

文件操作

胡船长

初航我带你，远航靠自己

本期内容

基础夯实

- 阶段1：将数据放在文件中
- 阶段2：实现单条数据的修改
- 阶段3：实现数据的二进制存储

项目开发

- 阶段4：实现用户交互流程
- 阶段5：实现多表的注册功能
- 阶段6：实现数据的增删改查

项目测试

- 1. 增加多表功能测试
- 2. 交互流程功能测试
- 3. 切换多表功能测试

阶段1：将数据放在文件中


1. 初识程序中的文件类型
2. 详解：文件打开模式
3. 读写文件的方法与技巧
4. 子项目1：学生信息管理系统

阶段1：将数据放在文件中

1. 初识程序中的文件类型
2. 详解：文件打开模式
3. 读写文件的方法与技巧
4. 子项目1：学生信息管理系统

初识程序中的文件类型

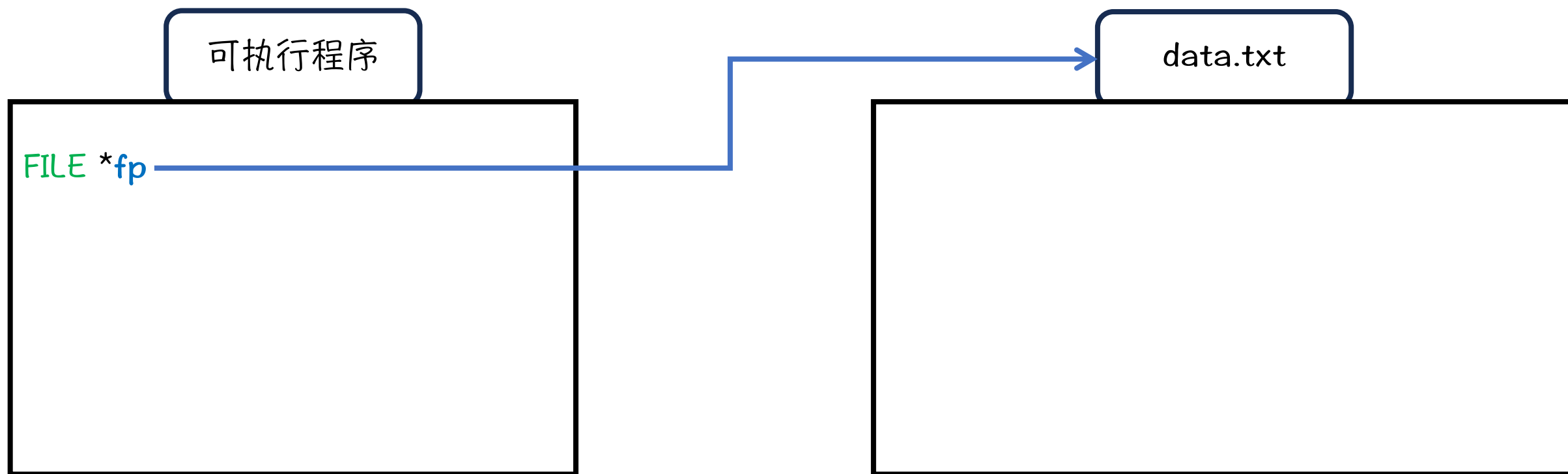
可执行程序

A large, empty rectangular box with a black border, representing the main body of an executable file.

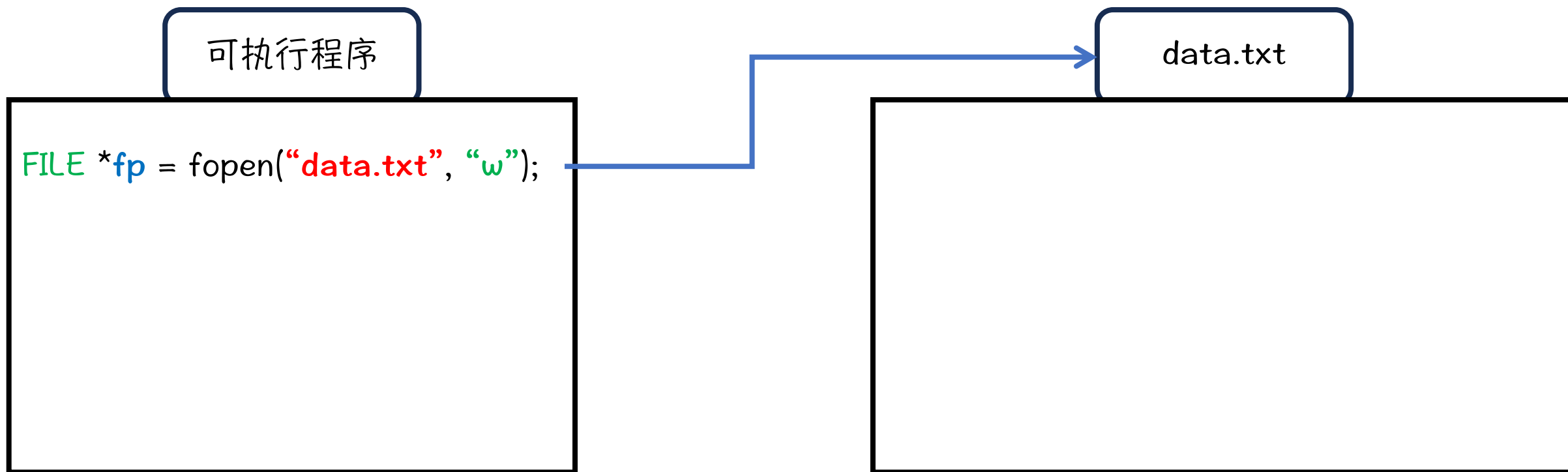
data.txt

A large, empty rectangular box with a black border, representing the main body of a data file.

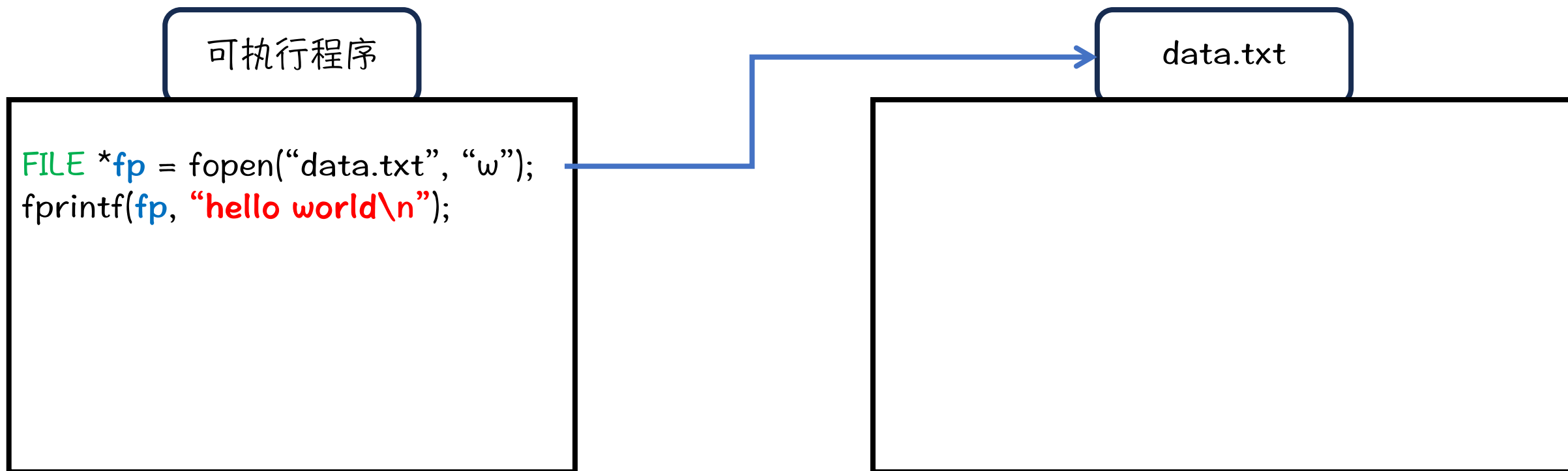
初识程序中的文件类型



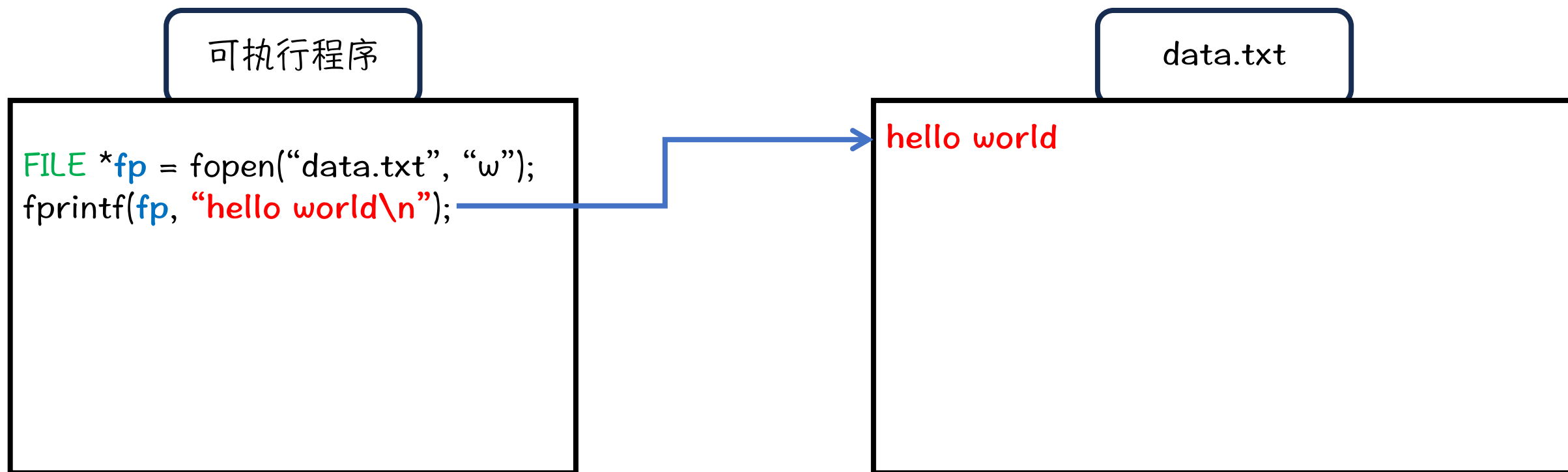
初识程序中的文件类型



初识程序中的文件类型



初识程序中的文件类型



阶段1：将数据放在文件中

1. 初识程序中的文件类型
2. 详解：文件打开模式
3. 读写文件的方法与技巧
4. 子项目1：学生信息管理系统

详解：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

详解：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

文件访问模式	含 义	解 释	文件存在的操作	文件不存在的操作
"r"	读	以读方式打开文件	从头开始读	打开失败
"w"	写	以写方式创建文件	清空内容	创建新文件
"a"	追加	向文件中追加内容	写入到末尾	创建新文件
"r+"	读扩展	以读写的方式打开一个文件	从头开始读	出错
"w+"	写扩展	以读写的方式创建一个文件	清空内容	创建新文件
"a+"	追加扩展	以读写方式打开一个文件	写入到末尾	创建新文件

详解：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

文件访问模式	含 义	解 释	文件存在的操作	文件不存在的操作
"r"	读	以读方式打开文件	从头开始读	打开失败
"w"	写	以写方式创建文件	清空内容	创建新文件
"a"	追加	向文件中追加内容	写入到末尾	创建新文件
"r+"	读扩展	以读写的方式打开一个文件	从头开始读	出错
"w+"	写扩展	以读写的方式创建一个文件	清空内容	创建新文件
"a+"	追加扩展	以读写方式打开一个文件	写入到末尾	创建新文件

