);
    listDepartWrapper(depart_wrapper);
    if (depart_wrapper->depart != NULL) {
        int oper_code;
        printf("Set cursor to department: "); scanf("%d", &oper_code);
        DepartWrapper *cur = depart_wrapper;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        if (cur == NULL || oper_code == 0) {
            if (cur == NULL) { puts("Out of range!"); }
            cleanupDepartWrapper(depart_wrapper);
            return;
        }
        cursor.type = 1; cursor.val = (void *)cur->depart;
    }
    cleanupDepartWrapper(depart_wrapper);
}

/* 按名称查找团队 */
void queryTeamByName(void) {
    char team_name[30];
    printf("query/team::name > "); scanf("%s", team_name);
    TeamWrapper *team_wrapper = \
        getTeamByName(mp.team_head, NULL, team_name);
    listTeamWrapper(team_wrapper);
    if (team_wrapper->team != NULL) {
        int oper_code;
        printf("Set cursor to team: "); scanf("%d", &oper_code);
        TeamWrapper *cur = team_wrapper;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        if (cur == NULL || oper_code == 0) {
            if (cur == NULL) { puts("Out of range!"); }
            cleanupTeamWrapper(team_wrapper);
            return;
        }
        cursor.type = 2; cursor.val = (void *)cur->team;
```

```
    }
    cleanupTeamWrapper(team_wrapper);
}

/* 按教师人数查找团队 */
void queryTeamByTeacherNum(void) {
    Where cond;
    printf("query/team::condition > "); scanf("%s %d",
           &(cond.direction), &(cond.value));
    // NOTE: 查询方向 direction 的错误输入处理交给查询函数处理
    TeamWrapper *team_wrapper = \
        getTeamByTeacherNum(mp.team_head, NULL, cond);
    listTeamWrapper(team_wrapper);
    if (team_wrapper->team != NULL) {
        int oper_code;
        printf("Set cursor to team: "); scanf("%d", &oper_code);
        TeamWrapper *cur = team_wrapper;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        if (cur == NULL || oper_code == 0) {
            puts("Out of range!");
            cleanupTeamWrapper(team_wrapper);
            return;
        }
        cursor.type = 2; cursor.val = (void *)cur->team;
    }
    cleanupTeamWrapper(team_wrapper);
}

/* 按编号查找项目 */
void queryProjectById(void) {
    char project_id[15];
    printf("query/project::ID > "); scanf("%s", project_id);
    ProjectWrapper *project_wrapper = \
        getProjectById(mp.project_head, NULL, project_id);
    // NOTE: 该过程为精确查找，只可能有一个结果
    //       故直接定向用户指针
    if (project_wrapper->project != NULL) {
        cursor.type = 3; cursor.val = (void *)project_wrapper->project;
    }
    cleanupProjectWrapper(project_wrapper);
}

    /**** Adding ****/

/* 添加记录类别选择器 */
void selectAddObjectType(void) {
    while (1) {
        puts(DOC_ADD_TYPE);
        int oper_code = 0;
        printf("add > "); scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: {   // 院系
                appendDepart(mp.depart_head,
                             initDepartData());
                return;
            }
            case 2: {   // 团队
                appendTeam(mp.team_head,
                           initTeamData(),
                           mp.depart_head);
                return;
            }
            case 3: {   // 项目
                appendProject(mp.project_head,
                              initProjectData(),
```

```
                                mp.team_head);
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }
}

/**** Endpoint ****/

    /** Routing **/

/* 院系操作选择器 */
void selectDepartOperation(void) {
    while (1) {
        puts(DOC_DEPART);
        int oper_code = 0;
        printf("depart/%s > ", ((Depart *)cursor.val)->data->name);
        scanf("%d", &oper_code);
        switch (oper_code) {
            // 列出所有属性
            case 1: { listDepartAttr(); break; }
            // 选择要修改的属性
            case 2: { selectDepartModifyAttr(); break; }
            case 3: {   // 删除节点
                if (removeDepart(&(mp.depart_head),
                                 (Depart *)cursor.val)) {   // 成功删除
                    // 用户指针置空
                    cursor.type = 0; cursor.val = NULL;
                    return;
                  // 错误提示
              } else { puts("Failed to remove record!"); break; }
            }
            case 4: { listDepartChildTeam(); return; }
            case 0: {
                cursor.type = 0; cursor.val = NULL;
                return;
            }
            default: { break; }
        }
    }
}

/* 团队操作选择器 */
void selectTeamOperation(void) {
    while (1) {
        puts(DOC_TEAM);
        int oper_code = 0;
        printf("team/%s > ", ((Team *)cursor.val)->data->name);
        scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: { listTeamAttr(); break; }
            case 2: { selectTeamModifyAttr(); break; }
            case 3: {
                if (removeTeam(&(mp.team_head),
                               (Team *)cursor.val)) {
                    cursor.type = 0; cursor.val = NULL;
                    return;
                } else { puts("Failed to remove record!"); break; }
            }
            case 4: { listTeamChildProject(); return; }
            case 5: {   // 重定向至所属院系
                cursor.type = 1;
                cursor.val = ((Team *)cursor.val)->parent_depart;
```

```
                return;
            }
            case 0: {
                cursor.type = 0; cursor.val = NULL;
                return;
            }
            default: { break; }
        }
    }
}

/* 项目操作选择器 */
void selectProjectOperation(void) {
    while (1) {
        puts(DOC_PROJECT);
        int oper_code = 0;
        printf("project/%s > ", ((Project *)cursor.val)->data->id);
        scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: { listProjectAttr(); break; }
            case 2: { selectProjectModifyAttr(); break; }
            case 3: {
                if (removeProject(&(mp.project_head),
                                  (Project *)cursor.val)) {
                    cursor.type = 0; cursor.val = NULL;
                    return;
                } else { puts("Failed to remove record!"); break; }
            }
            case 4: {   // 重定向至所属团队
                cursor.type = 2;
                cursor.val = ((Project *)cursor.val)->parent_team;
                return;
            }
            case 0: {
                cursor.type = 0; cursor.val = NULL;
                return;
            }
            default: { break; }
        }
    }
}

    /** Listing info **/

/* 列出当前指向院系的属性 */
void listDepartAttr(void) {
    Depart *tgt = (Depart *)cursor.val;
    puts("\
        ATTR     |    VALUE\n\
     ------------|-------------------");
    printf("\
      Name       |  %s\n", tgt->data->name);
    printf("\
      Manager    |  %s\n", tgt->data->manager);
    printf("\
      Telephone  |  %s\n", tgt->data->mobile);
    putchar('\n');
}

/* 列出当前指向团队的属性 */
void listTeamAttr(void) {
    Team *tgt = (Team *)cursor.val;
    puts("\
        ATTR     |    VALUE\n\
     ------------|-------------------");
    printf("\
```

```
      Name        |  %s\n", tgt->data->name);
    printf("\
      Manager     |  %s\n", tgt->data->manager);
    printf("\
      Faculty     |  %s\n", tgt->data->faculty);
    printf("\
      TeacherNum  |  %d\n", tgt->data->teacher_num);
    printf("\
      StudentNum  |  %d\n", tgt->data->student_num);
    putchar('\n');
}

/* 列出当前指向项目的属性 */
void listProjectAttr(void) {
    Project *tgt = (Project *)cursor.val;
    puts("\
        ATTR      |    VALUE\n\
     ------------|-------------------");
    printf("\
      ID          |  %s\n", tgt->data->id);
    char type_str[30] = {'\0'};
    printf("\
      Type        |  %s\n",
          parseTypeCodeToString(type_str, tgt->data->type));
    printf("\
      StartDate   |  %s\n", tgt->data->start_date);
    printf("\
      Funding     |  %f *10000 CNY\n", tgt->data->funding);
    printf("\
      Manager     |  %s\n", tgt->data->manager);
    printf("\
      Team        |  %s\n", tgt->data->team);
    putchar('\n');
}

    /** Modifying **/

/* 修改院系属性 */
void selectDepartModifyAttr(void) {
    Depart *tgt = (Depart *)cursor.val;
    while (1) {
        puts(DOC_DEPART_MODIFY);
        int oper_code = 0;
        printf("depart/%s > ", tgt->data->name); scanf("%d", &oper_code);
        // NOTE: 属性修改函数限定了用户能够修改的内容
        switch (oper_code) {
            case 1: {   // 负责人
                printf("depart/%s::manager > ", tgt->data->name);
                scanf("%s", tgt->data->manager);
                return;
            }
            case 2: {   // 联系电话
                printf("depart/%s::telephone > ", tgt->data->name);
                scanf("%s", tgt->data->mobile);
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }
}

/* 修改团队属性 */
void selectTeamModifyAttr(void) {
    Team *tgt = (Team *)cursor.val;
    while (1) {
```

```
        puts(DOC_TEAM_MODIFY);
        int oper_code = 0;
        printf("team/%s > ", tgt->data->name); scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: {   // 负责人
                printf("team/%s::manager > ", tgt->data->name);
                scanf("%s", tgt->data->manager);
                return;
            }
            case 2: {   // 教师人数
                printf("team/%s::teacher_num > ", tgt->data->name);
                scanf("%d", &(tgt->data->teacher_num));
                return;
            }
            case 3: {   // 学生人数
                printf("team/%s::student_num > ", tgt->data->name);
                scanf("%d", &(tgt->data->student_num));
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }
}

/* 修改项目属性 */
void selectProjectModifyAttr(void) {
    Project *tgt = (Project *)cursor.val;
    while (1) {
        puts(DOC_PROJECT_MODIFY);
        int oper_code = 0;
        printf("project/%s > ", tgt->data->id); scanf("%d", &oper_code);
        switch (oper_code) {
            case 1: {
                printf("project/%s::type > ", tgt->data->id);
                do { scanf("%c", &(tgt->data->type)); }
                while (tgt->data->type < '1' || tgt->data->type > '5');
                return;
            }
            case 2: {
                printf("project/%s::funding > ", tgt->data->id);
                scanf("%f", &(tgt->data->funding));
                return;
            }
            case 3: {
                printf("project/%s::manager > ", tgt->data->id);
                scanf("%s", tgt->data->manager);
                return;
            }
            case 0: { return; }
            default: { break; }
        }
    }
}

    /** Listing Child **/

/* 列出院系下属团队 */
void listDepartChildTeam(void) {
    Depart *tgt = (Depart *)cursor.val;
    TeamWrapper *child_team = getTeamByDepart(tgt);
    listTeamWrapper(child_team);
    // ADD: 用户可以直接导航到某一个下属团队
    if (child_team->team != NULL) {
        int oper_code;
        printf("Set cursor to team: "); scanf("%d", &oper_code);
```

```
        TeamWrapper *cur = child_team;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        if (cur == NULL || oper_code == 0) {
            puts("Out of range!");
            cleanupTeamWrapper(child_team);
            return;
        }
        cursor.type = 2; cursor.val = (void *)cur->team;
    }
    cleanupTeamWrapper(child_team);
}

/* 列出团队开设的项目 */
void listTeamChildProject(void) {
    Team *tgt = (Team *)cursor.val;
    ProjectWrapper *child_project = getProjectByTeam(tgt);
    listProjectWrapper(child_project);
    if (child_project->project != NULL) {
        int oper_code;
        printf("Set cursor to project: "); scanf("%d", &oper_code);
        ProjectWrapper *cur = child_project;
        for (; oper_code > 1 && cur; cur = cur->next, --oper_code) ;
        if (cur == NULL || oper_code == 0) {
            puts("Out of range!");
            cleanupProjectWrapper(child_project);
            return;
        }
        cursor.type = 3; cursor.val = (void *)cur->project;
    }
    cleanupProjectWrapper(child_project);
}

/******** Stat ********/

/* 数据统计功能选择器 */
void selectStatItem(void) {
    while (1) {
        puts(DOC_STAT);
        int oper_code = 0;
        printf("stat > "); scanf("%d", &oper_code);
        switch (oper_code) {
            // 院系人数统计
            case 1: { listDepartHRStat(); return; }
            // 院系项目统计
            case 2: { listDepartProjectStat(); return; }
            // 团队国家自然科学基金项目统计
            case 3: { listTeamNSFCProjectStat(); return; }
            // 团队项目/教师比统计
            case 4: { listTeamProjectStat(); return; }
            // 院系项目平均经费统计
            case 5: { listDepartFundingStat(); return; }
            case 0: { return; }
            default: { break; }
        }
    }
}

/*********  Main Bone **********/

/* 顶层（用户指针置空时）用户导航逻辑以及系统初始化与退出 */
int main(int argc, char const *argv[]) {

    // 欢迎屏幕
    puts(START);
```

```
// 从命令行参数读取目标数据文件夹路径
if (argc == 1) {
    // 用户未指定路径
    // 使用默认值，即程序根目录
    puts("Data folder unspecified!");
    puts("Finding data files in program root...");
} else if (argc == 2) {
    // 用户指定了路径
    strcpy(TGT_PATH, argv[1]);
    printf("Loading data from folder \"%s\"...\n", TGT_PATH);
} else { puts("Too many arguments!"); exit(0); }

// 加载数据
mp = loadData(TGT_PATH);
// 检查数据完整性
if (mp.depart_head == NULL
        || mp.team_head == NULL
        || mp.project_head == NULL) {
    // 数据加载时出现错误
    //     无数据文件，数据损坏等等
    puts("Data file not found!");
    // 清理现有的含有错误数据的链表
    cleanupDepart(mp.depart_head);
    cleanupTeam(mp.team_head);
    cleanupProject(mp.project_head);
    // 询问用户是否在当前文件夹创建数据文件
    printf("Create data file in %s ? [Y/N]: ", TGT_PATH);
    char option = 0;
    scanf("%c", &option);
    switch (option) {
        case 'Y':   case 'y': {
            // NOTE: 此时数据文件并不会立即创建
            //       由于fwrite()函数的特性，数据文件在保存时自动创建
            mp.depart_head = createDepartHead();
            mp.team_head = createTeamHead();
            mp.project_head = createProjectHead();
            printf("Data will be saved to %s !\n", TGT_PATH);
            break;
        }
        case 'N':   case 'n':   default: {
            // 没有数据文件，退出程序
            puts("No data available!");
            puts("exit");
            return 0;
        }
    }
}

// 成功载入数据
puts("Successfullly linked all data!"); putchar('\n');

// 用户导航
while (1) {
    switch (cursor.type) {  // 根据用户指针指向判断用户所在层级
        case 0: {   // 顶层（用户指针置空）
            puts(DOC_ROOT);
            int oper_code = 0;
            printf("> "); scanf("%d", &oper_code);
            switch (oper_code) {
                // 列出所有记录
                case 1: { listAllNodes(); break; }
                // 数据查询
                case 2: { selectQueryObjects(); break; }
                // 数据统计
```

```
                    case 3: { selectStatItem(); break; }
                    // 新增记录
                    case 4: { selectAddObjectType(); break; }
                    case 0: {
                        // 退出程序时自动保存数据
                        // 数据有效性检查
                        if (!(mp.depart_head->data
                              && mp.team_head->data
                              && mp.project_head->data)) {
                            puts("Basic info not complete!");
                            printf("Quit without saving? [Y/N] ");
                            char option;
                            do {    // 循环吸收之前残余的'\n'
                                scanf("%c", &option);
                                switch (option) {
                                    case 'y':   case 'Y': {
                                        puts("exit"); return 0;
                                    }
                                    default: { break; }
                                }
                            } while (option != 'n' && option != 'N');
                            break;
                        }
                        // NOTE: 没有单独的数据保存选项
                        if (saveData(mp, TGT_PATH) == 0) {  // 存盘操作失败
                            puts("Error while saving...");
                            printf("Does the directory %s ever exist?\n",
                                   TGT_PATH);
                            puts("GG! MY FRIEND! (  ¯ y∇, ¯ )┌ ");
                            exit(-1);
                        }                           // 释放内存
                        cleanupDepart(mp.depart_head);
                        cleanupTeam(mp.team_head);
                        cleanupProject(mp.project_head);
                        // 正确退出
                        puts("exit");
                        return 0;
                    }
                    default: { break; }
                }   // inner switch
                break;
            }
            case 1: { selectDepartOperation(); break; }     // 院系层
            case 2: { selectTeamOperation(); break; }       // 团队层
            case 3: { selectProjectOperation(); break; }    // 项目层
            // 指向错误的层级 - 直接退出，不做内存清理
            default: { printf("ERR > exit"); return -1; }
        }
    }
}
```

./doc_strings.h

