结构类型

课后练习题

1. 声明一个月份的结构体，包含一个前三个字母组成的该月名称的缩写、月份好、该月的天数，声明和使用如下：

```
#include<stdio.h>
#define NAME_LEN 4;

struct Month{
    char monthName[NAME_LEN];
    int monthIndex;
    int daysNumber;
}

int main(void){
    Month.monthName = "Feb";
    Month.monthIndex = 2;
    Month.daysNumber = 28;
    return 0;
}
```

(1) 指出上述程序段中的所有错误

宏定义多分号；结构体声明漏分号；结构体标识符和结构体变量混淆；字符串不能直接用=赋值，用 strcpy();

在修改的基础上，完成以下两问：

(2) 写一个函数，给定年份，生成包含该年 12 个月份的结构体数组。函数原型如下：

```
void createMonthsOfYear(int year, Month *months);
```

其中 year 为给定年份，months 为 12 个月份结构体的数组指针。

(3) 写一个函数，给定月份，计算某一年中到该月为止（包括该月）总共的天数。函数原型如下：

```
int calDaysBeforeMonth(int month, Month *months);
```

其中 month 为给定月份，months 为 12 个月份结构体的数组指针，返回值为总共天数。

参考程序：

```
#include<stdio.h>
#include<string.h>
#define NAME_LEN 4

struct Month{
    char monthName[NAME_LEN];
    int monthIndex;
    int daysNumber;
};

void createMonthsOfYear(int year, Month *months);
int calDaysBeforeMonth(int month, Month *months);
int isLeapYear(int year);

int main(void){
    Month months[12];
    createMonthsOfYear(2011, months);
    printf("%d\n", calDaysBeforeMonth(2, months));
    return 0;
}

void createMonthsOfYear(int year, Month *months){
    char monthNames[12][4] = {"Jan", "Feb", "Mar",
        "Apr", "May", "Jun", "Jul", "Aug", "Sep",
        "Oct", "Nov", "Dec"};
    int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    for(int i = 0; i < 12; i++){
        strcpy(months[i].monthName, monthNames[i]);
        months[i].monthIndex = i + 1;
        if(i==2 && isLeapYear(year))
            days[i]++;
        months[i].daysNumber = days[i];
    }
}

int calDaysBeforeMonth(int month, Month *months){
    int days = 0;
    for(int i = 0; i < 12; i++){
        if(months[i].monthIndex <= month)
            days += months[i].daysNumber;
    }
    return days;
}
```

```c
//判断是否为闰年
int isLeapYear(int year) {
    if ((year%4==0 && year%100!=0) || year %400==0 ) return 1;
    else return 0;
}
```

2. 图书管理系统

(1) 声明一个结构用于表示一本书，其内容包括书名 name，作者 author，图书位置 position，以及是否可借 isAvailable；

(2) 写一个函数，从给定文件中读取所有的图书记录，函数原型如下：

```
int getAllBooks(char *libName, Book *library, int len);
```

其中，libName 为给定文件名称，library 为存放所有图书的数组指针，len 为其长度，函数返回值为图书数量。（文件示例见题后说明）

(3) 写一个函数，添加一条图书记录，由用户依次输入书名、作者、位置与是否可借的信息，将该记录添加到文件以及数据 library 中，函数原型如下：

```
int addBook(char *libName, Book *library, int count, int len);
```

其中，libName 为给定文件名称，library 为存放所有图书的数组指针，len 为长度，count 为现有图书数量，函数返回是否添加成功。

(4) 写一个函数，根据书名查询图书：遍历图书，输出书名中包含给定名称字符段的所有图书，输出信息包括书名、作者、位置与是否可借

```
int findBookByName(char *name, Book *library, int count);
```

其中，name 为待查询书名字符段，library 为存放所有图书的数组指针，count 为现有图书数量，函数返回查询到的图书数量。

文件示例：（文件名为 lib.txt）

中国哲学史大纲	胡适	H503B3	1
西方政治哲学史	王岩	W415	0
中国哲学史	任继愈	B2-09	1
希腊哲学史	汪子嵩	B502	0

参考程序：

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

#define MAX_BOOK_NUM 1000
#define TITLE_LEN 41
#define AUTHOR_LEN 31
#define POSTION_LEN 41

struct Book{
    char title[TITLE_LEN];
    char author[AUTHOR_LEN];
    char position[POSTION_LEN];
    int isAvailable;
};

int getAllBooks(char *libName, Book *library, int len);
int addBook(char *libName, Book *library, int count, int len);
int findBookByName(char *name, Book *library, int count);

int main(void){
    Book library[MAX_BOOK_NUM];
    int count = getAllBooks("lib.txt", library, MAX_BOOK_NUM);
    addBook("lib.txt", library, count, MAX_BOOK_NUM);
    printf("%d\n",count);
    char fileName[TITLE_LEN];
    printf("Plz enter the book's name: \n");
    scanf("%s", fileName);
    findBookByName(fileName,library, count);
    return 0;
}

int getAllBooks(char *libName, Book *library, int len){
    FILE *libFile = fopen(libName, "rt");
    if(libFile == NULL)
        return 0;
    int count = 0;
    while(!feof(libFile) && count < len){
        fscanf(libFile, "%s%s%s%d", library[count].title,
            library[count].author, library[count].position,
            &library[count].isAvailable);
        count++;
    }
    fclose(libFile);
    return count;
}
```

```c
int addBook(char *libName, Book *library, int count, int len){
    if(count == len){
        printf("The library is full!\n");
        return count;
    }
    printf("Plz enter the book's name: \n");
    scanf("%s", library[count].title);
    printf("Plz enter the book's author: \n");
    scanf("%s", library[count].author);
    printf("Plz enter the book's position: \n");
    scanf("%s", library[count].position);
    library[count].isAvailable = 1;

    FILE *libFile = fopen(libName, "at");
    if(libFile == NULL){
        printf("cannot find the library!\n");
        return count;
    }
    fprintf(libFile, "\n%s\t%s\t%s\t%d", library[count].title,
        library[count].author, library[count].position,
        library[count].isAvailable);
    printf("new book added\n");
    fclose(libFile);
    return ++count;
}

int findBookByName(char *name, Book *library, int count){
    Book book;
    int number = 0;
    for(int i = 0; i < count; i++){
        book = library[i];
        if(strstr(book.title, name) != NULL){
            number++;
            printf("%s\t%s\t%s\t%d\n", book.title, book.author,
                book.position, book.isAvailable);
        }
    }
    if(number == 0)
        printf("cannot find this book!\n");
    return number;
}
```