详解：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

文件访问模式	含 义	解 释	文件存在的操作	文件不存在的操作
"r"	读	以读方式打开文件	从头开始读	打开失败
"w"	写	以写方式创建文件	清空内容	创建新文件
"a"	追加	向文件中追加内容	写入到末尾	创建新文件
"r+"	读扩展	以读写的方式打开一个文件	从头开始读	出错
"w+"	写扩展	以读写的方式创建一个文件	清空内容	创建新文件
"a+"	追加扩展	以读写方式打开一个文件	写入到末尾	创建新文件

阶段1：将数据放在文件中

1. 初识程序中的文件类型
2. 详解：文件打开模式
3. 读写文件的方法与技巧
4. 子项目1：学生信息管理系统

阶段1：将数据放在文件中

1. 初识程序中的文件类型
2. 详解：文件打开模式
3. 读写文件的方法与技巧
4. 子项目1：学生信息管理系统

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
```

```
mysql >
```

1.学生列表

id	name	age	class	height
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

2.添加学生

```
add new item : (name, age, class, height)
mysql > Hug 18 4 1.96
```

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

modify id : 0

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0
add new item : (name, age, class, height)
mysql > CaptainHu 32 3 1.73
```

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

4.删除学生

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99
3	test	18	2	1.96

```
delete id : 3
confirm (y / n) : y
```

子项目1：学生信息管理系统

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

4.删除学生

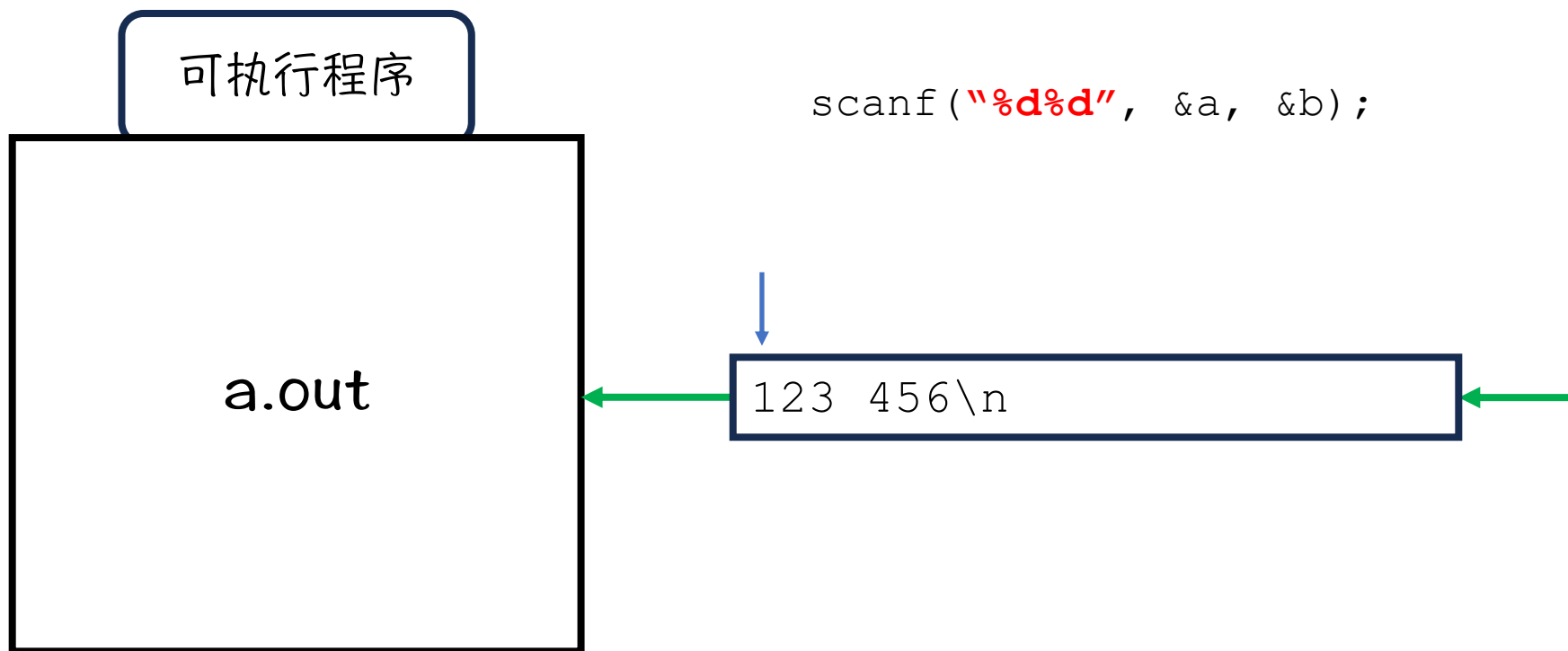
id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99
3	test	18	2	1.96

```
delete id : 3
confirm (y / n) : y
delete success
```

阶段2：实现单条数据的修改

1. 操作文件读写位置：fseek 与 ftell
2. 再学：文件打开模式
3. 优化1：实现数据的单条修改操作

scanf 函数的读入缓冲区



fseek 设置文件位置

可执行程序

```
FILE *fp = fopen("data.txt", "w");  
fprintf(fp, "0123456789");
```

data.txt

0123456789



fseek 设置文件位置

可执行程序

```
FILE *fp = fopen("data.txt", "w");  
fprintf(fp, "0123456789");  
fseek(fp, 2, SEEK_SET);
```

data.txt

↓
0123456789

fseek 设置文件位置

可执行程序

```
FILE *fp = fopen("data.txt", "w");  
fprintf(fp, "0123456789");  
fseek(fp, 2, SEEK_SET);  
fprintf(fp, "abc");
```

data.txt

↓
0123456789

fseek 设置文件位置

可执行程序

```
FILE *fp = fopen("data.txt", "w");  
fprintf(fp, "0123456789");  
fseek(fp, 2, SEEK_SET);  
fprintf(fp, "abc");
```

data.txt

01abc56789



fseek 设置文件位置

```
fseek(fp, 2, SEEK_SET);
```

SEEK_SET : 以文件开头位置为中心

SEEK_CUR : 以文件当前位置为中心

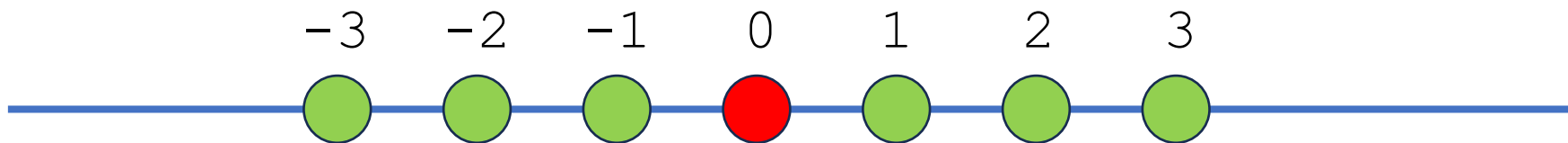
SEEK_END : 以文件结尾位置为中心

fseek 设置文件位置

SEEK_SET : 以文件开头位置为中心

SEEK_CUR : 以文件当前位置为中心

SEEK_END : 以文件结尾位置为中心



阶段2：实现单条数据的修改

1. 操作文件读写位置：fseek 与 ftell
2. 再学：文件打开模式
3. 优化1：实现数据的单条修改操作

再学：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

文件访问模式	含 义	解 释	文件存在的操作	文件不存在的操作
"r"	读	以读方式打开文件	从头开始读	打开失败
"w"	写	以写方式创建文件	清空内容	创建新文件
"a"	追加	向文件中追加内容	写入到末尾	创建新文件
"r+"	读扩展	以读写的方式打开一个文件	从头开始读	出错
"w+"	写扩展	以读写的方式创建一个文件	清空内容	创建新文件
"a+"	追加扩展	以读写方式打开一个文件	写入到末尾	创建新文件

再学：文件打开模式

```
FILE *fp = fopen("data.txt", "w");
```

文件访问模式	含 义	解 释	文件存在的操作	文件不存在的操作
"r"	读	以读方式打开文件	从头开始读	打开失败
"w"	写	以写方式创建文件	清空内容	创建新文件
"a"	追加	向文件中追加内容	写入到末尾	创建新文件
"r+"	读扩展	以读写的方式打开一个文件	从头开始读	出错
"w+"	写扩展	以读写的方式创建一个文件	清空内容	创建新文件
"a+"	追加扩展	以读写方式打开一个文件	写入到末尾	创建新文件

阶段2：实现单条数据的修改

1. 操作文件读写位置：fseek 与 ftell
2. 再学：文件打开模式
3. 优化1：实现数据的单条修改操作

优化1：实现数据的单条修改操作

交互界面

```
1 : list students
2 : add a student
3 : modify a student
4 : delete a student
5 : quit
mysql >
```

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0
add new item : (name, age, class, height)
mysql > CaptainHu 32 3 1.73
```

优化1：实现数据的单条修改操作

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0  
add new item : (name, age, class, height)  
mysql > CaptainHu 32 3 1.73
```

data.txt

```
CaptainHu 33 3 1.73  
sue 32 1 1.98  
song 50 2 1.99
```

优化1：实现数据的单条修改操作

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0  
add new item : (name, age, class, height)  
mysql > CaptainHu 32 3 1.73
```

data.txt

```
CaptainHu 32 3 1.73  
sue 32 1 1.98  
song 50 2 1.99
```

优化1：实现数据的单条修改操作

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0  
add new item : (name, age, class, height)  
mysql > CaptainHu 32 3 1.73
```

data.txt



```
CaptainHu 32 3 1.73  
sue 32 1 1.98  
song 50 2 1.99
```

优化1：实现数据的单条修改操作

3.修改信息

id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

```
modify id : 0  
add new item : (name, age, class, height)  
mysql > CaptainHu 32 3 1.73
```

data.txt



```
Hug 32 3 1.73  
sue 32 1 1.98  
song 50 2 1.99
```


优化1：实现数据的单条修改操作

3.修改信息

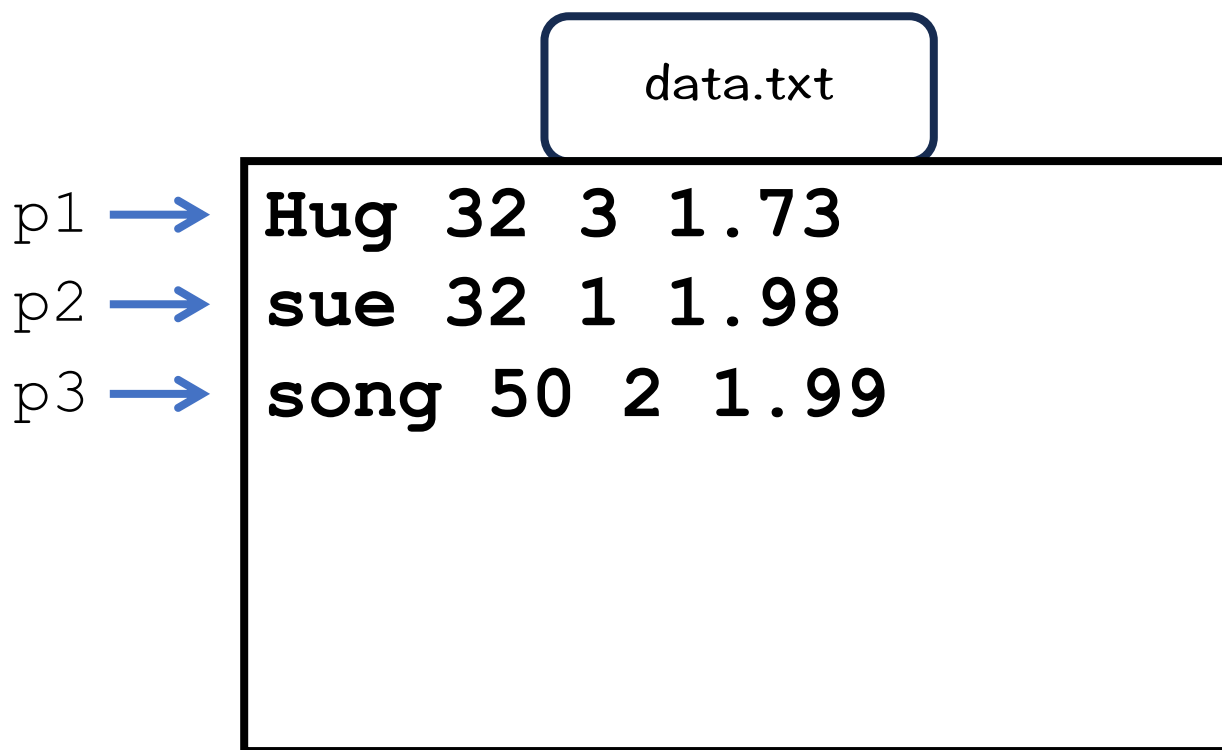
id	name	age	class	height
=====				
0	CaptainHu	33	3	1.73
1	sue	32	1	1.98
2	song	50	2	1.99