```
/*  $PROGRAM_ROOT/doc_strings.h
 *  用户界面字符串
 */

#ifndef _DOC_STRING_H
#define _DOC_STRING_H

// 欢迎界面
char *START = "============================================================\n\
  \\ +---------+  +---------+**+-------+**\\ +---------+\n\
  *\\|  +------+  |  +---+  |\\*|        \\**\\|  +------+\n\
```

```
  **|  |   \\**\\--|  |-\\ |  |*\\|  ++  +  |**|  |\n\
  --|  +------+--|  +---+  |**|  ||  |  |--|  +------+\n\
    +------+  |\\ |  +------+**|  ||  |  |  +------+  |\n\
  --------\\|  |*\\|  |      \\**|  ||  |  |--------\\|  |\n\
  \\*+------+  |\\*|  |       \\*|  ||  |  |\\*+------+  |\n\
   \\+---------+ \\+--+        \\+--++--+--+ \\+---------+    by smdsbz\n\
============================================================\n\
The Science Project Managing System\n\
Version: 0.1\n\
Author: smdsbz@GitHub.com (i.e. U201610136 朱晓光)\n\
Enjoy your ride!\n\
";

// 顶层
char *DOC_ROOT = "Operations Available:\n\
     0) Quit\n\
     1) List out all records\n\
     2) Query utilities\n\
     3) Statistics utilities\n\
     4) Add a new record\n\
";

// 查询对象选择
char *DOC_QUERY_OBJ = "Query Objects:\n\
     0) Go back\n\
     1) Departments\n\
     2) Teams\n\
     3) Projects\n\
";

// 查询院系方法
char *DOC_QUERY_DEPART_METHOD = "Query Methods:\n\
     0) Go back\n\
     1) By name\n\
     2) By manager\n\
";

// 查询团队方法
char *DOC_QUERY_TEAM_METHOD = "Query Methods:\n\
     0) Go back\n\
     1) By name\n\
     2) By teacher number\n\
";

// 添加记录类型选择
char *DOC_ADD_TYPE = "Add object:\n\
     0) Go back\n\
     1) Department\n\
     2) Team\n\
     3) Project\n\
";

// 院系操作
char *DOC_DEPART = "Operations Available:\n\
     0) Unfocus\n\
     1) List info\n\
     2) Modify\n\
     3) Delete\n\
     4) List teams\n\
";

// 选择修改院系属性
char *DOC_DEPART_MODIFY = "Modify Attribute:\n\
     0) Go back\n\
     1) Manager\n\
     2) Telephone\n\
```

```
";

// 团队操作
char *DOC_TEAM = "Operations Available:\n\
     0) Unfocus\n\
     1) List info\n\
     2) Modify\n\
     3) Delete\n\
     4) List projects\n\
     5) Focus on parent department\n\
";

// 选择修改团队属性
char *DOC_TEAM_MODIFY = "Modify Attribute:\n\
     0) Go back\n\
     1) Manager\n\
     2) Teacher number\n\
     3) Student number\n\
";

// 项目操作
char *DOC_PROJECT = "Operations Available:\n\
     0) Unfocus\n\
     1) List info\n\
     2) Modify\n\
     3) Delete\n\
     4) Focus on parent team\n\
";

// 选择修改项目属性
char *DOC_PROJECT_MODIFY = "Modify Attribute:\n\
     0) Go back\n\
     1) Type\n\
     2) Funding\n\
     3) Manager\n\
";

// 选择统计
char *DOC_STAT = "Statistics Tools Available:\n\
     0) Go back\n\
     1) Human Resource\n\
     2) Project Overview\n\
     3) Top 10 NSFC Teams\n\
     4) Top 5 Project/Teacher Ratio Teams\n\
     5) Top 5 Average Project Funding Departments\n\
";

#endif
```

./utils/views.h

```
/*  $PROGRAM_ROOT/views.h
 *  所有视图函数与用户指针结构体声明
 */

#ifndef _VIEWS_H
#define _VIEWS_H

/** 顶层 **/
/* 列出所有节点 */
extern void listAllNodes(void);
/* 列出院系搜索结果中所有内容 */
extern void listDepartWrapper(DepartWrapper *);
/* 列出团队搜索结果中所有内容 */
extern void listTeamWrapper(TeamWrapper *);
```

```
/* 列出项目搜索结果的全部内容 */
extern void listProjectWrapper(ProjectWrapper *);
/* 将数据中存储的项目类型代码，转换为对应类别的名称 */
extern char *parseTypeCodeToString(char *, const char);

/**** 查询 ****/
/* 查询对象选择器 */
extern void selectQueryObjects(void);

/****** 查询方法选择 ******/
/* 查询院系方法选择器 */
extern void selectQueryDepartMethod(void);
/* 查询团队方法选择器 */
extern void selectQueryTeamMethod(void);

/******** 查询过程 ********/
/* 按名称查找院系 */
extern void queryDepartByName(void);
/* 按负责人查找院系 */
extern void queryDepartByManager(void);
/* 按名称查找团队 */
extern void queryTeamByName(void);
/* 按教师人数查找团队 */
extern void queryTeamByTeacherNum(void);
/* 按编号查找项目 */
extern void queryProjectById(void);

/**** 添加 ****/
/* 添加记录类别选择器 */
extern void selectAddObjectType(void);

/**** 统计 ****/
/* 数据统计功能选择器 */
extern void selectStatObject(void);
/* 列出院系人数统计结果 */
extern void listDepartHRStat(void);
/* 列出院系项目统计结果 */
extern void listDepartProjectStat(void);
/* 列出 NSFC 项目最多的 10 个团队 */
extern void listTeamNSFCProjectStat(void);
/* 列出项目数/教师比值前 5 团队 */
extern void listTeamProjectStat(void);
/* 列出院系项目平均经费统计结果 */
extern void listDepartFundingStat(void);

/*************************************************/

/** 院系层 **/
/* 院系操作选择器 */
extern void selectDepartOperation(void);
/* 列出当前指向院系的属性 */
extern void listDepartAttr(void);
/* 修改院系属性 */
extern void selectDepartModifyAttr(void);
extern void listDepartChildTeam(void);

/** 团队层 **/
/* 团队操作选择器 */
extern void selectTeamOperation(void);
/* 列出当前指向团队的属性 */
extern void listTeamAttr(void);
/* 修改团队属性 */
```

```
extern void selectTeamModifyAttr(void);
/* 列出团队开设的项目 */
extern void listTeamChildProject(void);

/** 项目层 **/
/* 项目操作选择器 */
extern void selectProjectOperation(void);
/* 列出当前指向项目的属性 */
extern void listProjectAttr(void);
/* 修改项目属性 */
extern void selectProjectModifyAttr(void);

/*****************************************************/

/** 用户指针结构体定义 **/
typedef struct _Cursor {
    int     type;   // 指向对象的类型
                    // 0-置空 1-院系 2-团队 3-项目
    void    *val;   // 指针值
} Cursor;

#endif
```

./utils/data_structure.h

```
/*  $PROJ_HOME/utils/data_structure.c
 *  所有的数据结构的定义都在这里
 */

#ifndef DATA_STRUCTURE
#define DATA_STRUCTURE  "data_structure.h"

#ifdef BUILDING
#undef BUILDING
#endif
// 正在开发标识 - 解注释启用单元测试
// #define BUILDING

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/******** Data ********/

    /**** Depart ****/

/* 院系数据 */
typedef struct _DepartData {
    // self.data
    char    name[20];       // 院系名称
    char    manager[12];    // 负责人
    char    mobile[15];     // 负责人电话
} DepartData;

/* 院系节点 */
typedef struct _Depart {
    struct _DepartData  *data;          // 数据域
    struct _Depart  *next;              // 下一个院系节点
    struct _Team    *child_team_head;   // 所有团队中的第一个
    struct _Team    *child_team_tail;   // '''''''中的最后一个
} Depart;

/* 院系搜索结果包装 */
typedef struct _DepartWrapper {
```

```
    struct _Depart          *depart;     // 符合条件的院系节点
    struct _DepartWrapper   *next;       // 下一个结果
} DepartWrapper;

/* 院系统计数据 */
typedef struct _DepartStatData {
    int     student_num;    // 学生总数
    int     teacher_num;    // 教师总数
    float   st_ratio;       // 学生/教师比例
    int     project_total;  // 项目总数
    int     project_973;    // 973项目数
    int     project_863;    // 863项目数
    float   funding;        // 科研总经费
    float   avg_funding;    // 项目平均经费
} DepartStatData;

/* 院系统计结果包装 */
typedef struct _DepartStatWrapper {
    struct _Depart              *depart;     // 源数据节点
    struct _DepartStatData      stat;        // 统计数据
    struct _DepartStatWrapper   *next;       // 下一个节点
} DepartStatWrapper;

    /**** Team ****/

/* 团队数据 */
typedef struct _TeamData {
    char    name[30];       // 团队名称
    char    manager[12];    // 负责人
    int     teacher_num;    // 教师人数
    int     student_num;    // 研究生人数
    char    faculty[20];    // 所属院系
} TeamData;

/* 团队节点 */
typedef struct _Team {
    struct _TeamData    *data;
    struct _Depart  *parent_depart; // 所属院系
    struct _Team    *next;
    struct _Project *child_project_head;
    struct _Project *child_project_tail;
} Team;

/* 团队搜索结果包装 */
typedef struct _TeamWrapper {
    struct _Team        *team;
    struct _TeamWrapper *next;
} TeamWrapper;

/* 团队统计数据 */
typedef struct _TeamStatData {
    int     project_total;  // 项目总数
    int     project_NSFC;   // NSFC项目数
    float   funding;        // 总经费
    float   pt_ratio;       // 项目数/教师人数
} TeamStatData;

/* 团队统计结果包装 */
typedef struct _TeamStatWrapper {
    struct _Team                *team;
    struct _TeamStatData        stat;
    struct _TeamStatWrapper     *next;
```

```
} TeamStatWrapper;

    /**** Project ****/

/* 项目数据 */
typedef struct _ProjectData {
    char    id[15];         // 项目编号
    char    type;           // 项目类别
    char    start_date[8];  // 起始时间
    float   funding;        // 项目经费
    char    manager[12];    // 项目负责人
    char    team[30];       // 所属团队
} ProjectData;

/* 项目节点 */
typedef struct _Project {
    struct _ProjectData *data;
    struct _Team        *parent_team;
    struct _Project     *next;
} Project;

/* 项目搜索结果包装 */
typedef struct _ProjectWrapper {
    struct _Project         *project;
    struct _ProjectWrapper  *next;
} ProjectWrapper;

/******** Search Condition ********/
/* 搜索条件 - 用于按教师人数查询团队功能 */
typedef struct _Where {
    char    direction[3];   // 查找条件 - ^[\>\<]=?|=$
    int     value;
} Where;

/******** Mounting Point Group ********/
/* 头节点挂载点组 - 用于返回一组指针 */
typedef struct _MountPoint {
    struct _Depart      *depart_head;   // 院系链表挂载点
    struct _Team        *team_head;     // 团队链表挂载点
    struct _Project     *project_head;  // 项目链表挂载点
} MountPoint;

/********** Unit Test **************/

#if defined(BUILDING)
void main(void) {
    puts("Successfully constructed!");
}
#endif

#endif
```

./utils/faculty_functions.h

```
/*  $PROGRAM_ROOT/utils/faculty_functions.h
 *  院系方法头文件
 */

#ifndef FACULTY_FUNCTIONS
#define FACULTY_FUNCTIONS

/************ Declaration ************/

    /**** POST | DELETE | PUT ****/
```

```
extern Depart *appendDepart(Depart *, DepartData);
/*  添加院系数据
 *  ARGS:   链表头，院系数据模板
 *  RETN:   新增节点的地址
 */

extern int modifyDepart(Depart *, DepartData);
/*  覆盖院系信息
 *  ARGS:   目标地址，已经修改数据的buffer
 *  RETN:   success code
 *  NOTE:   没有调用！（clang可能报错）
 */

extern int removeDepart(Depart **, Depart *);
/*  删除院系节点
 *  ARGS:   指向院系链表头节点地址的指针，目标地址 | NULL
 *  RETN:   success code
 */

extern DepartData initDepartData(void);
/*  创建院系数据模板
 *  ARGS:   void
 *  RETN:   根据在该函数执行过程中输入的数据所创建出来的模板
 *  NOTE:   包含用户输入前端
 */

extern Depart *createDepartHead(void);
/*  创建并初始化头节点
 *  ARGS:   void
 *  RETN:   头节点地址 || NULL
 *  NOTE:   头节点此时没有数据域
 */

    /**** SELECT ****/

// NOTE: 会申请DepartWrapper空间，记得调用cleanupDepartWrapper()释放内存

extern DepartWrapper *
getDepartByManager(Depart *, Depart *, const char *);
/*  通过负责人姓名查找院系
 *  ARGS:   院系链表，搜索结束点，院系负责人 char[12]
 *  RETN:   搜索结果挂载点 || NULL （没有结果时也返回挂载点地址）
 */

extern DepartWrapper *
getDepartByName(Depart *, Depart *, const char *);
/*  通过院系名称查找院系
 *  ARGS:   院系链表，搜索结束点，名称线索（不一定是全称）
 *  RETN:   搜索结果挂载点 || NULL
 */

    /**** STAT ****/

extern DepartStatWrapper *
buildDepartStatChainUnordered(Depart *, Depart *, int);
/*  统计所有院系数据
 *  ARGS:   院系链表头，遍历区域边界，查询目标年度
 *  RETN:   统计结果挂载点 || NULL
 */

extern DepartStatWrapper *
orderDepartStatWrapperBySTRatio(DepartStatWrapper *);
```

```
/*  将统计结果按学生/教师人数比排序
 *  ARGS:    统计结果挂载点
 *  RETN:    排序后统计结果链的挂载点
 */

extern DepartStatWrapper *
orderDepartStatWrapperByProjectTotal(DepartStatWrapper *);
/*  将统计结果按项目总数降序排序
 */

extern DepartStatWrapper *
orderDepartStatWrapperByAvgFunding(DepartStatWrapper *);
/*  将统计结果按项目平均经费降序排序
 */

extern Depart *getPrevDepart(Depart *, Depart *);
/*  获得当前院系节点的前一个节点
 *  ARGS:    当前节点，院系链表头
 *  RETN:    前一个节点 || NULL
 *  NOTE:    未调用!
 */

    /**** CLEANUPs ****/

extern void cleanupDepartWrapper(DepartWrapper *);
/*  清除搜索结果链表
 *  ARGS:    搜索结果链表头节点地址
 *  RETN:    void
 *  NOTE:    每次搜完了记得调用!
 *  NOTE:    调用后搜索结果会被完全清空，包括头节点本身
 */

extern void cleanupDepart(Depart *);
/*  释放院系链所占用的内存空间
 *  ARGS:    院系链表挂载点
 *  RETN:    void
 */

extern void cleanupDepartStatWrapper(DepartStatWrapper *);
/*  清除统计结果链表
 *  ARGS:    统计结果链表头节点地址
 *  RETN:    void
 */

