文件操作

函数库：

函数名	函数声明	头文件	函数功能	参数列表和返回值说明
fopen	FILE *fopen(const char *Filename, const char *Mode);	stdio.h	打开文件	打开文件，第一个参数是文件名，第二个参数为文件的打开模式。注①
fclose	int fclose(FILE *stream);	stdio.h	关闭文件	关闭文件并刷新文件缓冲区
fflush	int fflush(FILE *stream);	stdio.h	刷新文件缓冲区	刷新文件缓冲区
fscanf	int fscanf(FILE *stream, const char *format, [argument...]);	stdio.h	文件内容的标准格式化输入	用法参照scanf，第一个参数加入一个文件指针，表示使用该文件的缓冲区进行读取。
fprintf	int fprintf(FILE *stream, const char *format, [argument...]);	stdio.h	文件内容的标准格式化输出	用法参照printf，第一个参数加入一个文件指针，表示使用该文件的缓冲区进行输出。
sscanf	int sscanf(char *string, const char *format, [argument...]);	stdio.h	字符串内容的标准格式化输入	用法参照scanf，第一个参数加入一个字符串，表示使用该字符串作为缓冲区进行读取。
sprintf	int sprintf(char *string, const char *format, [argument...]);	stdio.h	字符串内容的标准格式化输出	用法参照printf，第一个参数加入一个字符串，表示使用该字符串作为缓冲区进行输出。
fgets	int fgets(char *buffer, size_t size, FILE *stream);	stdio.h	按行读取文件	按行读取文件，每次读取到换行或者文件末尾结束。文件读取完毕返回EOF。
fputs	int fputs(char *buffer, FILE *stream);	stdio.h	按行向文件打印	按行输出，参考puts，输出到第二个参数的文件流里。
fgetc	int fgetc(FILE *stream);	stdio.h	按字符读取文件	参考getchar，从文件中读取字符，完毕则按返回EOF.
fputc	int fputc(char ch, FILE *stream);	stdio.h	按字符向文件打印	参考putchar，向文件中打印字符。
fread	size_t fread(void *buffer, size_t size, size_t count, FILE *stream);	stdio.h	按原结构从文件中读取	按照二进制排列结构读取文件。第一个参数用来读取数据，二三参数相乘表示长度，最后一个是文件流。
fwrite	size_t fwrite(const void *buffer, size_t size, size_t count, FILE *stream);	stdio.h	按原结构向文件中写入	按照二进制排列结构写入文件。第一个参数是数据来源，二三参数相乘表示长度，最后一个是文件流
rewind	void rewind(FILE *stream);	stdio.h	重置文件指针	重置文件指针至文件开头。
fseek	int fseek(FILE *stream, int offset, int fromwhere);	stdio.h	按照需求跳转文件指针	跳转文件指针，第一个参数是文件流，第二个参数是距离指定位置的偏移量，第三个参数是指定位置。注②
ftell	int ftell(FILE *stream);	stdio.h	获取文件指针位置	返回该指针所在的位置距离文件开头有多少个字节。
feof	int feof(FILE *stream);	stdio.h	判断文件是否结束	文件结束返回非0，文件未结束返回0。

注①

文件的打开模式：

r模式：只读模式，如果文件不存在则返回NULL。

w模式：只写模式，如果文件不存在则创建，如果存在则覆盖。

a模式：只写追加模式，如果文件不存在则创建，如果存在则打开并指向文件末尾。

追加格式：

b模式：二进制打开文件。

+模式：让打开的文件可读可写。

例如如果我的需求是不存在则创建，二进制写入，可读可写，那就用**wb+**

注②

fseek的位置指定宏：

SEEK_SET：值为0，表示从文件开头算起

SEEK_CUR：值为1，表示从当前位置算起

SEEK_END：值为2，表示从文件末尾算起

指针补充

常量字符串的全局性：同一个常量字符串在一个进程中只有一个备份，用char*直接去取的时候，得到的地址是同一个位置。这也解释了为什么常量字符串的内容是在静态区的。

数组指针的特殊性：数组指针其实是指向数组的指针，可以用“取址符+数组名”的形式得到。但指向了数组的指针实际是指向了数组整体，它的位置和原数组相同，区别在于它的一个偏移量是整体数组的长度。

函数指针

函数指针是一种非常特殊的指针，它可以赋值，却不是真正的指针。

函数指针的定义：将函数声明中的标识符用括号括起来，前面加个星号，就是函数指针的定义。

※只要参数列表和返回值有一点不同，函数指针的类型就是不同的。

※针对函数指针的所有解引用操作都是无效的。

※函数名就是天然的函数指针。

函数指针的本质：函数指针其实是一个“假指针”，可以视为代码的入口，后面就叫入口指针。在赋值的时候操作的是这个入口指针的值。当你调用这个入口指针的时候，就会根据该函数的代码生成一个栈帧空间，进行执行，所以入口指针和实际栈所在的空间是不同的。