modify id : 0
add new item : (name, age, class, height)
mysql > CaptainHu 32 3 1.73

data.txt

→ Hug 32 3 1.73
→ sue 32 1 1.98
→ song 50 2 1.99

优化1：实现数据的单条修改操作



优化1：实现数据的单条修改操作

	data.txt			
p1 →	Hug	32	3	1.73
p2 →	sue	32	1	1.98
p3 →	song	50	2	1.99

阶段3：实现数据的二进制存储

1. 二进制文件的写与读：fwrite 与 fread
2. 优化2：实现数据的二进制存储

二进制文件的写与读

data.txt

Hug	32	3	1.73
sue	32	1	1.98
song	50	2	1.99

data.dat

```
0111001010101001110
010101100100101001010
010100101001001010010
010010101001010010010
101001010010101001001
00101010101010101
```

二进制文件的写与读

fwrite 函数

功能：向文件中写入二进制数据

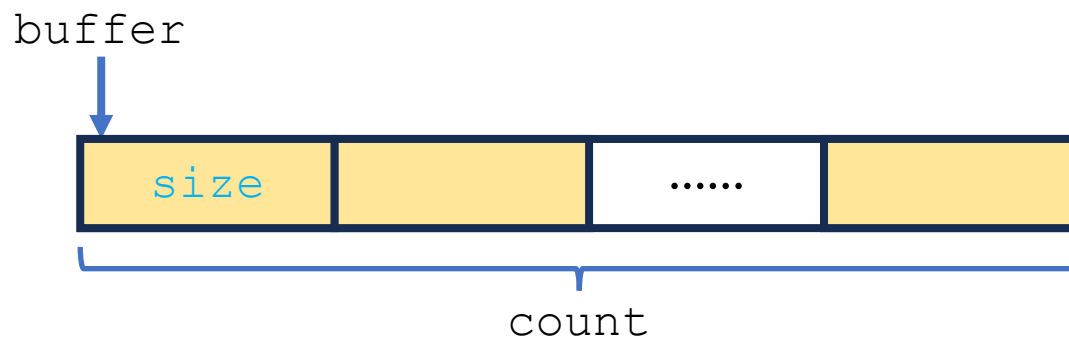
原型：`fwrite(const void * buffer, size_t size, size_t count, FILE *fp);`

`buffer`：数据区的首地址

`size`：每个数据的大小

`count`：写入数据个数

`fp`：文件指针



二进制文件的写与读

fread 函数

功能：从文件中读入二进制数据

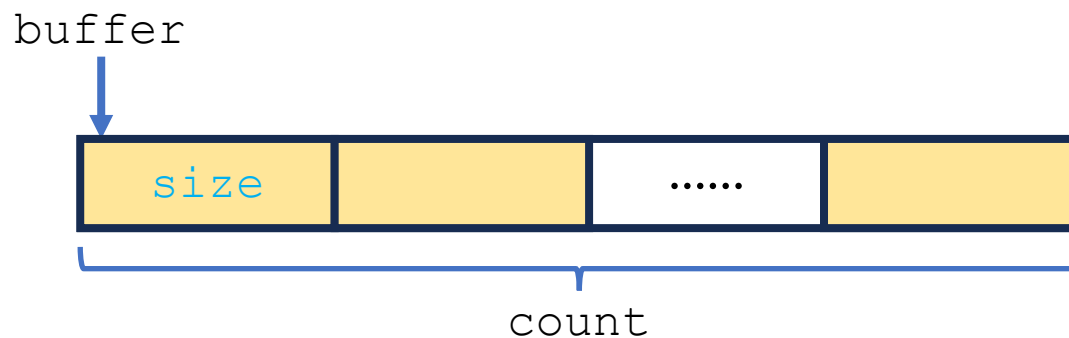
原型：`fread(const void * buffer, size_t size, size_t count, FILE *fp);`