#endif
```

./utils/faculty_functions.c

```
/*  $PROGRAM_ROOT/utils/faculty_functions.c
 *  院系方法定义与实现
 */

#ifndef DATA_STRUCTURE
#include "data_structure.h"
#endif

#ifdef BUILDING
#undef BUILDING
#endif
// #define BUILDING

#ifdef DEBUG
#undef DEBUG
#endif
```

```
// 解注释以查看 DEBUG 信息
// #define DEBUG

/************ Declaration ************/
// 函数调用说明见头文件
    /**** POST | DELETE | PUT ****/
Depart *appendDepart(Depart *head, DepartData new_one);
int modifyDepart(Depart *target, DepartData new_one);
int removeDepart(Depart **phead, Depart *target);
DepartData initDepartData(void);
Depart *createDepartHead(void);
    /**** SELECT ****/
DepartWrapper *getDepartByManager(Depart *, Depart *, const char *);
DepartWrapper *getDepartByName(Depart *, Depart *, const char *);
DepartStatWrapper *buildDepartStatChainUnordered(Depart *start, Depart *end,
int);
DepartStatWrapper *orderDepartStatWrapperBySTRatio(DepartStatWrapper
*start);
DepartStatWrapper *orderDepartStatWrapperByProjectTotal(DepartStatWrapper
*start);
DepartStatWrapper *orderDepartStatWrapperByAvgFunding(DepartStatWrapper
*start);
Depart *getPrevDepart(Depart *cur, Depart *head);
    /**** CLEANUPs ****/
void cleanupDepartWrapper(DepartWrapper *start);
void cleanupDepart(Depart *start);

/*********** Unit Test **************/
#if defined(BUILDING)

void printDepartToConsole(Depart *Comp) {
/*  向终端打印院系信息
 *  ARGS:   院系链表头
 *  RETN:   void
 */
    printf("<Department @ 0x%p>\n", Comp);
    printf("\tthis.name = %s\n", Comp->data->name);
    printf("\tthis.manager = %s\n", Comp->data->manager);
    printf("\t\tthis.mobile = %s\n", Comp->data->mobile);
    #if defined(CHILD_COUNTER)
    printf("\tthis.team_num = %d\n", Comp->data->team_num);
    #endif
    printf("\n\tnext_node @ 0x%p\n", Comp->next);
    printf("\tteam from 0x%p to 0x%p\n", Comp->child_team_head, Comp-
>child_team_tail);
    putchar('\n'); putchar('\n');
}

void printDepartWrapperToConsole(DepartWrapper *head) {
/*  向终端打印院系搜索结果
 *  ARGS:   院系搜索结果链表头
 *  RETN:   void
 */
    printf("<DepartWrapper @ 0x%p>\n", head);
    int counter = 0;
    while (head->depart != NULL) {
        ++counter;
        printf("\tfound %s @ 0x%p\n", head->depart->data->name, head-
>depart);
        head = head->next;
        if (head == NULL) { break; }
    }
    if (!counter) { puts("no match!"); }
    putchar('\n'); putchar('\n');
}
```

```
// 单元测试
void main(void) {
    Depart *HEAD = createDepartHead();
    // first node creation demo
    appendDepart(HEAD, initDepartData());
    printDepartToConsole(HEAD);
    // second node creation demo
    appendDepart(HEAD, initDepartData());
    printDepartToConsole(HEAD->next);
    // query demo
    auto char buf[40] = {'\0'};
    printf("search by manager: "); scanf("%s", buf);
    DepartWrapper *RST_LIST = getDepartByManager(HEAD, NULL, buf);
    printDepartWrapperToConsole(RST_LIST);
    cleanupDepartWrapper(RST_LIST);
    // search demo
    printf("search by name: "); scanf("%s", buf);
    RST_LIST = getDepartByName(HEAD, NULL, buf);
    printDepartWrapperToConsole(RST_LIST);
    cleanupDepartWrapper(RST_LIST);
    // remove demo
    puts("removing HEAD->next");
    removeDepart(&HEAD, HEAD->next);
    printDepartToConsole(HEAD);
    // end of test
    free(HEAD->next);
    free(HEAD);
}
#endif

/*************** Function Realizations ***************/

    /**** POST | DELETE | PUT ****/

/* 创建并初始化头节点 */
Depart *createDepartHead(void) {
    Depart *HEAD = (Depart *)malloc(sizeof(Depart));
    if (HEAD == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in createDepartHead():\n\tfailed to malloc for
depart (head)");
        #endif
        return NULL;
    }
    HEAD->data = NULL; HEAD->next = NULL;
    HEAD->child_team_head = NULL; HEAD->child_team_tail = NULL;
    return HEAD;
}

/* 创建院系数据模板 */
DepartData initDepartData(void) {
    DepartData Computer;
    printf("add/depart::name > "); scanf("%s", Computer.name);
    printf("add/depart::manager > "); scanf("%s", Computer.manager);
    printf("add/depart::mobile > "); scanf("%s", Computer.mobile);
    #if defined(CHILD_COUNTER)
    Computer.team_num = 0;
    #endif
    return Computer;
}

/* 添加院系数据 */
Depart *appendDepart(Depart *head, DepartData new_one) {
    // 检查是否已经有同名的院系存在
    DepartWrapper *depart_wrapper = \
        getDepartByName(head, NULL, new_one.name);
```

```
    if (depart_wrapper->depart != NULL) {
        // NOTE: 实际上若已经有 Computer，则 Comp 也会在这里被拦截
        printf("Record for %s department already exists!\n", new_one.name);
        cleanupDepartWrapper(depart_wrapper);
        return NULL;
    }
    cleanupDepartWrapper(depart_wrapper);
    // 获取院系链表尾节点
    Depart *tail = head;
    for (; tail->next; tail = tail->next) ;
    // 开始添加数据
    // case 1 - 院系链表此时还没有任何记录
    if (head == tail && tail->data == NULL) {
        // 只需要修改头节点的数据域
        // NOTE: 此时头节点的数据域还没有内存空间
        tail->data = (DepartData *)malloc(sizeof(DepartData));
        if (tail->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendDepart():\n\tfailed to alloc memory
for data (head)");
            #endif
            return NULL;
        }
        // 院系数据赋值
        *(tail->data) = new_one; tail->next = NULL;
        tail->child_team_head = NULL; tail->child_team_tail = NULL;
        return tail;    // 返回新添加的节点
    }
    // else { case 2 - 院系链表非空 }
    // 在尾部链接新节点
    // 为新节点申请内存空间
    tail->next = (Depart *)malloc(sizeof(Depart));
    if (tail->next == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in appendDepart():\n\tfailed to alloc memory for
container");
        #endif
        return NULL;
    }
    // 为新节点的数据域申请内存空间
    tail->next->data = (DepartData *)malloc(sizeof(DepartData));
    if (tail->next->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in appendDepart():\n\tfailed to alloc memory for
data");
        #endif
        return NULL;
    }
    *(tail->next->data) = new_one; tail->next->next = NULL;
    tail->next->child_team_head = NULL;
    tail->next->child_team_tail = NULL;
    return tail->next;   // 返回新节点地址
}

/* 覆盖院系信息 - 未调用 */
int modifyDepart(Depart *target, DepartData new_one) {
    if (target == NULL) {   // 输入检查
        #if defined(DEBUG)
        puts("[LOG] Error in modifyDepart():\n\ttarget is NULL");
        #endif
        return 0;   // error code
    }
    // 数据覆盖
    *(target->data) = new_one;
```

```
    return 1;
}

/* 删除院系节点 */
int removeDepart(Depart **phead, Depart *tgt) {
    if (tgt == NULL) {   // 输入检查
        #if defined(DEBUG)
        puts("[LOG] Error in removeDepart():\n\ttgt is NULL");
        #endif
        return 0;
    }
    // 检查院系是否有团队，防止团队信息丢失
    if (tgt->child_team_head) {
        puts("The department you are deleting has affiliated teams!");
        return 0;
    }
    // 使链表绕开要删除的节点
    // case 1 - 要删除的节点是头节点且不是唯一结点
    if (*phead == tgt && tgt->next) {
        *phead = tgt->next; // 头节点重新赋值
    } else { // case 2 - 要删除的节点不是头节点（故不可能为唯一节点）
        // NOTE: 若 tgt 是唯一的节点，phead 不能指向 NULL
        Depart *phead_safe = *phead;
        // 遍历找到目标节点之前的节点
        for (; phead_safe->next != tgt; phead_safe = phead_safe->next) ;
        phead_safe->next = tgt->next;
    }
    // 删除节点
    free(tgt->data); // 释放数据域
    // case 3 - 唯一结点，需要保留链表挂载点
    if (tgt->next == NULL && *phead == tgt) { tgt->data = NULL; }
    else { free(tgt); }    // 释放节点本身占用的空间
    return 1;
}

    /**** CLEANUPs ****/

/* 清除搜索结果链表 */
void cleanupDepartWrapper(DepartWrapper *prev) {
    if (prev == NULL) {     // 输入检查
        #if defined(DEBUG)
        puts("[LOG] cleanupDepartWrapper(): got NULL");
        #endif
        return;
    }
    DepartWrapper *after = prev;
    while (1) {
        after = prev->next; // 备份下一个节点位置
        free(prev);         // 释放当前节点
        #if defined(DEBUG)
        printf("[LOG] cleanupDepartWrapper(): freed 0x%p\n", prev);
        #endif
        prev = after;       // 准备删除下一个节点
        if (prev == NULL) { break; }   // 已经全部清空
    }
    return;
    // NOTE: 此时，传进来的头节点也没了
}

/* 清除统计结果链表 */
void cleanupDepartStatWrapper(DepartStatWrapper *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("[LOG] cleanupDepartStatWrapper(): got NULL");
```

```
        #endif
        return;
    }
    DepartStatWrapper *after = prev;
    while (1) {
        after = prev->next;
        free(prev);
        #if defined(DEBUG)
        printf("[LOG] cleanupDepartStatWrapper(): freed 0x%p\n", prev);
        #endif
        prev = after;
        if (prev == NULL) { break; }
    }
    return;
}

/* 释放院系链所占用的内存空间 */
void cleanupDepart(Depart *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("[LOG] cleanupDepart(): got NULL");
        #endif
        return;
    }
    Depart *after = prev;
    while (1) {
        after = prev->next;
        // 这里不光要释放节点本身的内存，还要释放数据域的空间
        free(prev->data); free(prev);
        #if defined(DEBUG)
        printf("[LOG] cleanupDepart(): freed 0x%p\n", prev);
        #endif
        prev = after;
        if (prev == NULL) { break; }
    }
    return;
}

    /**** SELECT ****/

/* 通过负责人姓名查找院系 */
DepartWrapper *
getDepartByManager(Depart *start, Depart *end, const char *manager) {
    // 创建搜索结果挂载点
    DepartWrapper *rtn = (DepartWrapper *)malloc(sizeof(DepartWrapper));
    if (rtn == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getDepartByManager():\n\tfailed to malloc for
result mounting point");
        #endif
        return NULL;
    }
    DepartWrapper *rtn_head = rtn;  // 保存搜索结果链的头指针
    rtn_head->depart = NULL; rtn_head->next = NULL;   // 初始化
    if (start->data == NULL) {  // 检查院系链是否有数据
        #if defined(DEBUG)
        puts("[LOG] getDepartByManager(): searching an empty chain");
        #endif
        // 若无数据，返回没有任何数据的挂载点
        return rtn_head;
    }
    // 遍历查找
    for (; start != end; start = start->next) {
        if (!strcmp(start->data->manager, manager)) {  // BINGO!
            #if defined(BUILDING)
```

```
            printf("[LOG] getDepartByManager(): found %s @ 0x%p\n", start-
>data->name, start);
            #endif
            // 将当前记录添加至搜索结果链表
            // case 1 - 此前没有找到任何满足要求的链表
            if (rtn_head->depart == NULL) {
                // 向挂载点写入数据
                rtn->depart = start;
            } else {    // case 2 - 已经有结果了
                // 为新结果申请空间
                rtn->next = (DepartWrapper *)malloc(sizeof(DepartWrapper));
                if (rtn->next == NULL) {
                    #if defined(DEBUG)
                    puts("[LOG] Error in getDepartByManager():\n\tfailed to
malloc for result container");
                    #endif
                    // 申请空间失败则中断操作，并清空搜索结果，返回 NULL
                    cleanupDepartWrapper(rtn_head);
                    return NULL;
                }
                // 申请成功，注册搜索结果
                rtn = rtn->next; rtn->next = NULL;
                rtn->depart = start;
            }

        }
    }
    return rtn_head;    // 不管有没有找到，都返回搜索结果链表头
}

/* 通过院系名称查找院系 */
// 查询函数实现都极为相似，此后只在必要时给出注释
DepartWrapper *
getDepartByName(Depart *start, Depart *end, const char *name) {
    DepartWrapper *rtn = (DepartWrapper *)malloc(sizeof(DepartWrapper));
    if (rtn == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getDepartByName():\n\tfailed to malloc for
result mounting point");
        #endif
        return NULL;
    }
    DepartWrapper *rtn_head = rtn;
    rtn_head->depart = NULL; rtn_head->next = NULL;
    if (start->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] getDepartByName(): searching an empty chain");
        #endif
        return rtn_head;
    }
    for (; start != end; start = start->next) {
        // 和上一个函数唯一的不同
        if (strstr(start->data->name, name)) {  // 搜索内容是已保存内容的子串
            #if defined(DEBUG)
            printf("[LOG] getDepartByName(): found %s @ 0x%p\n", start-
>data->name, start);
            #endif
            if (rtn_head->depart == NULL) {
                rtn->depart = start;
            } else {
                rtn->next = (DepartWrapper *)malloc(sizeof(DepartWrapper));
                if (rtn->next == NULL) {
                    #if defined(DEBUG)
                    puts("[LOG] Error in getDepartByName():\n\tfailed to
malloc for result container");
```

```
                    #endif
                    cleanupDepartWrapper(rtn_head);
                    return NULL;
                }
                rtn = rtn->next; rtn->next = NULL;
                rtn->depart = start;
            }

        }
    }
    return rtn_head;
}

    /**** Stat ****/

/* 统计所有院系数据 */
DepartStatWrapper *
buildDepartStatChainUnordered(Depart *start, Depart *end, int year) {
    // 创建统计结果链挂载点
    DepartStatWrapper *unordered = \
        (DepartStatWrapper *)malloc(sizeof(DepartStatWrapper));
    if (unordered == NULL) { return NULL; }
    if (start->data == NULL) {  // 检查传入链表是否有数据
        #if defined(DEBUG)
        puts("[LOG] Error in getDepartOrderedByMasterTeacherRatio():
building with an empty chain");
        #endif
        unordered->depart = NULL; unordered->next = NULL;
        return unordered;
    }
    // 开始统计
    Depart *start_bak = start; DepartStatWrapper *unordered_bak = unordered;
    char year_str[10];  // 由于项目创建时间以字符串形式存储，在此设立buffer
    // 创建第一条统计数据
    // NOTE: <do-while>标识的区域内的内容完全相同
    // <do-while>
    // 节点统计数据初始化
    unordered_bak->depart = start_bak; unordered_bak->next = NULL;
    unordered_bak->stat.student_num = 0; unordered_bak->stat.teacher_num =
0;
    unordered_bak->stat.project_total = 0; unordered_bak->stat.project_973 =
0;
    unordered_bak->stat.project_863 = 0; unordered_bak->stat.funding = 0;
    // 遍历团队
    Team *child_team = start_bak->child_team_head;
    if (child_team != NULL) {   // 检查是否有下属团队，防止跨界访问
        for (; child_team != start_bak->child_team_tail->next;
               child_team = child_team->next) {
            // 统计学生、教师人数
            unordered_bak->stat.student_num += child_team->data-
>student_num;
            unordered_bak->stat.teacher_num += child_team->data-
>teacher_num;
            // 遍历项目
            Project *child_project = child_team->child_project_head;
            if (child_project != NULL) {    // 同上，防止跨界访问
                for (; child_project != child_team->child_project_tail-
>next;
                       child_project = child_project->next) {
                    sprintf(year_str, "%d", year);
                    if (year == 0   // 当用户输入的年份为0时，统计所有数据
                            || strstr(child_project->data->start_date,
                                      year_str) != NULL) {
                        // NOTE: 由于记录时间的字符串位数不多，直接判断输入年份是否为
字串即可
```

