函数与多文件补遗

头文件中可以放的内容：

头文件包含
类型声明
宏定义
声明变量
函数声明

关键字

变量声明与定义的区别：

变量必须先声明
    声明可以被多次声明
    声明不开辟空间
后定义
    定义只能定义一次
    定义会开辟空间
再使用
只有在头文件中使用**`int a;`**这种形式的才算声明

静态变量

`static int a;`
定义变量的时候，前面加关键字**`static`**则变为静态变量
生命周期变长，整个程序存续期间都存在；
上一次调用的值能够保存下来
初始化语句只运行一次；
多文件编程时，**`static`**修饰的变量，只能在当前文件使用，其他文件不能使用

外部变量

`extern`
声明外部变量，告诉编译器，变量声明在其他文件

暂时忽略未定义的错误，一般用在全局变量上

const 常量

将变量变为常量，变量的值不能被修改：
靠近谁，修饰谁

```
const char * p = &ch;
*p = 'd';       //f
p = &ch1;       //t
// 指向的值不能改变
char const *p1 = &ch;
// 指向的值不能改变
char * const p2 = &ch;
// 指向的地址不能改变
*p = 'd';       //t
p = &ch1;       //f
const char * const p3 = &ch;
// 都不能变
```

结构体

C 数组允许定义可存储相同类型数据项的变量
结构是 C 编程中另一种用户自定义的可用的数据类型
它允许存储不同类型的数据项。

声明结构体类型：

```
struct stuff{
        char job[20];
        int age;
        float height;
};
```

定义结构体变量：

```
struct stuff yourname;
```

访问结构成员：

用成员访问运算符（.）
yourname.job;
yourname.age;

指向结构体的指针：

结构体也是数据类型的一种，因此可以定义指针指向结构体变量：
struct stuff * p = & yourname;
使用指针访问结构体成员：
p->job;
p->age;

结构体作为函数参数:

可以把结构作为函数参数，传参方式与其他类型的变量或指针类似
不要改变值，传变量
要改变值，传指针
实例（不改变值）：

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

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* 函数声明 */
void printBook( struct Books book );
int main( )
{
   struct Books Book1;        /* 声明 Book1，类型为 Books */
   struct Books Book2;        /* 声明 Book2，类型为 Books */

   /* Book1 详述 */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* Book2 详述 */
```

```
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* 输出 Book1 信息 */
   printBook( Book1 );

   /* 输出 Book2 信息 */
   printBook( Book2 );

   return 0;
}
void printBook( struct Books book )
{
   printf( "Book title : %s\n", book.title);
   printf( "Book author : %s\n", book.author);
   printf( "Book subject : %s\n", book.subject);
   printf( "Book book_id : %d\n", book.book_id);
}
```

实例2（传指针，不改变值）：

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

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* 函数声明 */
void printBook( struct Books *book );
int main( )
{
   struct Books Book1;        /* 声明 Book1，类型为 Books */
   struct Books Book2;        /* 声明 Book2，类型为 Books */

   /* Book1 详述 */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;
```

```
   /* Book2 详述 */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* 通过传 Book1 的地址来输出 Book1 信息 */
   printBook( &Book1 );

   /* 通过传 Book2 的地址来输出 Book2 信息 */
   printBook( &Book2 );

   return 0;
}
void printBook( struct Books *book )
{
   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
```

实例3（传指针，改变值）：

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

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* 函数声明 */
void printBook( struct Books *book );
void changeBook(struct * Books * book);
int main( )
{
   struct Books Book1;        /* 声明 Book1, 类型为 Books */
   struct Books Book2;        /* 声明 Book2, 类型为 Books */

   /* Book1 详述 */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
```

```
   Book1.book_id = 6495407;

   /* Book2 详述 */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* 通过传 Book1 的地址来输出 Book1 信息 */
   printBook( &Book1 );

   /* 通过传 Book2 的地址来输出 Book2 信息 */
   printBook( &Book2 );
   changeBook(&Book1);
   printBook( &Book1 );
   return 0;
}
void printBook( struct Books *book )
{
   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
void changeBook(struct * Books * book)
{
    strcpy(book->title,"no title");
    strcpy(book->author,"no author");
    strcpy(book->subject,"no subject");
    book->book_id = 0;
}
```