`buffer`：数据区的首地址

`size`：每个数据的大小

`count`：读入数据个数

`fp`：文件指针



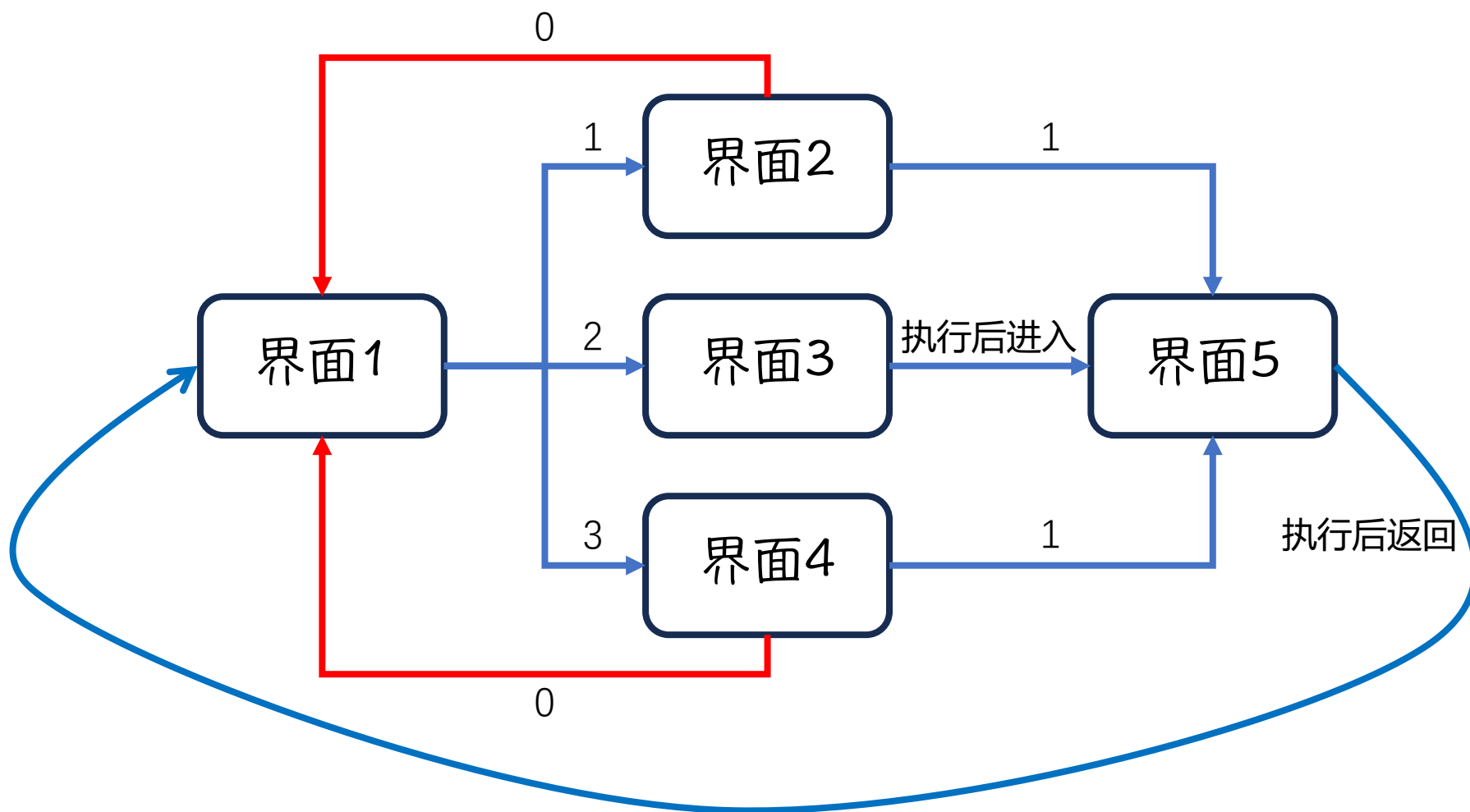
阶段3：实现数据的二进制存储

1. 二进制文件的写与读：fwrite 与 fread
2. 优化2：实现数据的二进制存储

阶段4：实现用户交互流程

1. 程序设计：操作界面之间的切换流程
2. 项目设计：分离操作层与数据层
3. 项目实施1：用户交互流程

操作界面之间的切换流程




阶段4：实现用户交互流程

1. 程序设计：操作界面之间的切换流程
2. 项目设计：分离操作层与数据层
3. 项目实施1：用户交互流程

分离操作层与数据层

分离操作层与数据层

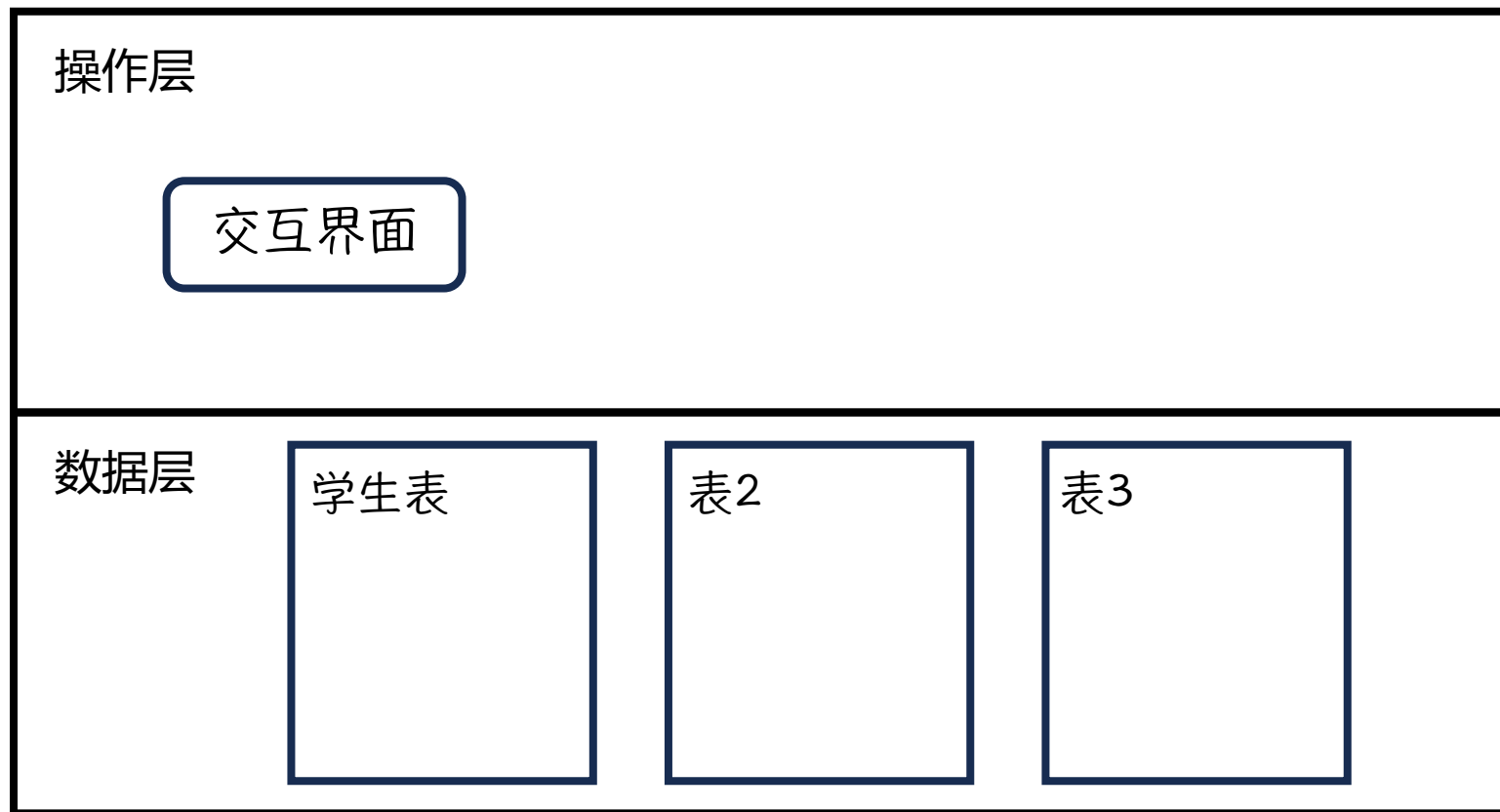


操作层

分离操作层与数据层