```
                        unordered_bak->stat.project_total += 1;
                        if (child_project->data->type == '1')
                            { unordered_bak->stat.project_973 += 1; }
                        if (child_project->data->type == '3')
                            { unordered_bak->stat.project_973 += 1; }
                        unordered_bak->stat.funding += child_project->data-
>funding;
                    }
                }
            }
        }
    }
    // 计算学生/教师人数比
    if (unordered_bak->stat.teacher_num == 0)    // 防止零除数
        { unordered_bak->stat.st_ratio = -1; }
    else {
        unordered_bak->stat.st_ratio = \
            (float)unordered_bak->stat.student_num \
            / unordered_bak->stat.teacher_num;
    }
    // 计算项目平均经费
    if (unordered_bak->stat.project_total == 0) // 同上，防止零除数
        { unordered_bak->stat.avg_funding = -1; }
    else {
        unordered_bak->stat.avg_funding = \
            unordered_bak->stat.funding \
            / unordered_bak->stat.project_total;
    }
    // </do-while>
    for (start_bak = start_bak->next; start_bak != end; start_bak =
start_bak->next) {
        unordered_bak->next = (DepartStatWrapper
*)malloc(sizeof(DepartStatWrapper));
        if (unordered_bak->next == NULL) { return NULL; }
        unordered_bak = unordered_bak->next;
        // <do-while>
        unordered_bak->depart = start_bak; unordered_bak->next = NULL;
        unordered_bak->stat.student_num = 0; unordered_bak->stat.teacher_num
= 0;
        unordered_bak->stat.project_total = 0; unordered_bak-
>stat.project_973 = 0;
        unordered_bak->stat.project_863 = 0; unordered_bak->stat.funding =
0;
        Team *child_team = start_bak->child_team_head;
        if (child_team != NULL) {
            for (; child_team != start_bak->child_team_tail->next;
                   child_team = child_team->next) {
                unordered_bak->stat.student_num += child_team->data-
>student_num;
                unordered_bak->stat.teacher_num += child_team->data-
>teacher_num;
                Project *child_project = child_team->child_project_head;
                if (child_project != NULL) {
                    for (; child_project != child_team->child_project_tail-
>next;
                           child_project = child_project->next) {
                        sprintf(year_str, "%d", year);
                        if (year == 0   // 当用户输入的年份为0时，统计所有数据
                                || strstr(child_project->data->start_date,
                                          year_str) != NULL) {
                            unordered_bak->stat.project_total += 1;
                            if (child_project->data->type == '1')
                                { unordered_bak->stat.project_973 += 1; }
                            if (child_project->data->type == '3')
                                { unordered_bak->stat.project_973 += 1; }
                            unordered_bak->stat.funding += child_project-
```

```
>data->funding;
                        }
                    }
                }
            }
        }
        if (unordered_bak->stat.teacher_num == 0)
            { unordered_bak->stat.st_ratio = -1; }
        else {
            unordered_bak->stat.st_ratio = \
                (float)unordered_bak->stat.student_num \
                / unordered_bak->stat.teacher_num;
        }
        if (unordered_bak->stat.project_total == 0)
            { unordered_bak->stat.avg_funding = -1; }
        else {
            unordered_bak->stat.avg_funding = \
                unordered_bak->stat.funding \
                / unordered_bak->stat.project_total;
        }
        // </do-while>
    }
    return unordered;
}
/* 将统计结果按学生/教师人数比排序 */
DepartStatWrapper *
orderDepartStatWrapperBySTRatio(DepartStatWrapper *start) {
    if (start == NULL) { return NULL; } // 输入检查
    DepartStatWrapper *start_bak = start;
    DepartStatWrapper *cur = start;
    Depart *Depart_tmp; DepartStatData DepartStatData_tmp;
    // 冒泡排序
    for (; start_bak->next; start_bak = start_bak->next) {
        for (cur = start; cur->next; cur = cur->next) {
            // 🐓 图个方便，这里就牺牲点效率了
            if (cur->stat.st_ratio < cur->next->stat.st_ratio) {
                // 交换院系
                Depart_tmp = cur->next->depart;
                cur->next->depart = cur->depart;
                cur->depart = Depart_tmp;
                // 交换统计信息
                DepartStatData_tmp = cur->next->stat;
                cur->next->stat = cur->stat;
                cur->stat = DepartStatData_tmp;
            }
        }
    }
    return start;
}

/* 将统计结果按项目总数降序排序 */
DepartStatWrapper *
orderDepartStatWrapperByProjectTotal(DepartStatWrapper *start) {
    if (start == NULL) { return NULL; }
    DepartStatWrapper *start_bak = start;
    DepartStatWrapper *cur = start;
    Depart *Depart_tmp; DepartStatData DepartStatData_tmp;
    for (; start_bak->next; start_bak = start_bak->next) {
        for (cur = start; cur->next; cur = cur->next) {
            if (cur->stat.project_total < cur->next->stat.project_total) {
                // swap depart
                Depart_tmp = cur->next->depart;
                cur->next->depart = cur->depart;
                cur->depart = Depart_tmp;
                // swap stat
                DepartStatData_tmp = cur->next->stat;
```

```
                cur->next->stat = cur->stat;
                cur->stat = DepartStatData_tmp;
            }
        }
    }
    return start;
}

/* 将统计结果按项目平均经费降序排序 */
DepartStatWrapper *
orderDepartStatWrapperByAvgFunding(DepartStatWrapper *start) {
    if (start == NULL) { return NULL; }
    DepartStatWrapper *start_bak = start;
    DepartStatWrapper *cur = start;
    Depart *Depart_tmp; DepartStatData DepartStatData_tmp;
    for (; start_bak->next; start_bak = start_bak->next) {
        for (cur = start; cur->next; cur = cur->next) {
            if (((float)cur->stat.funding / cur->stat.project_total) <
((float)cur->next->stat.funding / cur->next->stat.project_total)) {
                // swap depart
                Depart_tmp = cur->next->depart;
                cur->next->depart = cur->depart;
                cur->depart = Depart_tmp;
                // swap stat
                DepartStatData_tmp = cur->next->stat;
                cur->next->stat = cur->stat;
                cur->stat = DepartStatData_tmp;
            }
        }
    }
    return start;
}

/* 获得当前院系节点的前一个节点 - 未调用*/
Depart *getPrevDepart(Depart *cur, Depart *head) {
    for (; head->next != cur && head != NULL;
           head = head->next) ;
    return head;
}

#ifdef BUILDING
#undef BUILDING
#endif

#ifdef DEBUG
#undef DEBUG
#endif
```

./utils/team_functions.h

```
/*  $PROGRAM_ROOT/utils/team_functions.h
 *  团队方法头文件
 */

#ifndef TEAM_FUNCTIONS
#define TEAM_FUNCTIONS

    /**** POST | DELETE | PUT ****/

extern TeamData initTeamData(void);
/*  创建团队数据模板
 *  ARGS:   void
 *  RETN:   根据在该函数执行过程中输入的数据所创建出来的原型
 *  NOTE:   用户输入前端
 */
```

```
extern Team *appendTeam(Team *, TeamData, Depart *);
/*  添加团队数据
 *  ARGS:   链表头，团队数据模板，母结点链（院系链）
 *  RETN:   新增节点的地址
 */

extern int modifyTeam(Team *, TeamData);
/*  覆盖团队信息
 *  ARGS:   目标地址，已经修改数据的buffer
 *  RETN:   success code
 */

extern int removeTeam(Team **, Team *);
/*  删除团队节点
 *  ARGS:   指向团队链表头节点地址的指针，目标地址 | NULL
 *  RETN:   success code
 */

extern Team *createTeamHead(void);
/*  创建并初始化头节点
 *  ARGS:   void
 *  RETN:   头节点地址 || NULL
 */

    /**** SELECT ****/

extern TeamWrapper *
getTeamByTeacherNum(Team *, Team *, const Where);
/*  通过教师数量查找团队
 *  ARGS:   团队链表，搜索结束点，查找条件
 *  RETN:   搜索结果挂载点 | NULL
 */

extern TeamWrapper *getTeamByName(Team *, Team *, const char *);
/*  通过团队名称查找团队
 *  ARGS:   团队链表，搜索结束点，团队名称线索（不一定是全称）
 *  RETN:   搜索结果挂载点 | NULL
 */

extern TeamWrapper *getTeamByDepart(Depart *);
/*  通过院系查找团队
 *  ARGS:   目标院系节点
 *  RETN:   搜索结果（该院系下所有团队）挂载点 || NULL
 */

    /**** STAT ****/

extern TeamStatWrapper *
buildTeamStatChainUnordered(Team *, Team *, char);
/*  统计所有团队数据
 *  ARGS:   团队链表头，遍历区域边界，查询NSFC项目flag
 *  RETN:   统计结果挂载点
 */

extern TeamStatWrapper *
orderTeamStatWrapperByNSFCProject(TeamStatWrapper *);
/*  将统计结果按NSFC项目数降序排序
 *  ARGS:   统计结果挂载点
 *  RETN:   排序后统计结果链的挂载点
 */

extern TeamStatWrapper *
orderTeamStatWrapperByPTRatio(TeamStatWrapper *);
```

```
/*  将统计结果按项目/教师比降序排序
 */

    /**** CLEANUPs ****/

extern void cleanupTeamWrapper(TeamWrapper *);
/*  清空搜索结果序列
 *  ARGS:   头节点地址
 *  RETN:   void
 */

extern void cleanupTeamStatWrapper(TeamStatWrapper *);
/*  清除统计结果链表
 *  ARGS:   统计结果链表头节点地址
 *  RETN:   void
 */

extern void cleanupTeam(Team *);
/*  释放 Team 链所占用的内存空间
 *  ARGS:   头节点地址
 *  RETN:   void
 */

#endif
```

./utils/team_functions.c

```
/*  $PROGRAM_ROOT/utils/team_functions.c
 *  团队方法定义与实现
 */

#ifndef DATA_STRUCTURE
#include "data_structure.h"
#endif

#ifndef FACULTY_FUNCTIONS
#include "faculty_functions.h"
#endif

#ifdef BUILDING
#undef BUILDING
#endif
// #define BUILDING

#ifdef DEBUG
#undef DEBUG
#endif
// #define DEBUG

/* 教师人数条件判断器 */
int isSmaller(int valve, int data) { return (data < valve); }
int isLarger(int valve, int data) { return (data > valve); }
int isEqualTo(int valve, int data) { return (data == valve); }
int noLargerThan(int valve, int data) { return (data <= valve); }
int noSmallerThan(int valve, int data) { return (data >= valve); }

int (*setJudger(const char *direction))(int, int) {
/*  教师人数条件判断选择器
 *  ARGS:   代表搜索规则的字符串(eg "<=")
 *  RETN:   int judger(int valve, int data)
 *          实际为上面申明的函数中的一种
 */
    if (!strcmp("<", direction)) { return isSmaller; }
    if (!strcmp(">", direction)) { return isLarger; }
    if (!strcmp("=", direction)) { return isEqualTo; }
    if (!strcmp("<=", direction)) { return noLargerThan; }
```

```
    if (!strcmp(">=", direction)) { return noSmallerThan; }
    else { return NULL; }
}

/*********** Declaration ***********/
    /**** POST | DELETE | PUT ****/
TeamData initTeamData(void);
Team *appendTeam(Team *, TeamData, Depart *);
int modifyTeam(Team *, TeamData);
int removeTeam(Team **, Team *);
Team *createTeamHead(void);
    /**** SELECT ****/
TeamWrapper *getTeamByTeacherNum(Team *, Team *, const Where);
TeamWrapper *getTeamByName(Team *, Team *, const char *);
TeamStatWrapper *buildTeamStatChainUnordered(Team *, Team *, char);
TeamStatWrapper *orderTeamStatWrapperByNSFCProject(TeamStatWrapper *);
TeamStatWrapper *orderTeamStatWrapperByPTRatio(TeamStatWrapper *);
    /**** CLEANUPs ****/
void cleanupTeamWrapper(TeamWrapper *);
void cleanupTeam(Team *);
void cleanupTeamStatWrapper(TeamStatWrapper *);

/************ Function Realizations *************/
// NOTE: 大部分实现与院系方法实现相同，下面只在必要处给出注释

    /**** POST | DELETE | PUT ****/

/* 创建并初始化头节点 */
Team *createTeamHead(void) {
    Team *Team_HEAD = (Team *)malloc(sizeof(Team));
    if (Team_HEAD == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in createTeamHead():\n\tfailed to malloc for team
(head)");
        #endif
        return NULL;
    }
    Team_HEAD->data = NULL; Team_HEAD->next = NULL;
    Team_HEAD->parent_depart = NULL;    // 母节点
    Team_HEAD->child_project_head = NULL;
    Team_HEAD->child_project_tail = NULL;
    return Team_HEAD;
}

/* 创建团队数据模板 */
TeamData initTeamData(void) {
    TeamData VirtusPro;
    printf("add/team::name > "); scanf("%s", VirtusPro.name);
    printf("add/team::manager > "); scanf("%s", VirtusPro.manager);
    printf("add/team::teacher_num > "); scanf("%d",
&(VirtusPro.teacher_num));
    printf("add/team::student_num > "); scanf("%d",
&(VirtusPro.student_num));
    printf("add/team::faculty > "); scanf("%s", VirtusPro.faculty);
    return VirtusPro;
}

/* 添加团队数据 */
Team *appendTeam(Team *head, TeamData new_one, Depart *depart_chain) {
    TeamWrapper *team_wrapper = getTeamByName(head, NULL, new_one.name);
    if (team_wrapper->team != NULL) {
        printf("Record for %s team already exists!\n", new_one.name);
        cleanupTeamWrapper(team_wrapper);
        return NULL;
    }
    cleanupTeamWrapper(team_wrapper);
```