二级函数指针：对函数指针取地址以后，可以生成二级函数指针，必须经过一次解引用后才可以调用。

函数指针数组：函数指针数组一般用来将相似的函数放在一起，方便通过下标直接调用。

复杂类型的解读方式和定义方式

解读方式：

1、寻找优先级最高的部分开始解读（先右后左，先近后远）

优先级最高的如果是*，则是个指针，如果是[]则是个数组，如果是()，则是一个函数声明。

2、将标识符和优先级最高的部分结合成整体后继续

结合规则：跟*结合一起去掉，跟[]结合则连[]里的数字一起去掉，跟()结合则跟()里的内容一起去掉。

由于()最多只能有一个是以类型情况出现的，所以当标识符和()一起去掉时，剩下的就是返回值类型。

3、重复前两步直到成为简单类型。

※如果存在多个标识符，则其中只有一个是真的，剩下的是参数的变量名。处理方法是分别处理每一个，如果在处理过程中某个标识符已经没有作用于它的运算符，则说明这个标识符是参数的变量名，可以删掉后重新处理，也可以把已经处理的部分当做整体。

定义方式：

1、先寻找句子的宾语，完成宾语的类型

2、去掉句子的宾语。

3、重复前两步直到宾语已经不再有定语。

以下是一个例子。

```
int (*(*fun(int*(*p)(int *)))[5]) (int*)

针对这句话，有两个类似标识符的，即fun和p
随便找一个开刀。如果p是标识符，那么跟标识符p联系最紧密的是*。
所以p是一个指针。什么指针？再把*p看成一个整体tmp：

int (*(*fun(int*tmp(int *)))[5]) (int*)
```

此时跟tmp最亲的是()，说明这是个函数指针。找到返回值和参数后，int*tmp(int *)作为一个整体pfunc继续处理：

```
int (*(*fun(pfunc))[5]) (int*)
```

好了，pfunc已经没有作用于它的单目运算符了，所以已经确定不是标识符了，那么标识符是fun。
已经处理掉的参数可以当做一个整体，这个函数的参数是一个函数指针，类型是int *(*)(int *)

因为跟fun结合最紧密的是小括号，所以确定，这是一个函数声明。
把函数名和参数列表一起去掉，就是它的返回值类型，所以返回值类型是：

```
int (*(*)[5]) (int*)
```

添加了一个p作为标识符来寻找p的类型

```
int (*(*p)[5]) (int*)
```

那么p是什么类型呢？首先结合最紧密的是*，所以p是指针，什么指针？合起来看：

```
int (*tmp[5]) (int*)
```

是个数组指针，什么样的数组？

```
int (*atmp) (int*)
```

是个函数指针数组。所以函数的返回值是个函数指针数组指针。

好了可以得到结论了：

这是一个函数声明，这个函数有一个参数列表为一个int*，返回值为int*的函数指针作为参数，返回值是一个指针，这个指针指向一个5个元素的数组，数组里的每个元素是一个参数列表为一个int*，返回值类型为int的函数指针。

利用这个语言来还原出定义：
宾语是函数声明，所以：

```
func()
```

将其分为参数和返回值两半来看，先看参数的句子：
参数列表为一个int*，返回值为int*的函数指针作为参数。
无需拆解，没有复杂的嵌套，非常清晰：

```
func(int *(*p)(int *)) ----结果1
```

返回值类型比较复杂：
是一个指针，这个指针指向一个5个元素的数组，数组里的每个元素是一个参数列表为一个int*，返回值类型为int的函数指针。

返回值是个指针：

```
*p
```

去掉指针，剩下的句子：
一个5个元素的数组，数组里的每个元素是一个参数列表为一个int*，返回值类型为int的函数指针。

宾语是数组，顺手把5个元素处理了：

```
(*p)[5] ----结果2
```

好，再去掉数组，看下一级：
是一个参数列表为一个int*，返回值类型为int的函数指针。

数组里是函数指针，先把函数指针定义出来：

```
int (*pfunc)(int *)
```

到现在各个部分就都有了。然后就是替换掉标识符的事了：
用刚刚写好的结果2替换掉临时标识符func：

```
int (*(*p)[5])(int *)
```

这就是返回值类型。上面还有函数名和参数列表：
将写好的结果1替换掉临时标识符p：

```
int (*(*func(int *(*p)(int *)))[5])(int *)
```

完成。