分离操作层与数据层



阶段4：实现用户交互流程

1. 程序设计：操作界面之间的切换流程
2. 项目设计：分离操作层与数据层
3. 项目实施1：用户交互流程

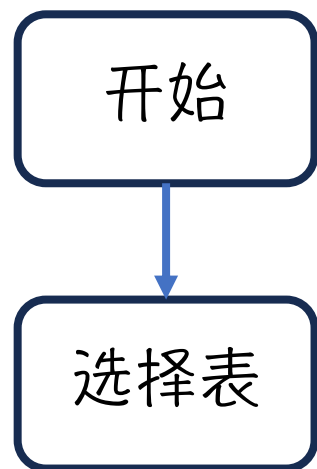
项目实施：用户交互流程



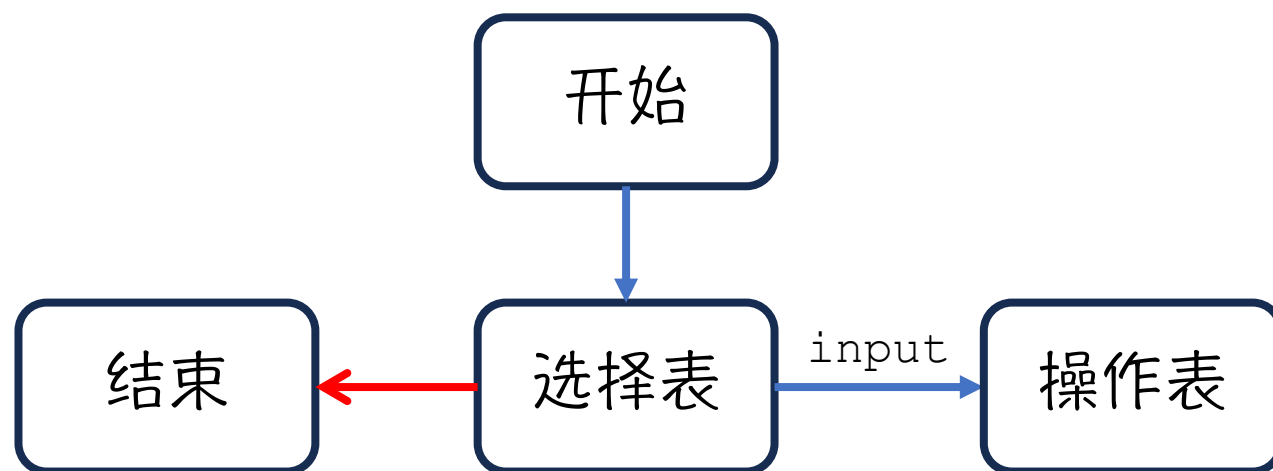
```
graph TD; A[开始]
```

开始

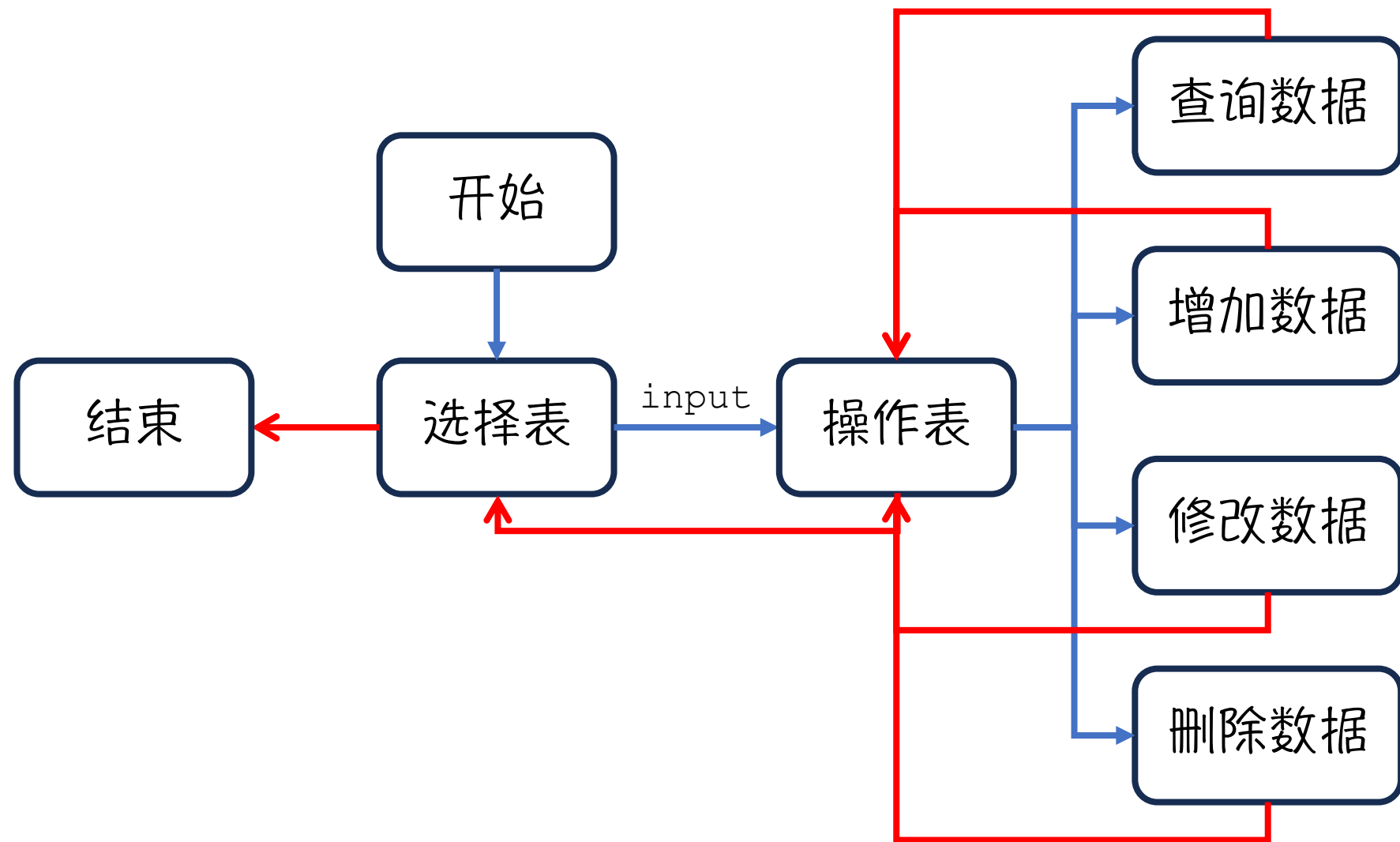
项目实施：用户交互流程



项目实施：用户交互流程



项目实施：用户交互流程



阶段5：实现多表的注册功能

1. 什么是『注册函数』
2. 项目设计：交互过程中的配置信息
3. 项目实施2：多表的注册功能

什么是『注册函数』

一切从 __attribute__ 关键字开始