```
    Team *tail = head;
    for (; tail->next; tail = tail->next) ;
    DepartWrapper *parent_depart_wrapper = \
        getDepartByName(depart_chain, NULL, new_one.faculty);
    if (parent_depart_wrapper == NULL) {    // 查找失败
        #if defined(DEBUG)
        puts("[LOG] appendTeam():\n\tgetDepartByName() failed");
        #endif
        return NULL;
    }
    if (parent_depart_wrapper->depart == NULL) {    // 没有符合要求的结果
        puts("Target parent department not found!\n");
        return NULL;
    }
    if (parent_depart_wrapper->next != NULL) {  // 找到了多个符合要求的结果
        puts("Multiple parent departments found!\n");
        return NULL;
    }
    if (strcmp(parent_depart_wrapper->depart->data->name,
               new_one.faculty)) {
        // 由于根据名称查找院系是模糊查询，这里需要做一次校验
        printf("Department named %s not found, do you mean %s instead?\n",
               new_one.faculty, parent_depart_wrapper->depart->data->name);
        return NULL;
    }
    // 此时，已经验证了 parent_depart_wrapper 中只有一个院系
    Depart *parent_depart = parent_depart_wrapper->depart;
    cleanupDepartWrapper(parent_depart_wrapper);
    #if defined(BUILDING)
    printf("[LOG] appendTeam(): parent_depart is %s @ 0x%p\n",
           parent_depart->data->name, parent_depart);
    #endif
    // 添加节点
    //   case 1: 团队链表中目前还没有数据
    if (tail == head && tail->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] appendTeam(): case 1");
        #endif
        tail->data = (TeamData *)malloc(sizeof(TeamData));
        if (tail->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendTeam():\n\tfailed to alloc memory for
data (head)");
            #endif
            return NULL;
        }
        *(tail->data) = new_one;
        tail->next = NULL;
        tail->child_project_head = NULL; tail->child_project_tail = NULL;
        // 子节点链接母节点
        tail->parent_depart = parent_depart;
        // 在母节点中注册子节点
        parent_depart->child_team_head = tail;
        parent_depart->child_team_tail = tail;
        return tail;
    }
    //   case 2 & 3: 需要添加节点的操作
    if (parent_depart->child_team_tail == NULL
            || parent_depart->child_team_tail->next == NULL) {
        #if defined(DEBUG)
        puts("[LOG] appendTeam(): case 2");
        #endif
        // case 2: 母结点在当前还没有子节点
        //         或者 母结点的团队链的尾就是整个团队链的尾节点
        //         --> 向尾节点 tail 后添加节点
```

```
        tail->next = (Team *)malloc(sizeof(Team));
        if (tail->next == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendTeam():\n\tfailed to alloc memory for
container");
            #endif
            return NULL;
        }
        // 给新节点的数据域申请空间
        tail->next->data = (TeamData *)malloc(sizeof(TeamData));
        if (tail->next->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendTeam():\n\tfailed to alloc memory for
data");
            #endif
            return NULL;
        }
        // 下一个节点的指向需要在这里设置
        tail->next->next = NULL;
    } else {    // case 3: 母结点已经有子节点了
        #if defined(DEBUG)
        puts("[LOG] appendTeam(): case 3");
        #endif
        // 将tail指向现有子节点链的尾部
        tail = parent_depart->child_team_tail;
        Team *after = tail->next;   // something or NULL
        tail->next = (Team *)malloc(sizeof(Team));
        if (tail->next == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendTeam():\n\tfailed to alloc memory for
container");
            #endif
            return NULL;
        }
        tail->next->data = (TeamData *)malloc(sizeof(TeamData));
        if (tail->next->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendTeam():\n\tfailed to alloc memory for
data");
            #endif
            return NULL;
        }
        // 下一个节点指向
        tail->next->next = after;
    }
    // 数据域复制
    *(tail->next->data) = new_one;
    // 在母节点中注册
    //   注册指针指向
    //     该团队为母节点的第一个子节点
    if (parent_depart->child_team_head == NULL) {
        parent_depart->child_team_head = tail->next;
    }
    parent_depart->child_team_tail = tail->next;
    // 子节点链接母结点
    tail->next->parent_depart = parent_depart;
    // 下一级节点指向初始化
    tail->next->child_project_head = NULL;
    tail->next->child_project_tail = NULL;
    return tail->next;
}

/* 覆盖团队信息 */
int modifyTeam(Team *tgt, TeamData new_one) {
    if (tgt == NULL) {
```

```
        #if defined(DEBUG)
        puts("[LOG] Error in modifyTeam():\n\ttarget is NULL");
        #endif
        return 0;
    }
    if (strcmp(tgt->data->faculty, new_one.faculty) != 0) {
        #if defined(DEBUG)
        printf("[LOG] Error in modifyTeam():\n\tchanging team faculty,");
        printf(" from %s to %s, strcmp() got %d\n",
               tgt->data->faculty, new_one.faculty,
               strcmp(tgt->data->faculty, new_one.faculty));
        #endif
        return 0;
    }
    *(tgt->data) = new_one;
    return 1;
}

/* 删除团队节点 */
int removeTeam(Team **phead, Team *tgt) {
    if (tgt == NULL) {   // 无效参数
        #if defined(DEBUG)
        puts("[LOG] Error in removeTeam():\n\ttarget is NULL");
        #endif
        return 0;
    }
    if (tgt->child_project_head) {
        // 该组有正在运行的项目，删除会导致项目信息失效，并留下垃圾数据
        #if defined(DEBUG)
        puts("[LOG] Error in removeTeam():\n\tteam has working porjects,
can't move");
        #endif
        puts("The team you are deleting has affiliated projects!");
        return 0;
    }
    Team *prev = *phead;
    for (; prev->next != NULL && prev->next != tgt; prev = prev->next) ;
    // 删除节点
    if (*phead == tgt && tgt->next) {
        *phead = tgt->next;
    } else {
        prev->next = tgt->next;
    }
    // 在母结点数据域注册删除操作
    // 在母结点的指针域注册
    if (tgt->parent_depart->child_team_head == \
         tgt->parent_depart->child_team_tail) {
        // case 1: tgt 是母结点下最后一个子节点，现在母结点没有子节点了
        tgt->parent_depart->child_team_head = NULL;
        tgt->parent_depart->child_team_tail = NULL;
    } else if (tgt->parent_depart->child_team_head == tgt) {
        // case 2: tgt 是母结点子链头节点
        tgt->parent_depart->child_team_head = tgt->next;
    } else if (tgt->parent_depart->child_team_tail == tgt) {
        // case 3: tgt 是母结点子链尾节点
        tgt->parent_depart->child_team_tail = prev;
    }
    #if defined(BUILDING)
    printf("[LOG] removeTeam(): freed %s @ 0x%p\n",
           tgt->data->name, tgt);
    #endif
    free(tgt->data);
    if (tgt->next == NULL && *phead == tgt) { tgt->data = NULL; }
    else { free(tgt); }
    return 1;
```

```
}

    /**** SELECT ****/

/* 通过教师数量查找团队 */
TeamWrapper *getTeamByTeacherNum(Team *start, Team *end, const Where cond) {
    TeamWrapper *rtn = (TeamWrapper *)malloc(sizeof(TeamWrapper));
    if (rtn == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getTeamByTeacherNum():\n\tfailed to malloc for
result mounting point");
        #endif
        return NULL;
    }
    TeamWrapper *rtn_head = rtn;
    rtn_head->team = NULL; rtn_head->next = NULL;
    if (start->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] getTeamByTeacherNum(): searching an empty chain");
        #endif
        return rtn_head;
    }
    // 设定条件判断器
    int (*judger)(int, int);
    judger = setJudger(cond.direction);
    if (judger == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getTeamByTeacherNum():\n\tunrecoginized
condition");
        #endif
        printf("Undefined query direction %s!\n", cond.direction);
        return NULL;
    }
    // 开始查询满足条件的节点
    for (; start != end; start = start->next) {
        if (judger(cond.value, start->data->teacher_num)) {
            #if defined(DEBUG)
            printf("[LOG] getTeamByTeacherNum(): found %s @ 0x%p\n",
                   start->data->name, start);
            #endif
            if (rtn_head->team == NULL) {
                rtn->team = start;
            }
            else {
                rtn->next = (TeamWrapper *)malloc(sizeof(TeamWrapper));
                if (rtn->next == NULL) {
                    #if defined(DEBUG)
                    puts("[LOG] Error in getTeamByTeacherNum():\n\tfailed to
malloc for result container");
                    #endif
                    cleanupTeamWrapper(rtn_head);
                    return NULL;
                }
                rtn = rtn->next; rtn->next = NULL;
                rtn->team = start;
            }
        }
    }
    return rtn_head;
}

/* 通过团队名称查找团队 */
TeamWrapper *getTeamByName(Team *start, Team *end, const char *hint) {
    TeamWrapper *rtn = (TeamWrapper *)malloc(sizeof(TeamWrapper));
    if (rtn == NULL) {
        #if defined(DEBUG)
```

```
        puts("[LOG] Error in getTeamByName():\n\tfailed to malloc for result
mounting point");
        #endif
        return NULL;
    }
    TeamWrapper *rtn_head = rtn;
    rtn_head->team = NULL; rtn_head->next = NULL;
    if (start->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] getTeamByName(): searching an empty chain");
        #endif
        return rtn_head;
    }
    for (; start != end; start = start->next) {
        if (strstr(start->data->name, hint) != NULL) {
            #if defined(BUILDING)
            printf("[LOG] getTeamByName(): found %s @ 0x%p\n",
                   start->data->name, start);
            #endif
            if (rtn_head->team == NULL) {
                rtn->team = start;
            } else {
                rtn->next = (TeamWrapper *)malloc(sizeof(TeamWrapper));
                if (rtn->next == NULL) {
                    #if defined(DEBUG)
                    puts("[LOG] Error in getTeamByName():\n\tfailed to
malloc for result container");
                    #endif
                    cleanupTeamWrapper(rtn_head);
                    return NULL;
                }
                rtn = rtn->next; rtn->next = NULL;
                rtn->team = start;
            }
        }
    }
    return rtn_head;
}

/* 通过院系查找团队 */
TeamWrapper *getTeamByDepart(Depart *parent_depart) {
    TeamWrapper *rtn_head = (TeamWrapper *)malloc(sizeof(TeamWrapper));
    if (rtn_head == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getTeamByDepart():\n\tfailed to malloc for
result maunting point");
        #endif
        return NULL;
    }
    rtn_head->team = NULL; rtn_head->next = NULL;
    if (parent_depart->child_team_head == NULL) {
        return rtn_head;
    }
    Team *cur = parent_depart->child_team_head;
    TeamWrapper *rtn = rtn_head;
    // 遍历获得院系下所有团队
    for (; cur != parent_depart->child_team_tail->next; cur = cur->next) {
        #if defined(DEBUG)
        printf("[LOG] getTeamByDepart(): reached %s @ 0x%p\n",
               cur->data->name, cur);
        #endif
        if (rtn_head->team == NULL) {
            rtn->team = cur;
        } else {
            rtn->next = (TeamWrapper *)malloc(sizeof(TeamWrapper));
            if (rtn->next == NULL) {
                #if defined(DEBUG)
```

```
                puts("[LOG] Error in getTeamByDepart():\n\tfailed to malloc
for result container");
                #endif
                cleanupTeamWrapper(rtn_head);
                return NULL;
            }
            rtn = rtn->next;
            rtn->team = cur; rtn->next = NULL;
        }
    }
    return rtn_head;
}

    /**** Stat ****/

/* 统计所有团队数据 */
TeamStatWrapper *
buildTeamStatChainUnordered(Team *start, Team *end, char NSFC_flag) {
    TeamStatWrapper *unordered = \
        (TeamStatWrapper *)malloc(sizeof(TeamStatWrapper));
    if (unordered == NULL) { return NULL; }
    if (start->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getTeamOrderedByMasterTeacherRatio(): building
with an empty chain");
        #endif
        unordered->team = NULL; unordered->next = NULL;
        return unordered;
    }
    Team *start_bak = start; TeamStatWrapper *unordered_bak = unordered;
    // <do-while>
    unordered_bak->team = start_bak; unordered_bak->next = NULL;
    unordered_bak->stat.project_total = 0;
    unordered_bak->stat.project_NSFC = 0;
    unordered_bak->stat.funding = 0;
    // 统计项目
    Project *child_project = start_bak->child_project_head;
    if (child_project != NULL) {
        for (; child_project != start_bak->child_project_tail->next;
               child_project = child_project->next) {
            unordered_bak->stat.project_total += 1;
            // TODO
            // UGLY: 若NSFC_flag为真，则funding项里只有NSFC经费
            if ((!NSFC_flag) || (NSFC_flag && child_project->data-
>type=='2'))
                { unordered_bak->stat.funding += child_project->data-
>funding; }
            if (child_project->data->type == '2')
                { unordered_bak->stat.project_NSFC += 1; }
        }
    }
    if (unordered_bak->team->data->teacher_num) {
        unordered_bak->stat.pt_ratio = \
            (float)unordered_bak->stat.project_total \
            / unordered_bak->team->data->teacher_num;
    } else { unordered_bak->stat.pt_ratio = -1; }
    // </do-while>
    for (start_bak = start_bak->next;
            start_bak != end; start_bak = start_bak->next) {
        unordered_bak->next = (TeamStatWrapper
*)malloc(sizeof(TeamStatWrapper));
        if (unordered_bak->next == NULL) { return NULL; }
        unordered_bak = unordered_bak->next;
        // <do-while>
        unordered_bak->team = start_bak; unordered_bak->next = NULL;
        unordered_bak->stat.project_total = 0;
```

```
        unordered_bak->stat.project_NSFC = 0;
        unordered_bak->stat.funding = 0;
        Project *child_project = start_bak->child_project_head;
        if (child_project != NULL) {
            for (; child_project != start_bak->child_project_tail->next;
                   child_project = child_project->next) {
                unordered_bak->stat.project_total += 1;
                if ((!NSFC_flag) || (NSFC_flag && child_project->data-
>type=='2'))
                    { unordered_bak->stat.funding += child_project->data-
>funding; }
                if (child_project->data->type == '2')
                    { unordered_bak->stat.project_NSFC += 1; }
            }
        }
        if (unordered_bak->team->data->teacher_num) {
            unordered_bak->stat.pt_ratio = \
                (float)unordered_bak->stat.project_total \
                / unordered_bak->team->data->teacher_num;
        } else { unordered_bak->stat.pt_ratio = -1; }
        // </do-while>
    }
    return unordered;
}

/* 将统计结果按NSFC项目数降序排序 */
TeamStatWrapper *orderTeamStatWrapperByNSFCProject(TeamStatWrapper *start) {
    if (start == NULL) { return NULL; }
    TeamStatWrapper *start_bak = start;
    TeamStatWrapper *cur = start;
    Team *Team_tmp; TeamStatData TeamStatData_tmp;
    for (; start_bak->next; start_bak = start_bak->next) {
        for (cur = start; cur->next; cur = cur->next) {
            if (cur->stat.project_NSFC < cur->next->stat.project_NSFC) {
                // swap team
                Team_tmp = cur->next->team;
                cur->next->team = cur->team;
                cur->team = Team_tmp;
                // swap stat
                TeamStatData_tmp = cur->next->stat;
                cur->next->stat = cur->stat;
                cur->stat = TeamStatData_tmp;
            }
        }
    }
    return start;
}

/* 将统计结果按项目/教师比降序排序 */
TeamStatWrapper *orderTeamStatWrapperByPTRatio(TeamStatWrapper *start) {
    if (start == NULL) { return NULL; }
    TeamStatWrapper *start_bak = start;
    TeamStatWrapper *cur = start;
    Team *Team_tmp; TeamStatData TeamStatData_tmp;
    for (; start_bak->next; start_bak = start_bak->next) {
        for (cur = start; cur->next; cur = cur->next) {
            if (cur->stat.pt_ratio < cur->next->stat.pt_ratio) {
                // swap team
                Team_tmp = cur->next->team;
                cur->next->team = cur->team;
                cur->team = Team_tmp;
                // swap stat
                TeamStatData_tmp = cur->next->stat;
                cur->next->stat = cur->stat;
                cur->stat = TeamStatData_tmp;
            }
        }
```

```
    }
    return start;
}

    /**** CLEANUPs ****/

/* 清空搜索结果序列 */
void cleanupTeamWrapper(TeamWrapper *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("Nothing left to be cleaned");
        #endif
        return;
    }
    TeamWrapper *after = prev;
    while (1) {
        after = prev->next;
        free(prev);
        #if defined(BUILDING)
        printf("[LOG] cleanupTeamWrapper(): freed 0x%p\n", prev);
        #endif
        prev = after;
        if (prev == NULL) { break; }
    }
    return;
}

