第 13 讲进阶作业指导

一、任务分析：

1. 本进阶任务就是要产生一个 n 行 Xm 列的二维数据表如表-1 所示，其中每列的类型是相同的。

表-1

第 1 行	第 1 列	第 2 列	……	第 m 列
第 2 行				
……				
第 n 行				

2. 每列的类型和长度由一个一维数据表确定如表-2 所示，且这个数据表的行数为 m，列数为 1，列的类型就是结构体 FIELD_INFO，就是说表-2 的每一行都是一个 FIELD_INFO 类型的元素。

表-2

第 1 行	
第 2 行	
……	
第 m 行	

表-2 中的每行指明了表-1 中每列的类型，长度，ID 和标题。假设这个表的首地址保存在 FIELD_INFO 类型的指针中 pf，则表-1 第 1 列的 ID(在本题中并未使用 ID 也可不定义)为 pf->Field_ID 的值，第 1 列的类型由 pf->Field_Type 的值确定，第 1 列的长度由 pf->Field_Len 确定，第 1 列的标题由 pf->Field_Caption 确定，而表-1 第 2 列的的 ID(在本题中并未使用 ID 也可不定义)为(pf+1)->Field_ID 的值，第 2 列的类型由(pf+1)->Field_Type 的值确定，第 2 列的长度由(pf+1)->Field_Len 确定，第 2 列的标题由(pf+1)->Field_Caption 确定……；依此类推，表-1 中第 m 列的的 ID(在本题中并未使用 ID 也可不定义)为(pf+m)->Field_ID 的值，第 m 列的类型由(pf+m)->Field_Type 的值确定，第 m 列的由(pf+1)->Field_Len 确定；第 m 列的标题由(pf+1)->Field_Caption 确定。

在输入或输出表-1 中的每个表项时要依赖于表-2，从表-2 中查出表-1 中的元素的类型、长度和标题才能正确地输入或输出表-1 中的各个元素，而表-2 要依赖于 FIELD_INFO 结构体，通过结构体才能查出表-2 每一个元素中的各个用结构体定义的数据项，从而确定表-1 元素的类型和长度。

二、程序设计：

根据上面分析的结果可知，表-1 依赖表-2，而表-2 依赖 struct FIELD_INFO。所以程序的基本流程是：

1. 定义 struct FIELD_INFO;为了能够在申请的内存空间中更快地找到字段，可增加 int Field_offset;
2. 读入表-1 的行 row 和列 col;
3. 使用 calloc()函数申请一片内存空间大小为 sizeof(struct FIELD_INFO)xcol;(为了方便在申请的内存空间中更快地找到字段，可在 struct 中增加偏移量项 int Field_offset;用于记录

本字段距首地址 pf 的字节数，这样在表-2 查找某字段信息是，要使用 pf+Field_offset 即可找到)。

4. 输入 col 个字段信息，包括 Field_Type;Field_Len 和 Field_Caption，如果 Field_Type 为's'，应将其值+1 以考虑字符串结尾'\0'的长度。如果 Field_Type 为 n，则直接将其赋值为 4 以避免输入产生的错误。同时还要计算偏移量项的值。把 struct FIELD_INFO 修改为如下：

```
struct FIELD_INFO
{
        int Field_ID
        char Field_Type;
        int Field_Len;
        char Field_Caption[11];
        int Field_offset;
}
```

5. 的所有字段信息(表-2)输入完成后，统计所有字段的长度 RecLen，并使用 malloc()函数申请一片大小为 RecLenXrow 如表-1 的内存区。
6. 借助表-2 的字段信息，逐行，逐项输入表-1 各字段内的数据。
7. 借助表-2 的字段信息，逐行，逐项输出表-1 的全部数据。
8. 程序结束。