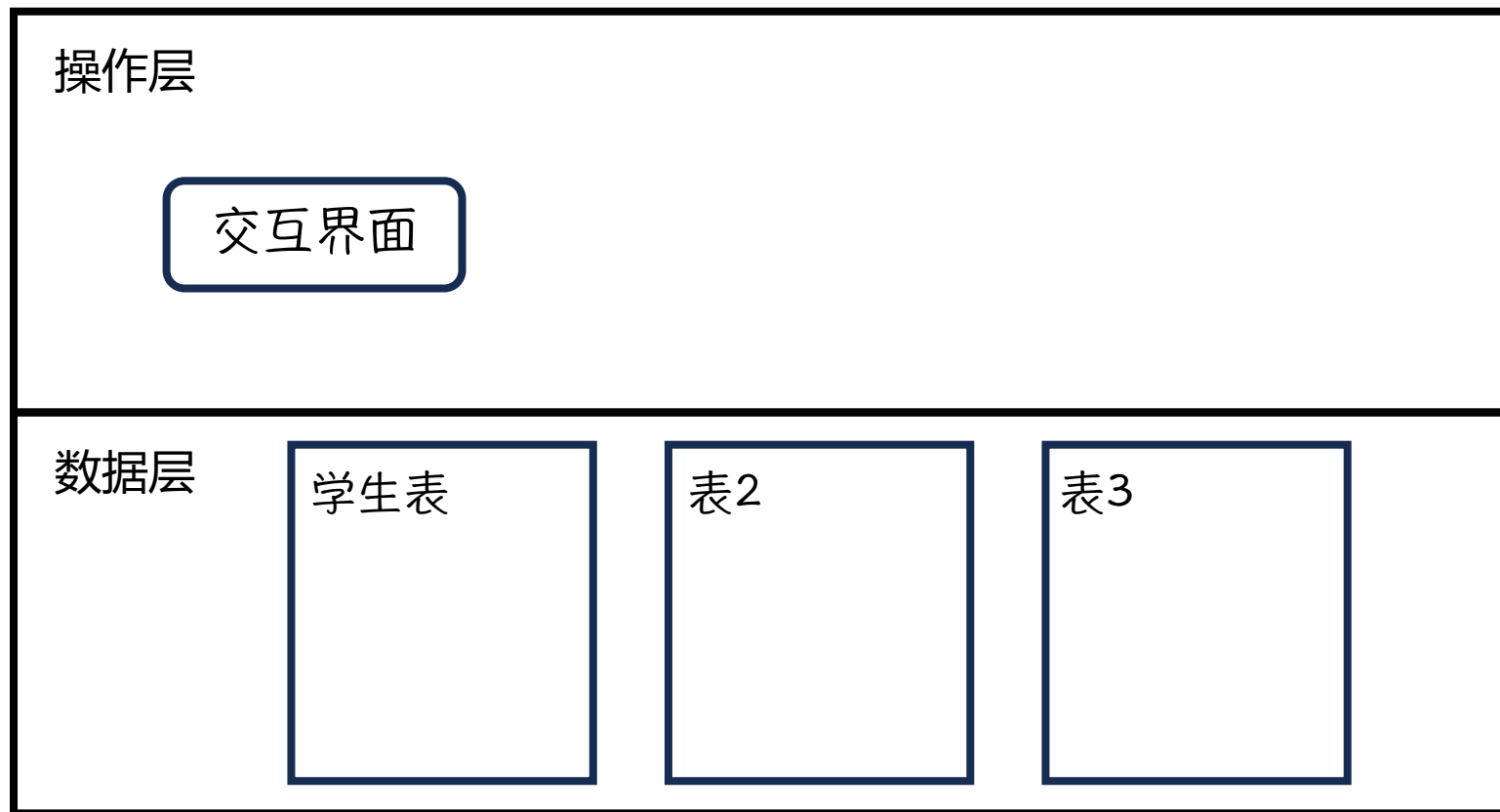
什么是『注册函数』

将某个功能注册到系统中的函数

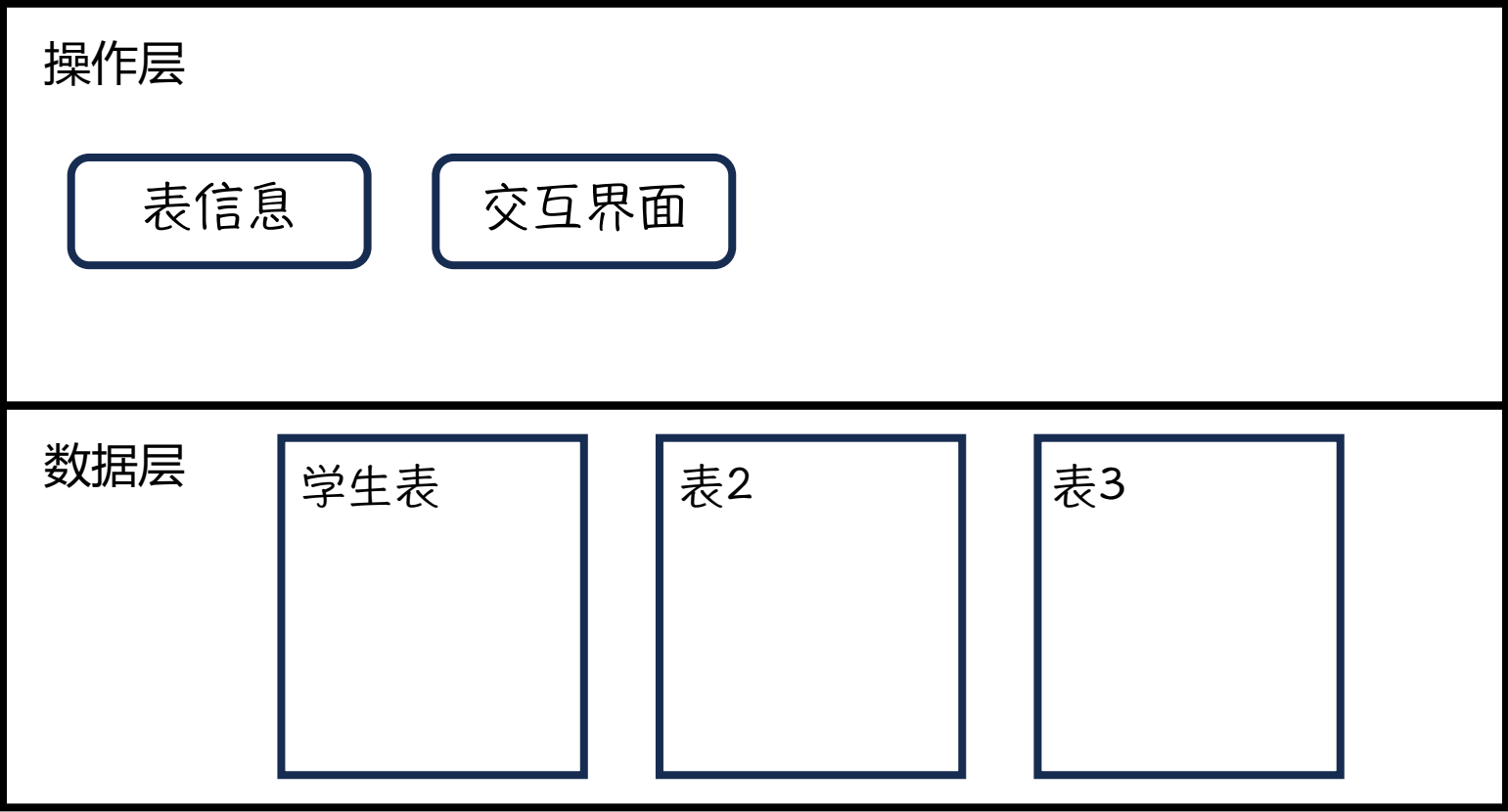
阶段5：实现多表的注册功能

1. 什么是『注册函数』
2. 项目设计：交互过程中的配置信息
3. 项目实施2：多表的注册功能

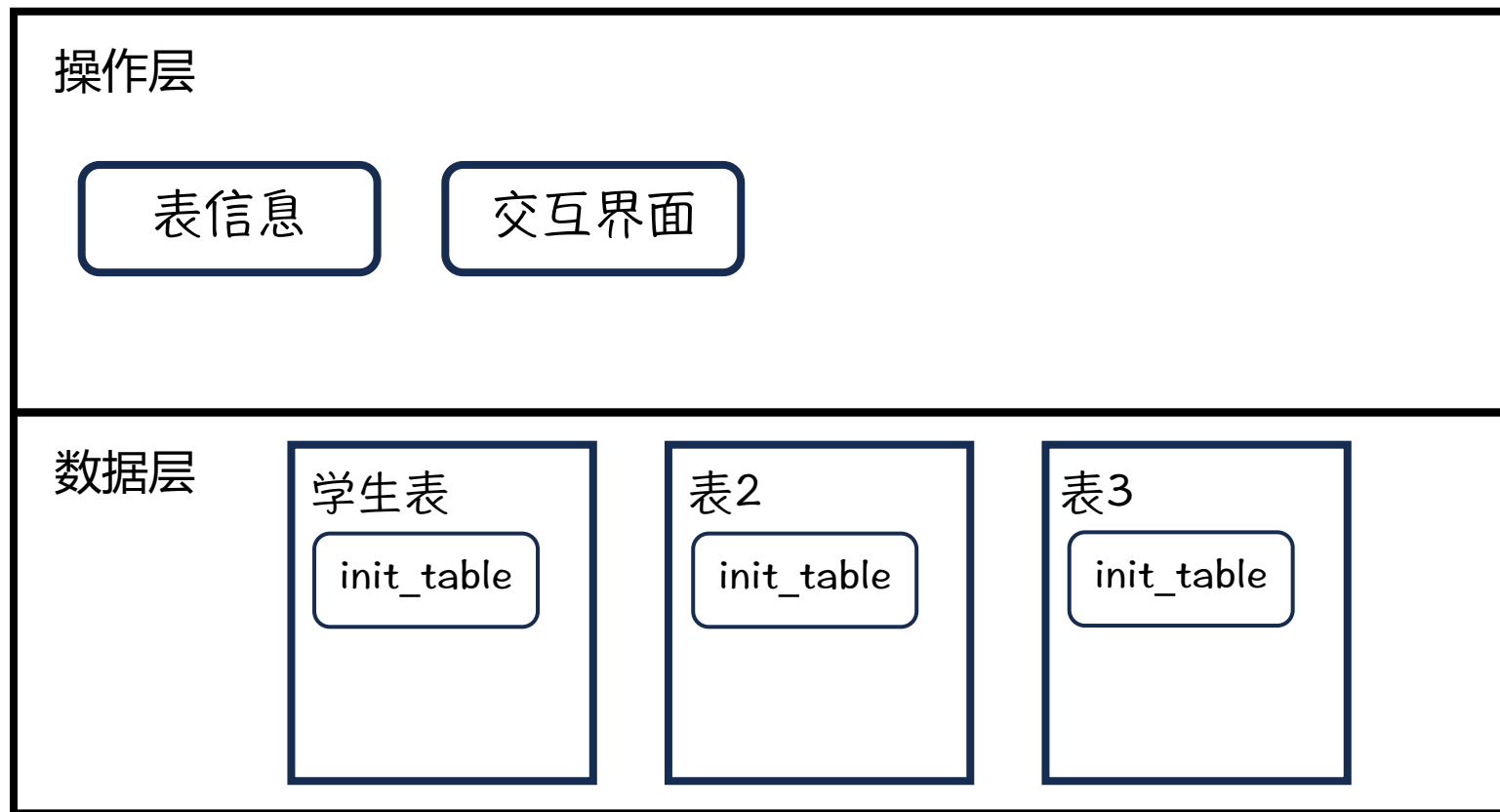
交互过程中的配置信息



交互过程中的配置信息



交互过程中的配置信息



阶段5：实现多表的注册功能

1. 什么是『注册函数』
2. 项目设计：交互过程中的配置信息
3. 项目实施2：多表的注册功能

阶段6：实现数据的增删改查

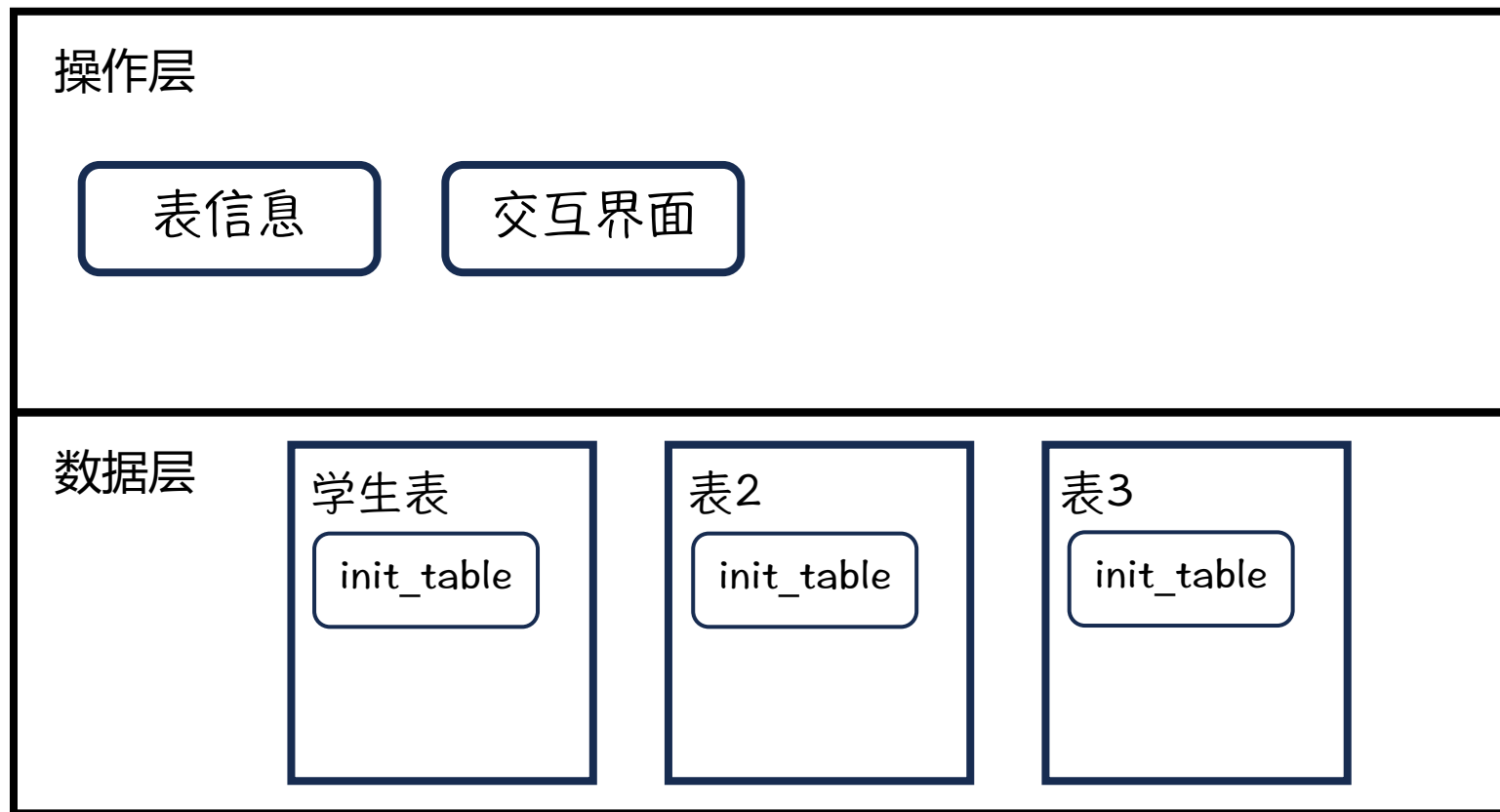
1. 项目实施3：查询

2. 项目实施4：增加