/* 清除统计结果链表 */
void cleanupTeamStatWrapper(TeamStatWrapper *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("Nothing left to be cleaned");
        #endif
        return;
    }
    TeamStatWrapper *after = prev;
    while (1) {
        after = prev->next;
        free(prev);
        #if defined(BUILDING)
        printf("[LOG] cleanupTeamWrapper(): freed 0x%p\n", prev);
        #endif
        prev = after;
        if (prev == NULL) { break; }
    }
    return;
}

/* 释放Team链所占用的内存空间 */
void cleanupTeam(Team *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("Nothing left to be cleaned");
        #endif
        return;
    }
    Team *after;
    while (1) {
        after = prev->next;
        #if defined(DEBUG)
        printf("[LOG] cleanupTeam(): freed 0x%p\n", prev);
        #endif
        free(prev->data);
        free(prev);
        prev = after;
        if (prev == NULL) { break; }
```

```
    }
    return;
}

/******** Unit Test ********/

#if defined(BUILDING)

void printTeamToConsole(Team *VirtusPro) {
    printf("<Team @ 0x%p>\n", VirtusPro);
    printf("\tthis.name = %s\n", VirtusPro->data->name);
    printf("\tthis.manager = %s\n", VirtusPro->data->manager);
    printf("\tthis.teacher_num = %d\n", VirtusPro->data->teacher_num);
    printf("\tthis.student_num = %d\n", VirtusPro->data->student_num);
    printf("\tthis.faculty = %s\n", VirtusPro->data->faculty);
}

void printTeamChainToConsole(Team *head) {
    if (head == NULL) {
        puts("no data!");
        return;
    }
    for (; head; head = head->next) {
        printTeamToConsole(head);
    }
    putchar('\n');
}

void printTeamWrapperToConsole(TeamWrapper *head) {
    printf("<TeamWrapper @ 0x%p>\n", head);
    if (head->team == NULL) {
        puts("\tnot match!");
    } else {
        for (; head; head = head->next) {
            printf("\tfound %s @ 0x%p\n",
                   head->team->data->name, head->team);
        }
    }
    putchar('\n');
}

void main(void) {
    // building test env
    DepartData depart_data_1 = {
        "计算机", "张三", "13322224444"
    };
    Depart depart_1 = {
        &depart_data_1, NULL, NULL, NULL
    };

    DepartData depart_data_2 = {
        "物理", "李四", "13344445555"
    };
    Depart depart_2 = {
        &depart_data_2, NULL, NULL, NULL
    };
    depart_1.next = &depart_2;
    Depart *Depart_HEAD = &depart_1;
    Team *Team_HEAD = createTeamHead();
    TeamData team_data_1 = {
        "火箭队", "武藏", 1, 2, "计算机"
    };
    TeamData team_data_2 = {
        "银河队", "小次郎", 2, 3, "物理"
    };
    TeamData team_data_3 = {
```

```
        "电子队", "洛伦兹", 3, 4, "计算机"
    };
    // appendTeam()
    puts("[LOG] adding team \"火箭队\"");
    appendTeam(Team_HEAD, team_data_1, Depart_HEAD);
    puts("[LOG] adding team \"银河队\"");
    appendTeam(Team_HEAD, team_data_2, Depart_HEAD);
    puts("[LOG] adding team \"电子队\"");
    appendTeam(Team_HEAD, team_data_3, Depart_HEAD);
    puts("Expecting sequence:\n\t 火箭队 --> 电子队 --> 银河队");
    printTeamChainToConsole(Team_HEAD);
    // modifyTeam()
    TeamData team_data_4 = {
        "电子队", "库仑", 3, 4, "计算机"
    };
    modifyTeam(Team_HEAD->next, team_data_4);
    printTeamChainToConsole(Team_HEAD);
    // getTeamByTeacherNum()
    TeamWrapper *Team_Wrapper_HEAD;
    Where cond = {">=", 2};
    Team_Wrapper_HEAD = getTeamByTeacherNum(Team_HEAD, NULL, cond);
    printTeamWrapperToConsole(Team_Wrapper_HEAD);
    cleanupTeamWrapper(Team_Wrapper_HEAD);
    // getTeamByName()
    Team_Wrapper_HEAD = getTeamByName(Team_HEAD, NULL, "火箭");
    printTeamWrapperToConsole(Team_Wrapper_HEAD);
    cleanupTeamWrapper(Team_Wrapper_HEAD);
    // removeTeam()
    puts("[LOG] removing team No.1");
    puts("Expecting sequence:\n\t 电子队 --> 银河队");
    removeTeam(&Team_HEAD, Team_HEAD);
    printTeamChainToConsole(Team_HEAD);
    puts("[LOG] removing team No.3");
    puts("Expecting sequence:\n\t 电子队");
    removeTeam(&Team_HEAD, Team_HEAD->next);
    printTeamChainToConsole(Team_HEAD);
    puts("[LOG] removing the last node");
    puts("Expecting literally NOTHING");
    removeTeam(&Team_HEAD, Team_HEAD);
    printTeamChainToConsole(Team_HEAD);
    cleanupTeam(Team_HEAD);
}

#endif

#ifdef BUILDING
#undef BUILDING
#endif

#ifdef DEBUG
#undef DEBUG
#endif
```

./utils/project_functions.h

```
/*  $PROGRAM_ROOT/utils/project_functions.h
 *  项目方法定义与实现
 */

#ifndef PROJECT_FUNCTIONS
#define PROJECT_FUNCTIONS

    /**** POST | DELETE | PUT ****/

extern ProjectData initProjectData(void);
```

```
/*  创建项目数据模板
 *  ARGS:   void
 *  RETN:   根据在该函数执行过程中输入的数据所创建出来的原型
 *  NOTE:   用户输入前端
 */

extern Project *appendProject(Project *, ProjectData, Team *);
/*  添加项目数据
 *  ARGS:   项目链表头，新增节点数据域模板，母结点链（团队链）
 *  RETN:   新增节点的地址
 */

extern int modifyProject(Project *, ProjectData);
/*  覆盖项目信息
 *  ARGS:   目标地址，已经修改数据域 buffer
 *  RETN:   success code
 */

extern int removeProject(Project **, Project *);
/*  删除项目节点
 *  ARGS:   指向团队链表头节点的指针，目标地址 | NULL
 *  RETN:   success code
 */

extern Project *createProjectHead(void);
/*  创建并初始化头节点
 *  ARGS:   void
 *  RETN:   头节点地址 || NULL
 */

    /**** SELECT ****/

extern ProjectWrapper *
getProjectById(Project *, Project *, const char *);
/*  通过 id 查找项目
 *  ARGS:   项目链表，搜索结束点，目标 id
 *  RETN:   搜索结果挂载点 | NULL （没有搜索结果时也返回挂载点地址）
 */

extern ProjectWrapper *getProjectByTeam(Team *);
/*  通过团队查找项目
 *  ARGS:   目标团队节点
 *  RETN:   搜索结果挂载点 || NULL （没有搜索结果也返回挂载点地址）
 */

    /**** CLEANUPs ****/

extern void cleanupProjectWrapper(ProjectWrapper *);
/*  清空搜索结果序列
 *  ARGS:   头节点地址
 *  RETN:   void
 */

extern void cleanupProject(Project *);
/*  释放 Project 链所占用的空间
 *  ARGS:   头节点地址
 *  RETN:   void
 */

#endif
```

./utils/project_functions.c

```
/*  $PROGRAM_ROOT/utils/project_functions.c
 *  项目方法定义与实现
 */

#ifndef DATA_STRUCTURE
#include "data_structure.h"
#endif

#ifndef FACULTY_FUNCTIONS
#include "faculty_functions.h"
#endif

#ifndef TEAM_FUNCTIONS
#include "team_functions.h"
#endif

#ifdef BUILDING
#undef BUILDING
#endif
// #define BUILDING

#ifdef DEBUG
#undef DEBUG
#endif
// #define DEBUG

/********** Declaration **********/
    /**** POST | DELETE | PUT ****/
ProjectData initProjectData(void);
Project *appendProject(Project *head, ProjectData new_one, Team
*team_chain);
int modifyProject(Project *tgt, ProjectData new_one);
int removeProject(Project **phead, Project *tgt);
Project *createProjectHead(void);
    /**** SELECT ****/
ProjectWrapper *getProjectById(Project *head, Project *, const char *id);
ProjectWrapper *getProjectByTeam(Team *parent_team);
    /**** CLEANUPs ****/
void cleanupProjectWrapper(ProjectWrapper *start);
void cleanupProject(Project *start);

/********** Function Realizations **********/

    /**** POST | DELETE | PUT ****/

/* 创建并初始化头节点 */
Project *createProjectHead(void) {
    Project *Project_HEAD = (Project *)malloc(sizeof(Project));
    if (Project_HEAD == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in createProjectHead():\n\tfailed to malloc for
project (head)");
        #endif
        return NULL;
    }
    Project_HEAD->data = NULL; Project_HEAD->next = NULL;
    Project_HEAD->parent_team = NULL;
    return Project_HEAD;
}

/* 创建项目数据模板 */
ProjectData initProjectData(void) {
    ProjectData Manhatan;
    Manhatan.type = '\0';
```

```
    printf("add/project::id > "); scanf("%s", Manhatan.id);
    printf("\
add/project::type \n\
    1 - 973\n\
    2 - NSFC\n\
    3 - 863\n\
    4 - International\n\
    5 - Transverse\n\
> ");
    int type_dec;
    // 类型代码值域检查
    do { scanf("%d", &type_dec); }
    while (type_dec < 1 || type_dec > 5);
    Manhatan.type = type_dec + '0';
    printf("add/project::start_date > "); scanf("%s", Manhatan.start_date);
    printf("add/project::funding > "); scanf("%f", &(Manhatan.funding));
    printf("add/project::manager > "); scanf("%s", Manhatan.manager);
    printf("add/project::team > "); scanf("%s", Manhatan.team);
    return Manhatan;
}

/* 添加项目数据 */
Project *appendProject(Project *head, ProjectData new_one,
                       Team *team_chain) {
    ProjectWrapper *project_wrapper = \
        getProjectById(head, NULL, new_one.id);
    if (project_wrapper->project != NULL) {
        printf("Record for project %s already exists!\n", new_one.id);
        cleanupProjectWrapper(project_wrapper);
        return NULL;
    }
    cleanupProjectWrapper(project_wrapper);
    Project *tail = head;
    for (; tail->next; tail = tail->next) ;
    // 获取所属团队节点
    TeamWrapper *parent_team_wrapper = \
        getTeamByName(team_chain, NULL, new_one.team);
    if (parent_team_wrapper == NULL) {
        #if defined(DEBUG)
        puts("[LOG] appendProject():\n\tgetTeamByName() failed");
        #endif
        return NULL;
    }
    if (parent_team_wrapper->team == NULL) {
        puts("Target parent team not found!\n");
        return NULL;
    }
    if (parent_team_wrapper->next != NULL) {
        puts("Multiple parent teams found!\n");
        return NULL;
    }
    if (strcmp(parent_team_wrapper->team->data->name, new_one.team)) {
        // 模糊查找，需要二次检查
        printf("Team named %s not found, do you mean %s instead?\n",
               new_one.team, parent_team_wrapper->team->data->name);
        return NULL;
    }
    Team *parent_team = parent_team_wrapper->team;
    cleanupTeamWrapper(parent_team_wrapper);
    #if defined(DEBUG)
    printf("[LOG] appendProject(): parent_team is %s @ 0x%p\n",
           parent_team->data->name, parent_team);
    #endif
    // 添加项目
    if (tail == head && tail->data == NULL) {
        tail->data = (ProjectData *)malloc(sizeof(ProjectData));
```

```
        if (tail->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendProject():\n\tfailed to malloc for
data (head)");
            #endif
            return NULL;
        }
        *(tail->data) = new_one; tail->next = NULL;
        tail->parent_team = parent_team;
        #if defined(CHILD_COUNTER)
        parent_team->data->project_num += 1;
        #endif
        parent_team->child_project_head = tail;
        parent_team->child_project_tail = tail;
        return tail;
    }
    if (parent_team->child_project_tail == NULL
            || parent_team->child_project_tail->next == NULL) {
        tail->next = (Project *)malloc(sizeof(Project));
        if (tail->next == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendProject():\n\tfailed to malloc for
data");
            #endif
            return NULL;
        }
        tail->next->data = (ProjectData *)malloc(sizeof(ProjectData));
        if (tail->next->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendProject():\n\tfailed to malloc for
data");
            #endif
            return NULL;
        }
        tail->next->next = NULL;
    } else {
        tail = parent_team->child_project_tail;
        Project *after = tail->next;
        tail->next = (Project *)malloc(sizeof(Project));
        if (tail->next == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendProject():\n\tfailed to malloc for
container");
            #endif
            return NULL;
        }
        tail->next->data = (ProjectData *)malloc(sizeof(ProjectData));
        if (tail->next->data == NULL) {
            #if defined(DEBUG)
            puts("[LOG] Error in appendProject():\n\tfailed to malloc for
data");
            #endif
            return NULL;
        }
        tail->next->next = after;
    }
    *(tail->next->data) = new_one;
    if (parent_team->child_project_head == NULL) {
        parent_team->child_project_head = tail->next;
    }
    parent_team->child_project_tail = tail->next;
    tail->next->parent_team = parent_team;
    return tail->next;
}

/* 覆盖项目信息 */
int modifyProject(Project *tgt, ProjectData new_one) {
```

```
    if (tgt == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in modifyProject():\n\ttarget is NULL");
        #endif
        return 0;
    }
    if (strcmp(tgt->data->team, new_one.team) != 0) {
        #if defined(DEBUG)
        printf("[LOG] Error in modifyProject():\n\tchanging project team,");
        printf(" from %s to %s, strcmp() got %d\n",
               tgt->data->team, new_one.team,
               strcmp(tgt->data->team, new_one.team));
        #endif
        return 0;
    }
    *(tgt->data) = new_one;
    return 1;
}

/* 删除项目节点 */
int removeProject(Project **phead, Project *tgt) {
    if (tgt == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in removeProject():\n\ttarget is NULL");
        #endif
        return 0;
    }
    Project *prev = *phead;
    for (; prev->next != NULL && prev->next != tgt; prev = prev->next) ;
    if (*phead == tgt && tgt->next) {
        *phead = tgt->next;
    } else {
        prev->next = tgt->next;
    }
    if (tgt->parent_team->child_project_head == tgt->parent_team-
>child_project_tail) {
        tgt->parent_team->child_project_head = NULL;
        tgt->parent_team->child_project_tail = NULL;
    } else if (tgt->parent_team->child_project_head == tgt) {
        tgt->parent_team->child_project_head = tgt->next;
    } else if (tgt->parent_team->child_project_tail == tgt) {
        tgt->parent_team->child_project_tail = prev;
    }
    #if defined(DEBUG)
    printf("[LOG] removeProject(): freed %s @ 0x%p\n",
           tgt->data->id, tgt);
    #endif
    free(tgt->data);
    if (tgt->next == NULL && *phead == tgt) { tgt->data = NULL; }
    else { free(tgt); }
    return 1;
}

    /**** SELECT ****/

/* 通过id查找项目 */
ProjectWrapper *
getProjectById(Project *start, Project *end, const char *id) {
    ProjectWrapper *rtn = (ProjectWrapper *)malloc(sizeof(ProjectWrapper));
    if (rtn == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getProjectById():\n\tfailed to malloc for
result mounting point");
        #endif
        return NULL;
    }
    ProjectWrapper *rtn_head = rtn;
```

```
    rtn_head->project = NULL; rtn_head->next = NULL;
    if (start->data == NULL) {
        #if defined(DEBUG)
        puts("[LOG] getProjectById(): searching an empty chain");
        #endif
        return rtn_head;
    }
    for (; start != end; start = start->next) {
        if (strcmp(start->data->id, id) == 0) {
            #if defined(BUILDING)
            printf("[LOG] getProjectById(): found %s @ 0x%p\n",
                   start->data->id, start);
            #endif
            if (rtn_head->project == NULL) {
                rtn->project = start;
            } else {
                rtn->next = (ProjectWrapper
*)malloc(sizeof(ProjectWrapper));
                if (rtn->next == NULL) {
                    #if defined(DEBUG)
                    puts("[LOG] Error in getProjectById():\n\tfailed to
malloc for result container");
                    #endif
                    cleanupProjectWrapper(rtn_head);
                    return NULL;
                }
                rtn = rtn->next; rtn->next = NULL;
                rtn->project = start;
            }
        }
    }
    return rtn_head;
}

/* 通过团队查找项目 */
ProjectWrapper *getProjectByTeam(Team *parent_team) {
    // 创建结果挂载点
    ProjectWrapper *rtn_head = (ProjectWrapper
*)malloc(sizeof(ProjectWrapper));
    if (rtn_head == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in getProjectByTeam():\n\tfailed to malloc for
result maunting point");
        #endif
        return NULL;
    }
    rtn_head->project = NULL; rtn_head->next = NULL;
    // 母结点没有子节点
    if (parent_team->child_project_head == NULL) {
        return rtn_head;    // 返回空结果
    }
    // 母结点下有子节点
    Project *cur = parent_team->child_project_head;
    ProjectWrapper *rtn = rtn_head;
    for (; cur != parent_team->child_project_tail->next; cur = cur->next) {
        #if defined(DEBUG)
        printf("[LOG] getProjectByTeam(): reached %s @ 0x%p\n",
               cur->data->id, cur);
        #endif
        if (rtn_head->project == NULL) {
            rtn->project = cur;
        } else {
            rtn->next = (ProjectWrapper *)malloc(sizeof(ProjectWrapper));
            if (rtn->next == NULL) {
                #if defined(DEBUG)
                puts("[LOG] Error in getProjectByTeam():\n\tfailed to malloc
```

```
for result container");
                #endif
                cleanupProjectWrapper(rtn_head);
                return NULL;
            }
            rtn = rtn->next;
            rtn->project = cur; rtn->next = NULL;
        }
    }
    return rtn_head;
}

    /**** CLEANUPs ****/

/* 清空搜索结果序列 */
void cleanupProjectWrapper(ProjectWrapper *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("Nothing left to be cleaned");
        #endif
        return;
    }
    ProjectWrapper *after;
    for (; prev; prev = after) {
        after = prev->next;
        #if defined(BUILDING)
        printf("[LOG] cleanupProjectWrapper(): freed 0x%p\n", prev);
        #endif
        free(prev);
    }
    return;
}

/* 释放 Project 链所占用的空间 */
void cleanupProject(Project *prev) {
    if (prev == NULL) {
        #if defined(DEBUG)
        puts("Nothing left to be cleaned");
        #endif
        return;
    }
    Project *after;
    for (; prev; prev = after) {
        after = prev->next;
        #if defined(DEBUG)
        printf("[LOG] cleanupProject(): freed 0x%p\n", prev);
        #endif
        free(prev->data);
        free(prev);
    }
    #if defined(DEBUG)
    puts("[LOG] cleanupProject(): exit");
    #endif
    return;
}

/********** Unit Test **********/

#if defined(BUILDING)

void printProjectToConsole(Project *Manhatan) {
    printf("<Project @ 0x%p>\n", Manhatan);
    printf("\tthis.id = %s\n", Manhatan->data->id);
    printf("\tthis.type = %c\n", Manhatan->data->type);
    printf("\tthis.start_date = %s\n", Manhatan->data->start_date);
    printf("\tthis.funding = %.2f\n", Manhatan->data->funding);
    printf("\tthis.manager = %s\n", Manhatan->data->manager);
```

```
    printf("\tthis.team = %s\n", Manhatan->data->team);
    printf("\tthis.next = 0x%p\n", Manhatan->next);
}

void printProjectChainToConsole(Project *start) {
    if (start == NULL) {
        puts("no data!");
        return;
    }
    for (; start; start = start->next) {
        printProjectToConsole(start);
    }
    putchar('\n');
}

void printProjectWrapperToConsole(ProjectWrapper *head) {
    printf("<ProjectWrapper @ 0x%p>\n", head);
    if (head->project == NULL) {
        puts("\tno data!");
        return;
    }
    for (; head; head = head->next) {
        printf("\tfound %s @ 0x%p\n",
               head->project->data->id, head->project);
    }
    putchar('\n');
    return;
}

void main(void) {
    // building test env
    DepartData depart_data_1 = {
        "计算机", "张三", "13344445555"
    };
    #if defined(CHILD_COUNTER)
    depart_data_1.team_num = 0;
    #endif
    TeamData team_data_1 = {
        "火箭队", "武藏", 2, 3, "计算机"
    };
    TeamData team_data_2 = {
        "银河队", "小次郎", 3, 4, "计算机"
    };
    ProjectData project_data_1 = {
        "123456", '1', "1970/01", 1.2, "王五", "火箭队"
    };
    ProjectData project_data_2 = {
        "123345", '2', "1980/01", 2.3, "赵六", "银河队"
    };
    ProjectData project_data_3 = {
        "123234", '3', "1990/01", 3.4, "李四", "火箭队"
    };
    Depart *Depart_HEAD = createDepartHead();
    Team *Team_HEAD = createTeamHead();
    Project *Project_HEAD = createProjectHead();
    appendDepart(Depart_HEAD, depart_data_1);
    appendTeam(Team_HEAD, team_data_1, Depart_HEAD);
    appendTeam(Team_HEAD, team_data_2, Depart_HEAD);
    // appendProject()
    puts("[LOG] adding project \"123456\"");
    appendProject(Project_HEAD, project_data_1, Team_HEAD);
    puts("[LOG] adding project \"123345\"");
    appendProject(Project_HEAD, project_data_2, Team_HEAD);
    puts("[LOG] adding project \"123234\"");
    appendProject(Project_HEAD, project_data_3, Team_HEAD);
    puts("Expecting sequence:\n\t123456 --> 123234 --> 123345");
```

```
    printProjectChainToConsole(Project_HEAD);
    // modifyProject()
    puts("[LOG] modifying project \"123234\"");
    ProjectData project_data_4 = {
        "123234", '4', "1990/01", 3.5, "李斯特", "火箭队"
    };
    modifyProject(Project_HEAD->next, project_data_4);
    printProjectChainToConsole(Project_HEAD);
    // SELECT
    ProjectWrapper *project_wrapper;
    puts("[LOG] getting project \"123234\" via id");
    project_wrapper = getProjectById(Project_HEAD, NULL, "123234");
    printProjectWrapperToConsole(project_wrapper);
    cleanupProjectWrapper(project_wrapper);
    // removeProject()
    puts("[LOG] removing project \"123456\"");
    removeProject(&Project_HEAD, Project_HEAD);
    printProjectChainToConsole(Project_HEAD);
    puts("[LOG] getting al project under \"火箭队\"");
    project_wrapper = getProjectByTeam(Team_HEAD);
    printProjectWrapperToConsole(project_wrapper);
    cleanupProjectWrapper(project_wrapper);
    puts("[LOG] removing project \"123345\"");
    removeProject(&Project_HEAD, Project_HEAD->next);
    printProjectChainToConsole(Project_HEAD);
    cleanupProject(Project_HEAD);
    cleanupTeam(Team_HEAD);
    cleanupDepart(Depart_HEAD);
    puts("Al Success!");
    return;
}

#endif

#ifdef BUILDING
#undef BUILDING
#endif

#ifdef DEBUG
#undef DEBUG
#endif
```

./utils/io_functions.h

```
/*  $PROGRAM_ROOT/utils/io_functions.h
 *  数据加载保存方法头文件
 */

#ifndef IO_FUNCTIONS
#define IO_FUNCTIONS

int saveData(MountPoint, const char *);
/*  保存方法（打包）
 *  ARGS:   数据挂载点。储存数据文件的目标文件夹
 *  RETN:   success code
 */

MountPoint loadData(const char *);
/*  载入数据方法（打包）
 *  ARGS:   储存数据文件的目标文件夹
 *  RETN:   院系链表、团队链表、项目链表的挂载点
 */

#endif
```

./utils/io_functions.c

```
/*  $PROGRAM_ROOT/utils/io_functions.c
 *  数据保存与载入方法定义与实现
 */


/********** Idea **********

${PROGRAMME_ROOT}
    +-- the_programme.exe
    +-- data/
        +-- DEPART.DAT
        +-- TEAM.DAT
        +-- PROJECT.DAT

*/

/********** Import **********/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "data_structure.h"
#include "faculty_functions.h"
#include "team_functions.h"
#include "project_functions.h"

/********** Controlers **********/

#ifdef BUILDING
#undef BUILDING
#endif
// #define BUILDING

#ifdef DEBUG
#undef DEBUG
#endif
// #define DEBUG

/********** Declaration **********/

int _saveDepartData(Depart *, const char *);
int _saveTeamData(Team *, const char *);
int _saveProjectData(Project *, const char *);
/*  分别保存方法
 *  ARGS:   要保存的数据链表，路径（endpoint 形式，无最后的'\'）
 *  RETN:   success code
 */
int saveData(MountPoint , const char *);
MountPoint loadData(const char *);

/********** Function Realization **********/

/* 内部函数 - 保存院系数据 */
int _saveDepartData(Depart *depart, const char *CUR_DIR) {
    // 生成目标地址
    char filename[100];
    sprintf(filename, "%s/%s", CUR_DIR, "DEPART.DAT");
    #if defined(BUILDING)
    printf("[LOG] _saveDepartData(): target URI is \"%s\"\n", filename);
    #endif
    // 尝试创建该文件
    FILE *fp;
    fp = fopen(filename, "wb");
```

```
    if (fp == NULL) {
        // NOTE: 若用户指定的文件夹不存在，也会抛出错误
        #if defined(DEBUG)
        printf("[LOG] Error in _saveDepartData():\n\tfailed to open file
%s\n",
               filename);
        #endif
        return 0;
    }
    // 写入数据
    for (; depart && depart->data; depart = depart->next) {
        // 直接整块写入结构体，读取也方便
        if (fwrite(depart->data, sizeof(DepartData), 1, fp) == 0) {
            #if defined(DEBUG)
            puts("[LOG] Error in _saveDepartData():\n\tfailed to write ");
            #endif
            return 0;
        }
        #if defined(DEBUG)
        printf("[LOG] _saveDepartData(): saved %s to file\n",
               depart->data->name);
        #endif
    }
    fclose(fp);
    return 1;
}

/* 内部函数 - 保存团队数据 */
int _saveTeamData(Team *team, const char *CUR_DIR) {
    char filename[100];
    sprintf(filename, "%s/%s", CUR_DIR, "TEAM.DAT");
    #if defined(BUILDING)
    printf("[LOG] _saveTeamData(): target URI is \"%s\"\n", filename);
    #endif
    FILE *fp;
    fp = fopen(filename, "wb");
    if (fp == NULL) {
        #if defined(DEBUG)
        printf("[LOG] Error in _saveTeamData():\n\tfailed to open file
%s\n",
               filename);
        #endif
        return 0;
    }
    for (; team && team->data; team = team->next) {
        if (fwrite(team->data, sizeof(TeamData), 1, fp) == 0) {
            #if defined(DEBUG)
            puts("[LOG] Error in _saveTeamData():\n\tfailed to write ");
            #endif
            return 0;
        }
        #if defined(DEBUG)
        printf("[LOG] _saveTeamData(): saved %s to file\n",
               team->data->name);
        #endif
    }
    fclose(fp);
    return 1;
}

/* 内部函数 - 保存项目数据 */
int _saveProjectData(Project *project, const char *CUR_DIR) {
    char filename[100];
    sprintf(filename, "%s/%s", CUR_DIR, "PROJECT.DAT");
    #if defined(BUILDING)
    printf("[LOG] _saveProjectData(): target URI is \"%s\"\n", filename);
```

```
    #endif
    FILE *fp;
    fp = fopen(filename, "wb");
    if (fp == NULL) {
        #if defined(DEBUG)
        printf("[LOG] Error in _saveProjectData():\n\tfailed to open file
%s\n",
               filename);
        #endif
        return 0;
    }
    for (; project && project->data; project = project->next) {
        if (fwrite(project->data, sizeof(ProjectData), 1, fp) == 0) {
            #if defined(DEBUG)
            puts("[LOG] Error in _saveProjectData():\n\tfailed to write ");
            #endif
            return 0;
        }
        #if defined(DEBUG)
        printf("[LOG] _saveProjectData(): saved %s to file\n",
               project->data->id);
        #endif
    }
    fclose(fp);
    return 1;
}

/* 数据保存方法 */
int saveData(MountPoint mp, const char *TGT_PATH) {
    if (_saveDepartData(mp.depart_head, TGT_PATH)
            && _saveTeamData(mp.team_head, TGT_PATH)
            && _saveProjectData(mp.project_head, TGT_PATH)) {
        return 1;
    }
    #if defined(DEBUG)
    puts("[LOG] Error in saveData():\n\tfailed to save");
    #endif
    return 0;
}

/* 数据加载方法 */
MountPoint loadData(const char *TGT_PATH) {
    // 创建挂载点副本
    MountPoint mp;
    mp.depart_head = createDepartHead();
    mp.team_head = createTeamHead();
    mp.project_head = createProjectHead();
    if (!(mp.depart_head && mp.team_head && mp.project_head)) {
        #if defined(DEBUG)
        puts("[LOG] Error in loadData():\n\tfailed to malloc for mounting
points");
        #endif
        return mp;
    }
    char filename[100];
    FILE *fp;
    DepartData depart_data_buf;
    TeamData team_data_buf;
    ProjectData project_data_buf;
    size_t counter = 0;     // 用于核实读取记录条数与文件大小是否相符
    // 加载院系数据
    // 打开数据文件
    sprintf(filename, "%s/%s", TGT_PATH, "DEPART.DAT");
    #if defined(DEBUG)
    printf("[LOG] loadData()::loadDepartData: target URI is \"%s\"\n",
filename);
```

```
    #endif
    fp = fopen(filename, "rb");
    if (fp == NULL) {       // 读取文件错误
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadDepartData:\n\tfailed to open
file");
        #endif
        free(mp.depart_head);
        // 找不到数据文件，不保留挂载点
        mp.depart_head = NULL;
        return mp;
    }
    // 开始读取
    counter = 0;
    while (fread(&depart_data_buf, sizeof(DepartData), 1, fp) == 1) {
        // 按结构体整块读入
        #if defined(DEBUG)
        printf("[LOG] loadData()::loadDepartData: reading %s to memory\n",
               depart_data_buf.name);
        #endif
        appendDepart(mp.depart_head, depart_data_buf);
        ++counter;
        // fp自己跳不用管
    }
    // 验证数据是否全部读入
    fseek(fp, 0, SEEK_END);     // fp挪到文件尾
    if (counter != ftell(fp) / sizeof(DepartData)) {    // 大小不一致
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadDepartData:\n\tfailed to read
al data");
        #endif
        // 仅保留挂载点
        cleanupDepart(mp.depart_head);
        mp.depart_head = createDepartHead();
        return mp;
    }
    fclose(fp);