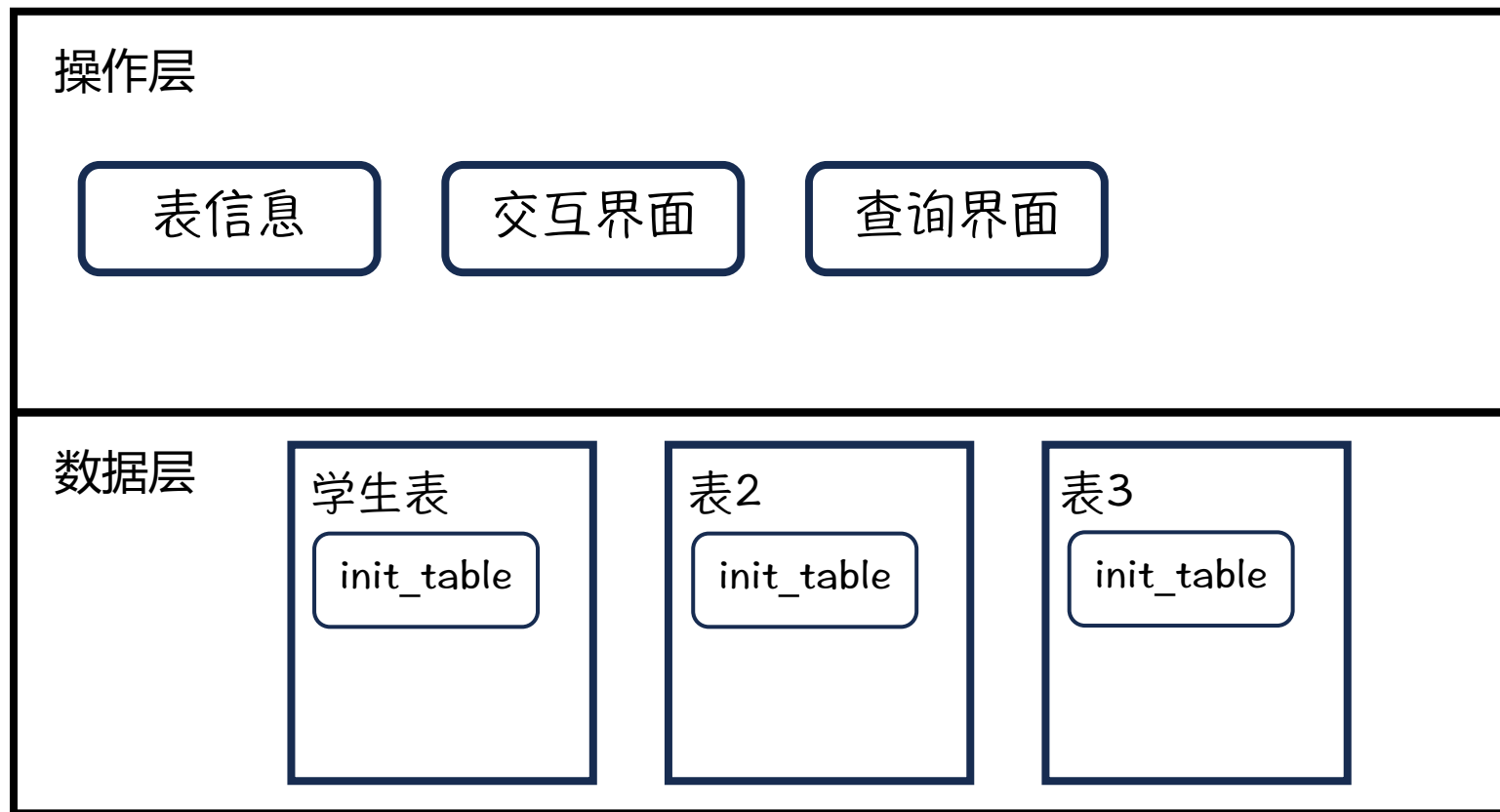
3. 项目实施5：修改

4. 项目实施6：删除

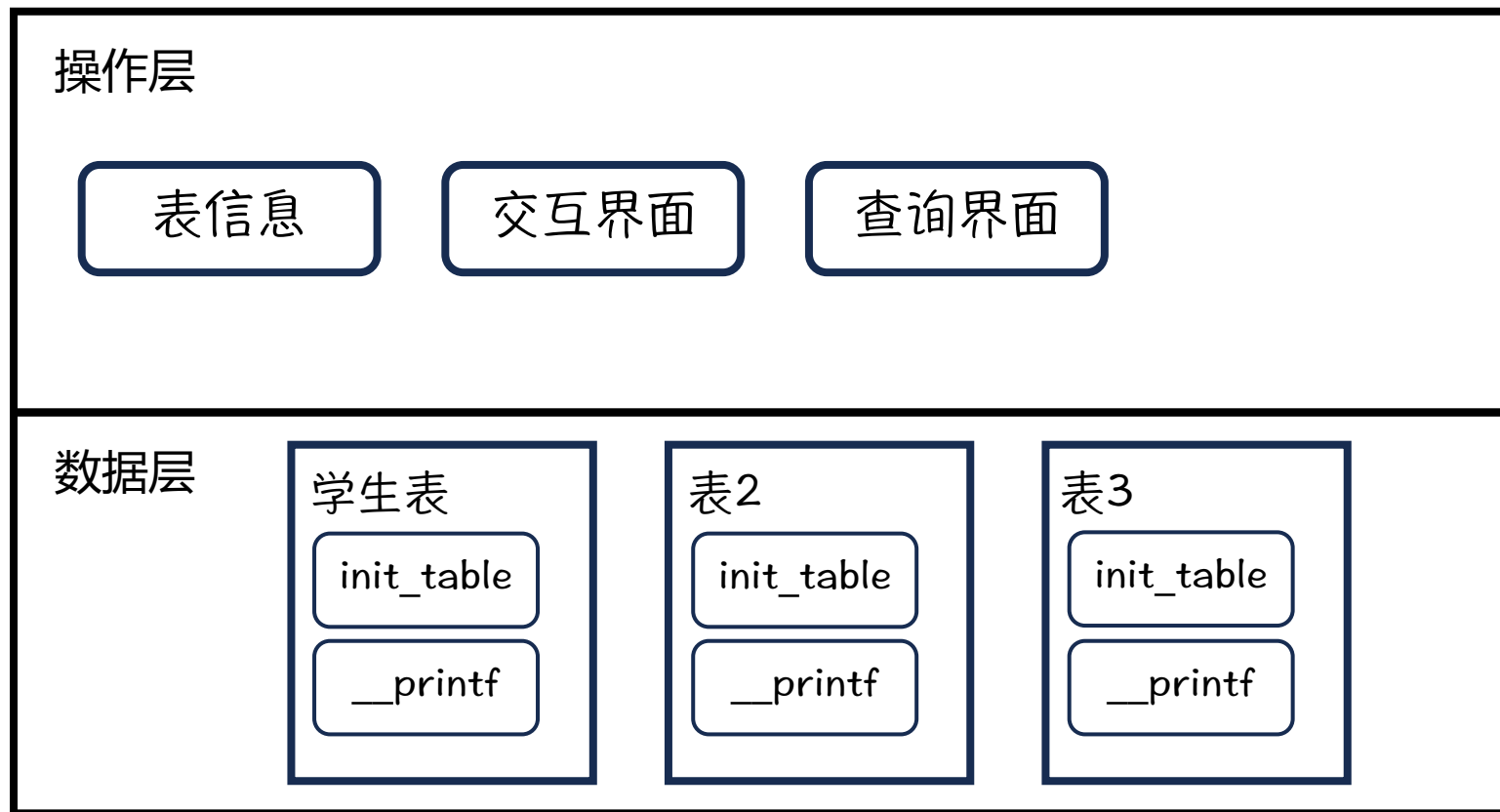
项目实施3：查询



项目实施3：查询



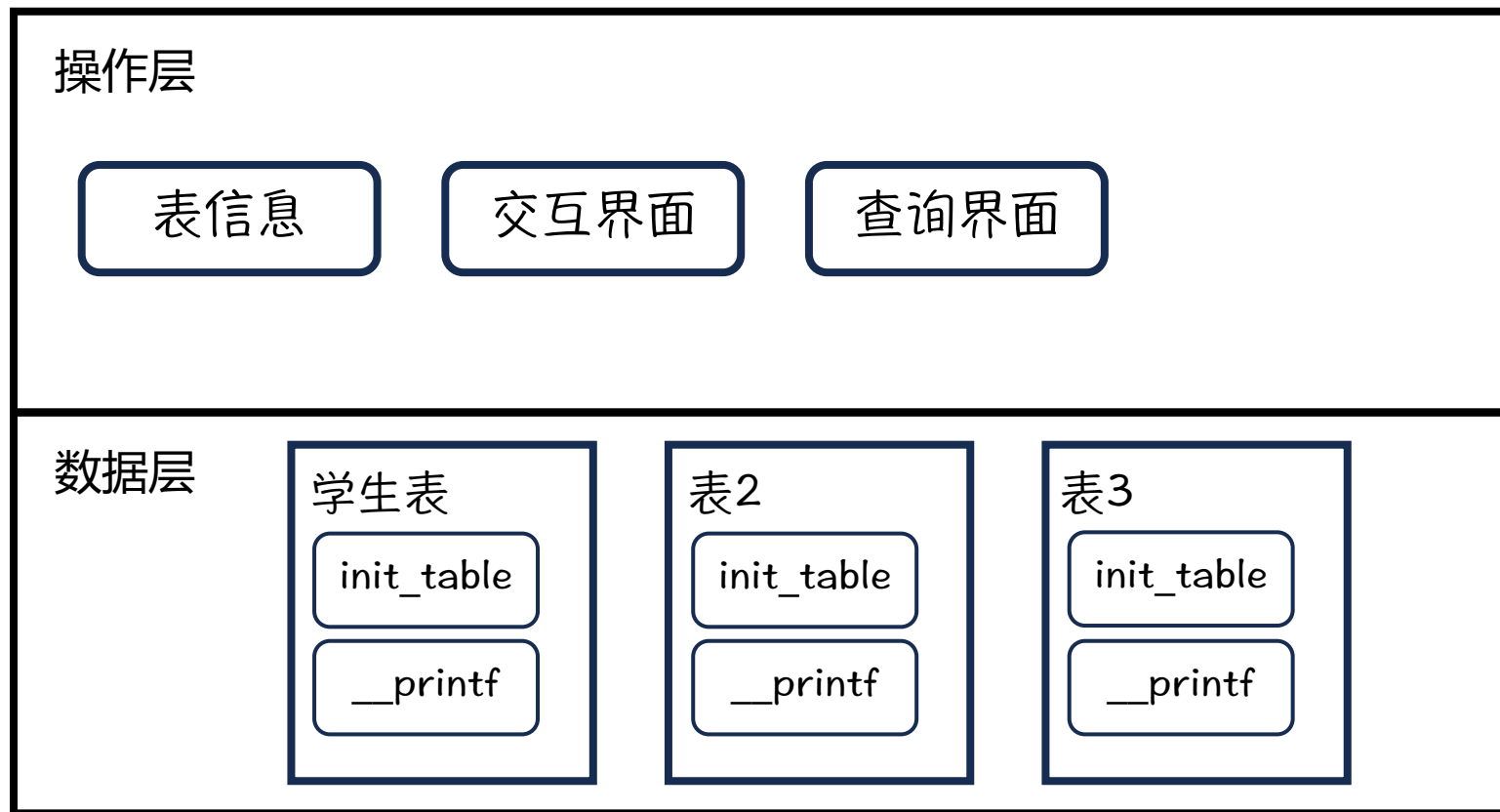
项目实施3：查询



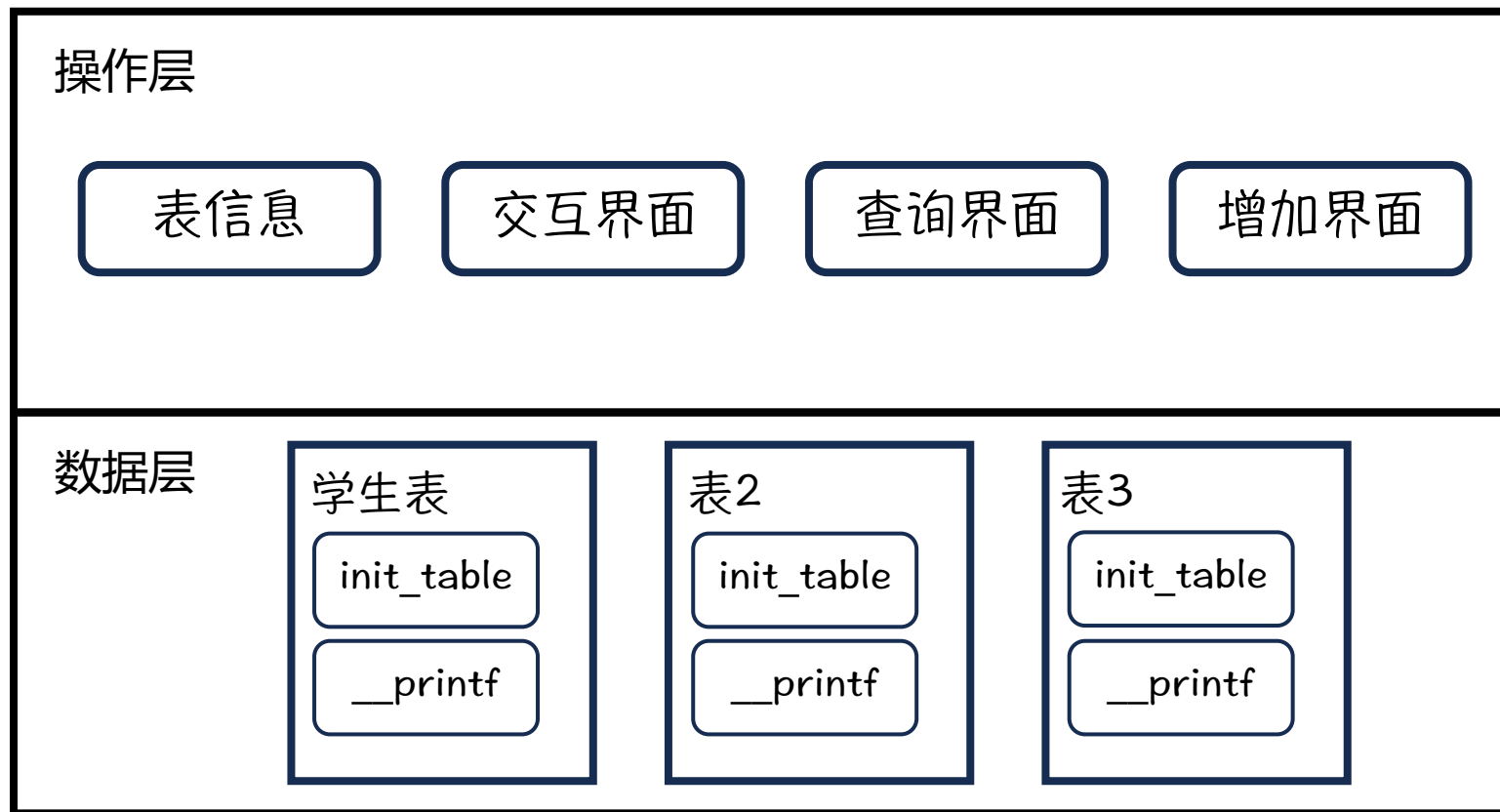
阶段6：实现数据的增删改查

1. 项目实施3：查询
- 2. 项目实施4：增加**
3. 项目实施5：修改
4. 项目实施6：删除

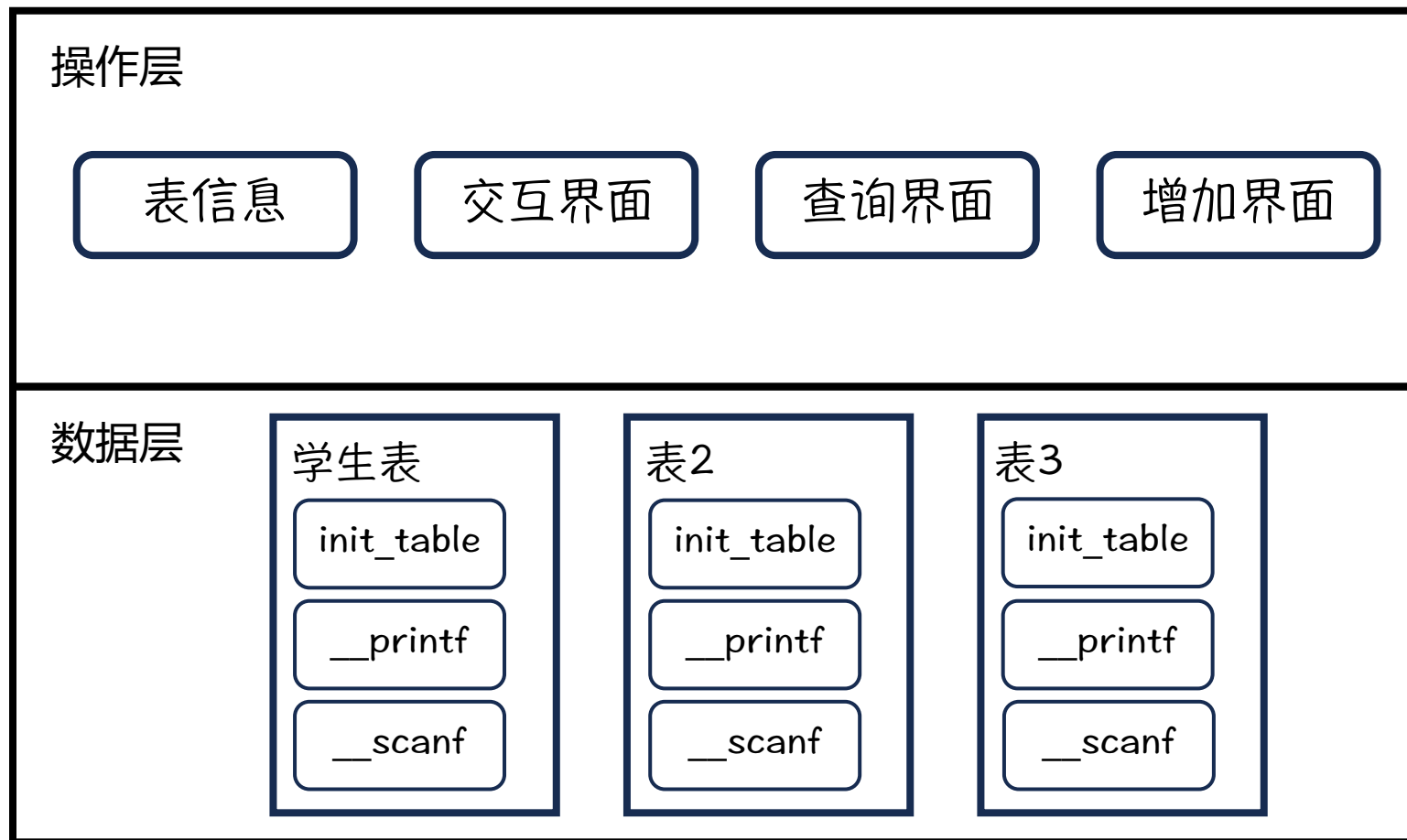
项目实施4：增加



项目实施4：增加



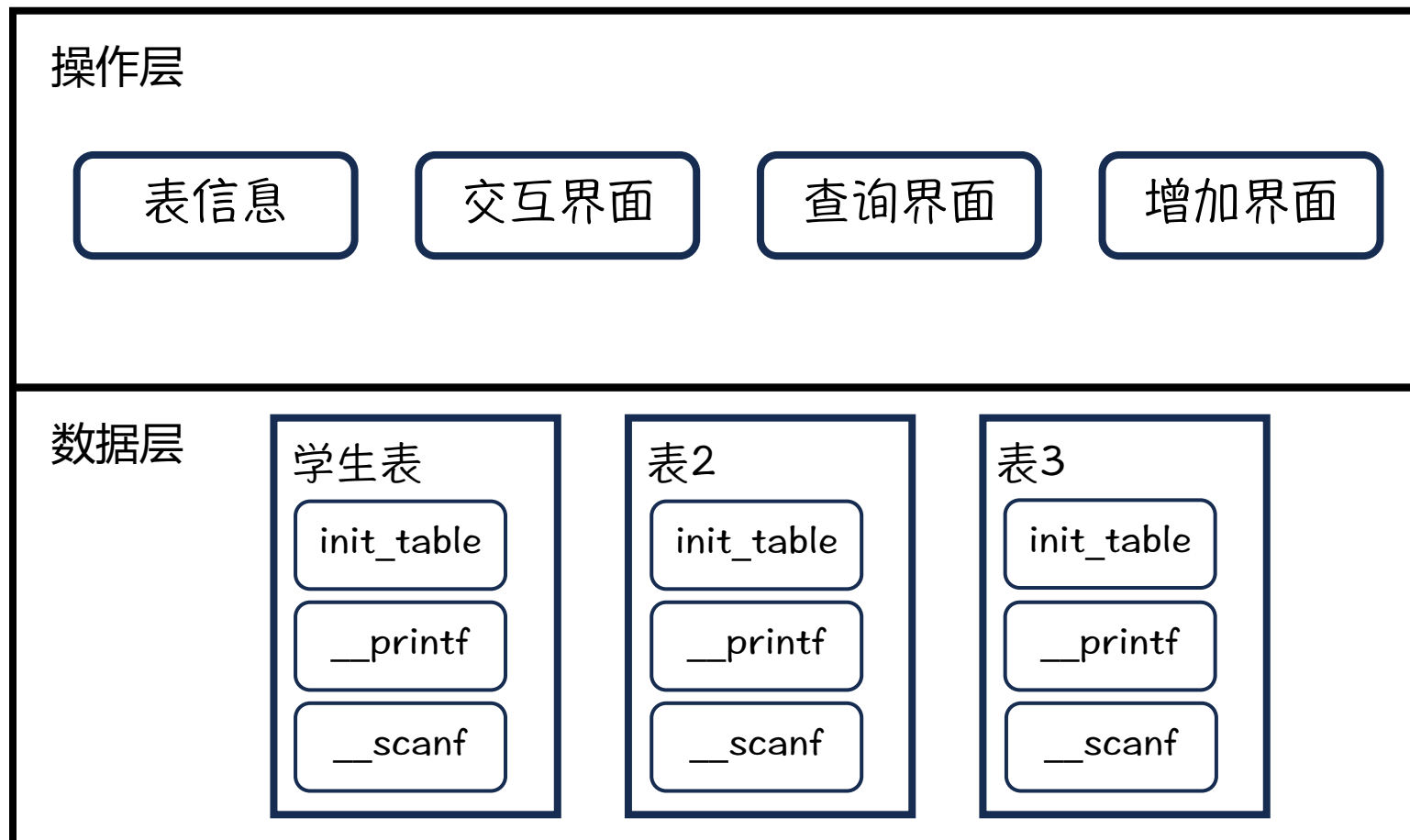
项目实施4：增加