    // 加载团队数据
    sprintf(filename, "%s/%s", TGT_PATH, "TEAM.DAT");
    #if defined(DEBUG)
    printf("[LOG] loadData()::loadTeamData: target URI is \"%s\"\n",
filename);
    #endif
    fp = fopen(filename, "rb");
    if (fp == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadTeamData:\n\tfailed to open
file");
        #endif
        free(mp.team_head);
        mp.team_head = NULL;
        return mp;
    }
    counter = 0;
    while (fread(&team_data_buf, sizeof(TeamData), 1, fp) == 1) {
        #if defined(DEBUG)
        printf("[LOG] loadData()::loadTeamData: reading %s to memory\n",
               team_data_buf.name);
        #endif
        appendTeam(mp.team_head, team_data_buf, mp.depart_head);
        ++counter;
    }
    fseek(fp, 0, SEEK_END);
    if (counter != ftell(fp) / sizeof(TeamData)) {
```

```
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadTeamData:\n\tfailed to read al
data");
        #endif
        cleanupTeam(mp.team_head);
        mp.team_head = createTeamHead();
        return mp;
    }
    fclose(fp);

    // 加载项目数据
    sprintf(filename, "%s/%s", TGT_PATH, "PROJECT.DAT");
    #if defined(BUILDING)
    printf("[LOG] loadData()::loadProjectData: target URI is \"%s\"\n",
filename);
    #endif
    fp = fopen(filename, "rb");
    if (fp == NULL) {
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadProjectData:\n\tfailed to open
file");
        #endif
        free(mp.project_head);
        mp.project_head = NULL;
        return mp;
    }
    counter = 0;
    while (fread(&project_data_buf, sizeof(ProjectData), 1, fp) == 1) {
        #if defined(DEBUG)
        printf("[LOG] loadData()::loadProjectData: reading %s to memory\n",
               project_data_buf.id);
        #endif
        appendProject(mp.project_head, project_data_buf, mp.team_head);
        ++counter;
    }
    fseek(fp, 0, SEEK_END);
    if (counter != ftell(fp) / sizeof(ProjectData)) {
        #if defined(DEBUG)
        puts("[LOG] Error in loadData()::loadProjectData:\n\tfailed to read
al data");
        #endif
        cleanupProject(mp.project_head);
        mp.project_head = createProjectHead();
        return mp;
    }
    fclose(fp);

    return mp;
}

/********** Unit Test **********/
#if defined(BUILDING)

void main(void) {
    // building test env
    DepartData depart_data_1 = {
        "Computer", "Zhang3", "13344445555"
    };
    #if defined(CHILD_COUNTER)
    depart_data_1.team_num = 0;
    #endif
    TeamData team_data_1 = {
        "Rocket", "Hanzo", 2, 3, "Computer"
    };
    TeamData team_data_2 = {
        "MilkyWay", "Genji", 3, 4, "Computer"
    };
```

```
    ProjectData project_data_1 = {
        "123456", '1', "1970/01", 1.2, "Wang5", "Rocket"
    };
    ProjectData project_data_2 = {
        "123345", '2', "1980/01", 2.3, "Zhao6", "MilkyWay"
    };
    ProjectData project_data_3 = {
        "123234", '3', "1990/01", 3.4, "Lee4", "Rocket"
    };
    Depart *Depart_HEAD = createDepartHead();
    Team *Team_HEAD = createTeamHead();
    Project *Project_HEAD = createProjectHead();
    appendDepart(Depart_HEAD, depart_data_1);
    appendTeam(Team_HEAD, team_data_1, Depart_HEAD);
    appendTeam(Team_HEAD, team_data_2, Depart_HEAD);
    appendProject(Project_HEAD, project_data_1, Team_HEAD);
    appendProject(Project_HEAD, project_data_2, Team_HEAD);
    appendProject(Project_HEAD, project_data_3, Team_HEAD);
    MountPoint mp;
    mp.depart_head = Depart_HEAD;
    mp.team_head = Team_HEAD;
    mp.project_head = Project_HEAD;
    // specifying path
    char *TGT_PATH = "./data";
    // _saveDepartData(Depart_HEAD, TGT_PATH);
    // _saveTeamData(Team_HEAD, TGT_PATH);
    // _saveProjectData(Project_HEAD, TGT_PATH);
    saveData(mp, TGT_PATH);
    // MountPoint loadData(const char *TGT_PATH) {
    mp = loadData(TGT_PATH);
    Depart *d; Team *t; Project *p;
    for (d = mp.depart_head; d != NULL; d = d->next) {
        printf("<Depart %s @ 0x%p>\n", d->data->name, d);
        for (t = d->child_team_head; t != d->child_team_tail->next; t = t-
>next) {
            printf("    <Team %s @ 0x%p>\n", t->data->name, t);
            for (p = t->child_project_head; p != t->child_project_tail-
>next; p = p->next) {
                printf("        <Project %s @ 0x%p>\n", p->data->id, p);
            }
        }
    }
    // cleanup
    // cleanupProject(Project_HEAD);
    // cleanupTeam(Team_HEAD);
    // cleanupDepart(Depart_HEAD);
    return;
}

#endif
```

./utils/makefile

```
#   $PROGRAM_ROOT/utils/makefile
#   单元测试用临时编译批处理
#   USAGE:
#       Win (with MinGW):   PS > mingw32-make
#                           PS > mingw32-make run
#       *nix:   $ make && ./a.exe
#   NOTE: 编译前解注释待测试模块的 DEBUG 宏，然后按需编译
#         如现在要测试团队模块功能，则编译命令应为:
            gcc -o a.exe ./faculty_functions.c ./team_functions.c

main:
    gcc -o a.exe ./faculty_functions.c ./team_functions.c
./project_functions.c ./io_functions.c
```

```
run:
    chcp 65001
    ./a.exe
```

II. 9 月 1 日更新前端展示

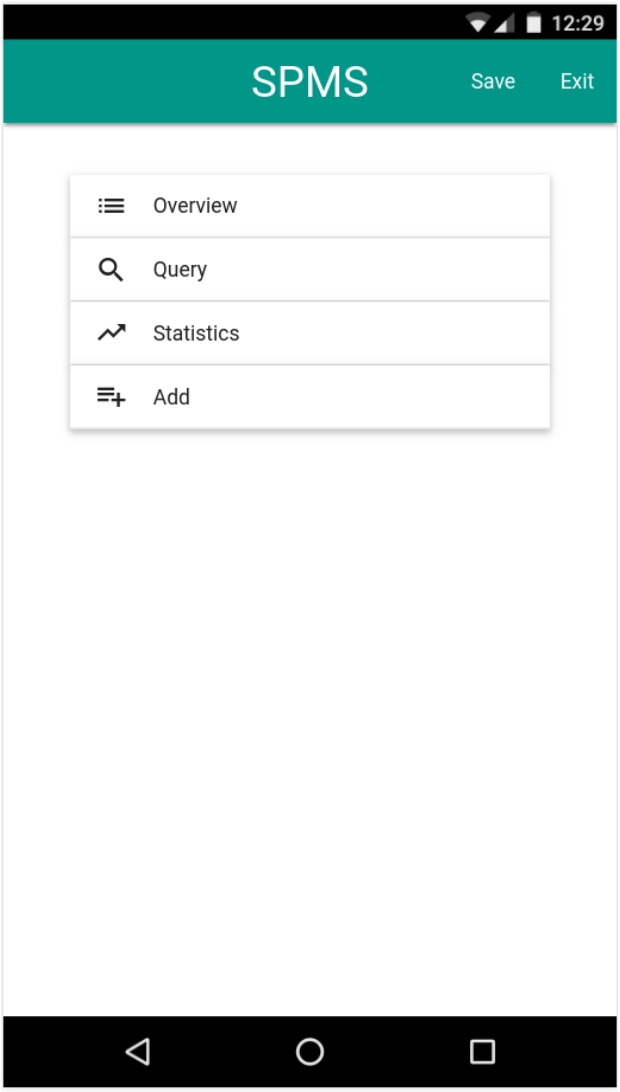

图 9.2.1 首页（手机版式）

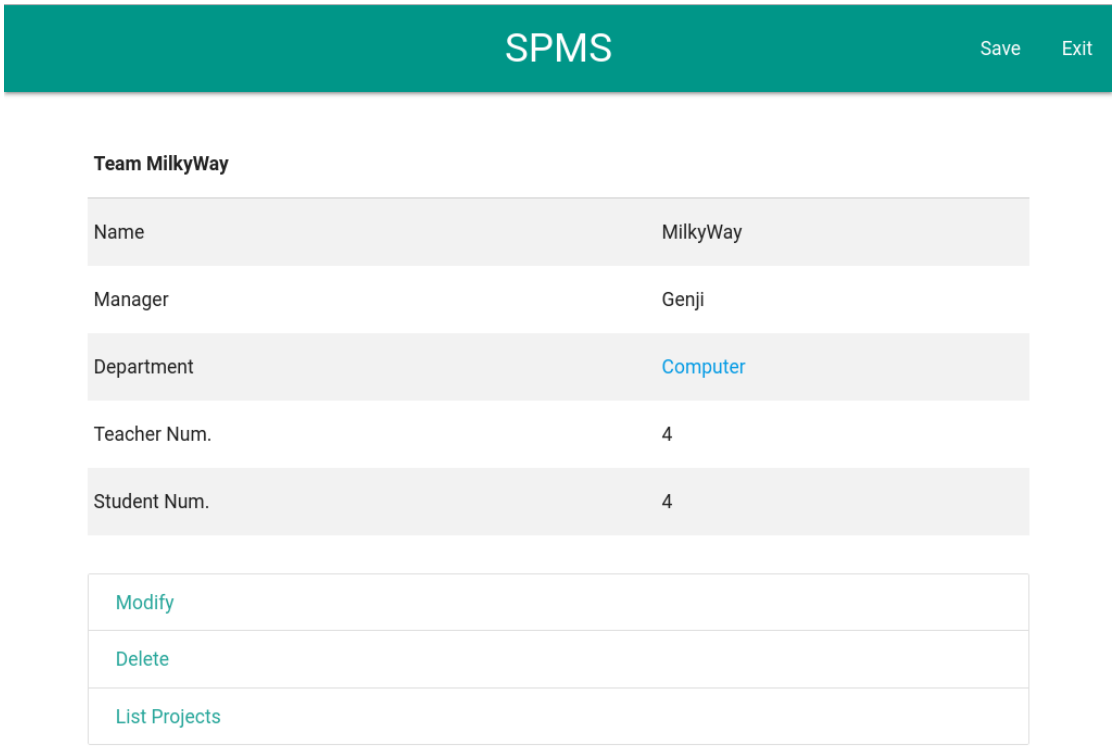

图 9.2.2 团队视图（平板版式）

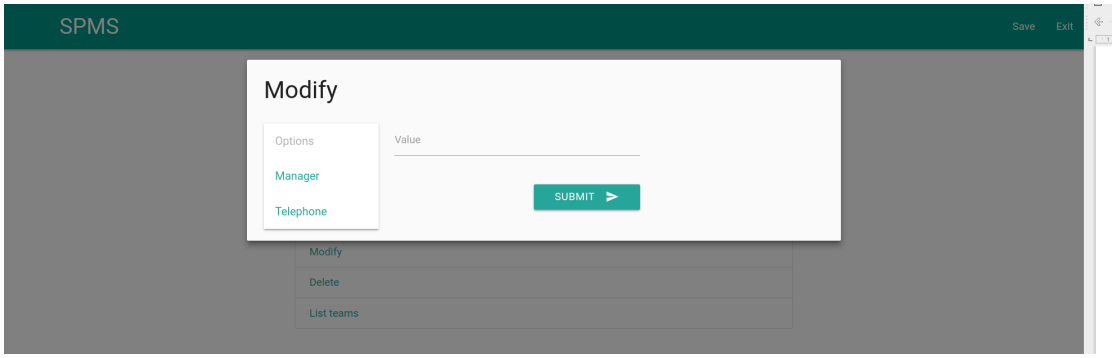

图 9.2.3 选择修改院系信息（桌面版式）

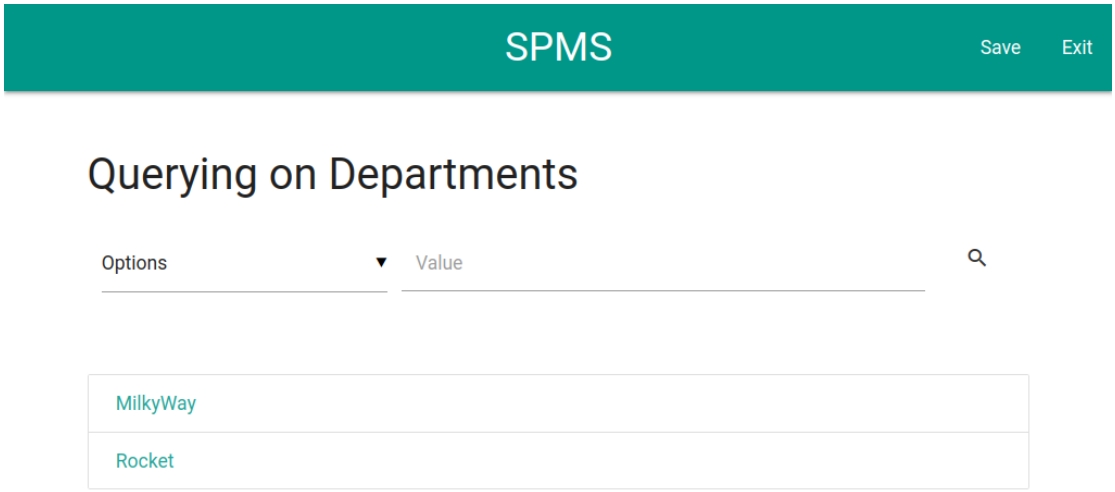

图 9.2.4 数据查询（平板版式）

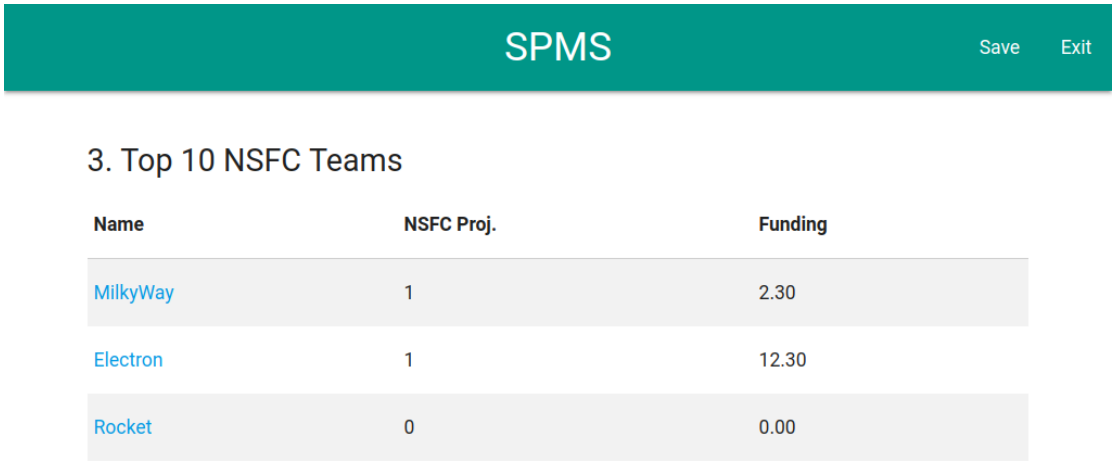

图 9.2.5 数据统计（平板视图）

更多内容还请自行探索。