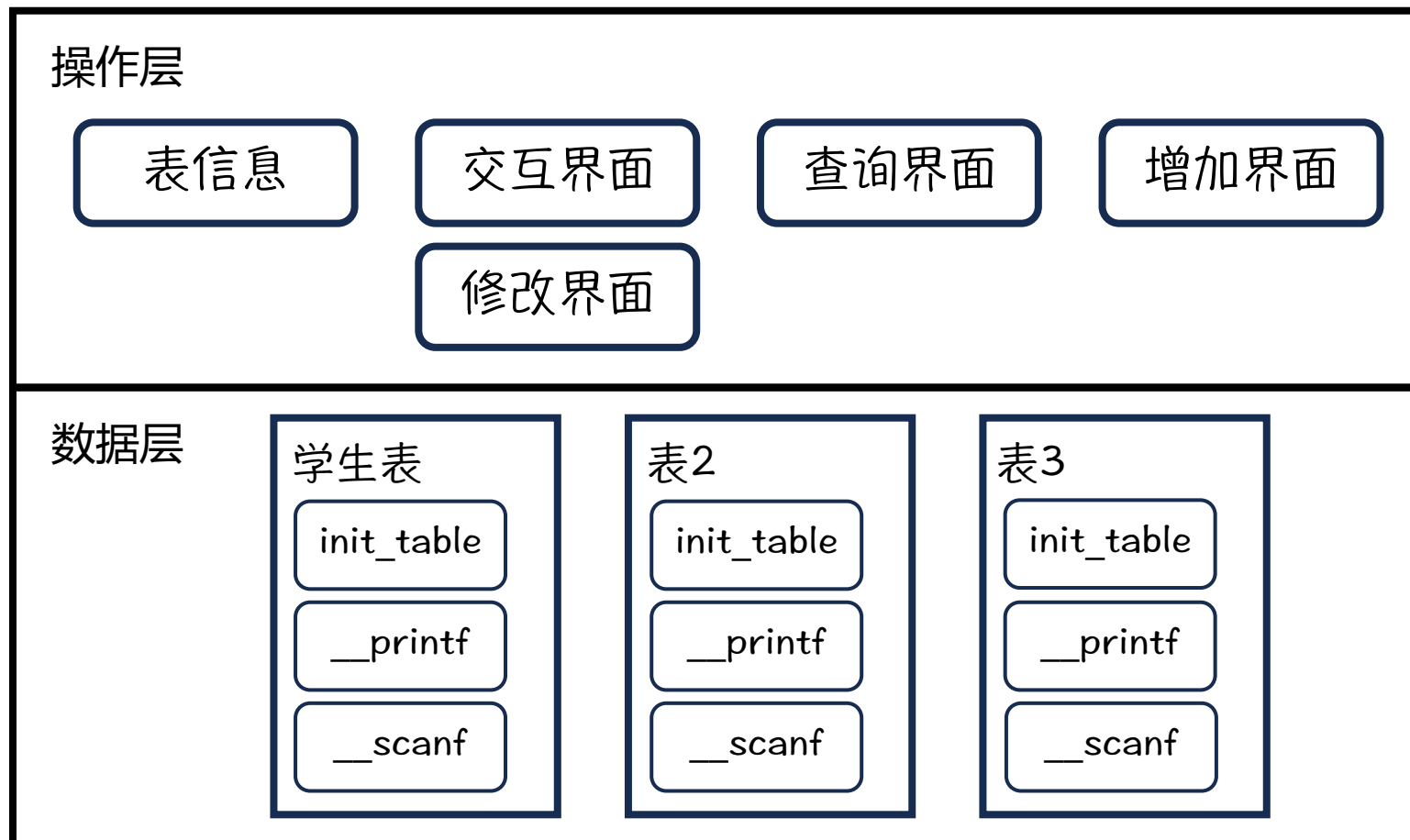
阶段6：实现数据的增删改查

1. 项目实施3：查询
2. 项目实施4：增加
3. 项目实施5：修改
4. 项目实施6：删除

项目实施5：修改



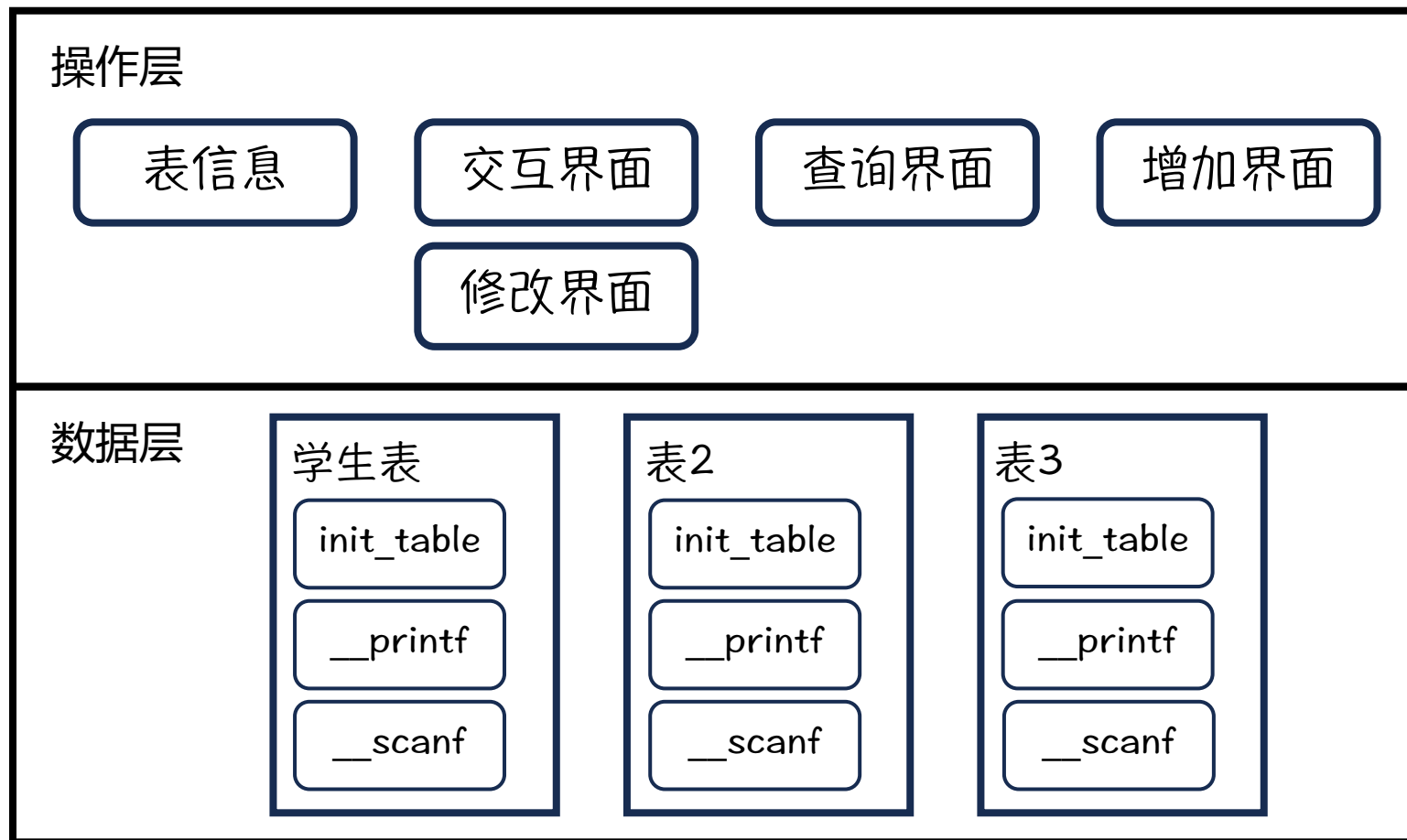
项目实施5：修改



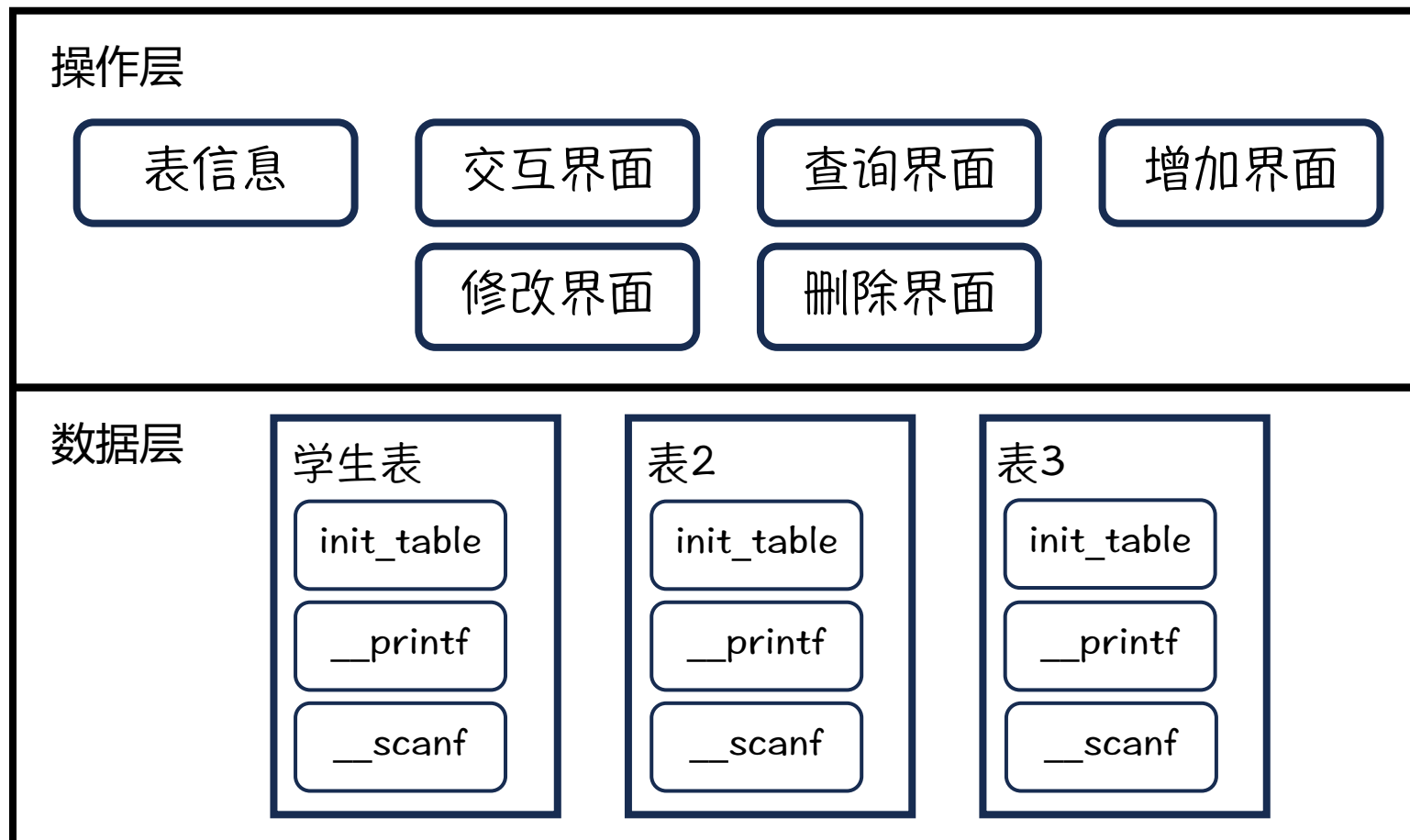
阶段6：实现数据的增删改查

1. 项目实施3：查询
2. 项目实施4：增加
3. 项目实施5：修改
4. 项目实施6：删除

项目实施6：删除



项目实施6：删除



项目测试

1. 增加多表功能测试
2. 交互流程功能测试
3. 切换多表功能测试

不要考虑太多，坚持看完，
你就已经超过了95%的人。

5. 整型数据类型



| 3.58万次播放

54. 主函数参数



| 2892次播放