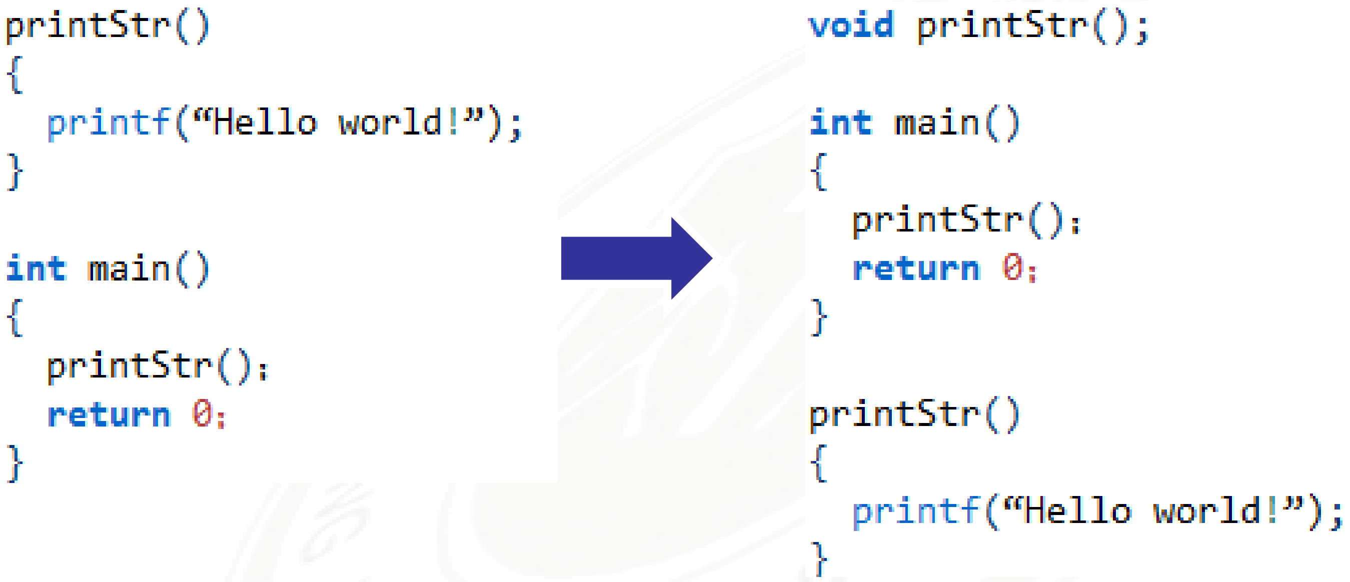
库文件的编写

从函数的作用域说起

函数的作用域

- 从声明点延伸到源程序文本结束

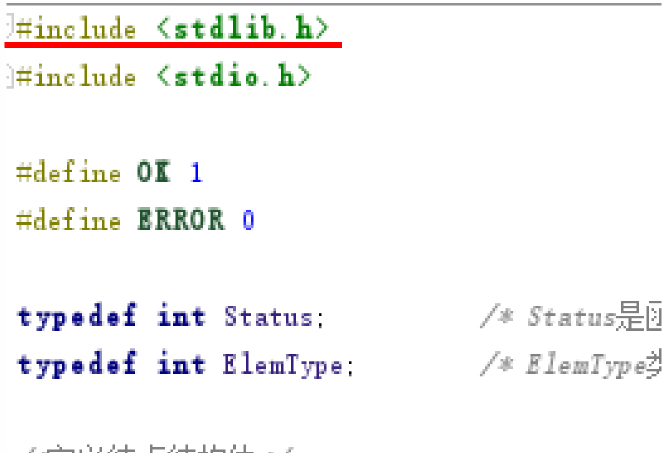


```
printStr()
{
  printf(“Hello world!”);
}

int main()
{
  printStr();
  return 0;
}
```

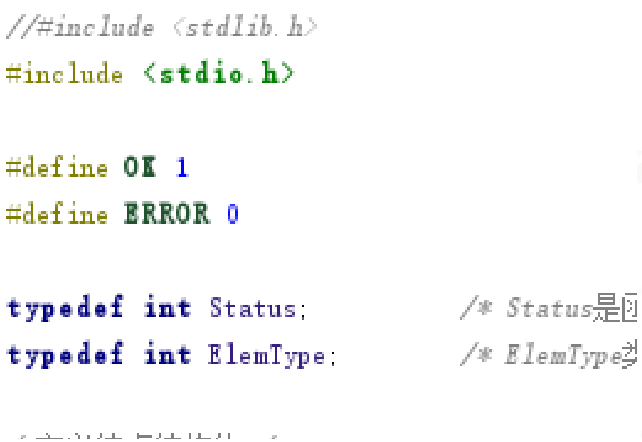
```
void printStr();

int main()
{
  printStr();
  return 0;
}

printStr()
{
  printf(“Hello world!”);
}
```

- 全局变量、全局结构体也是如此。

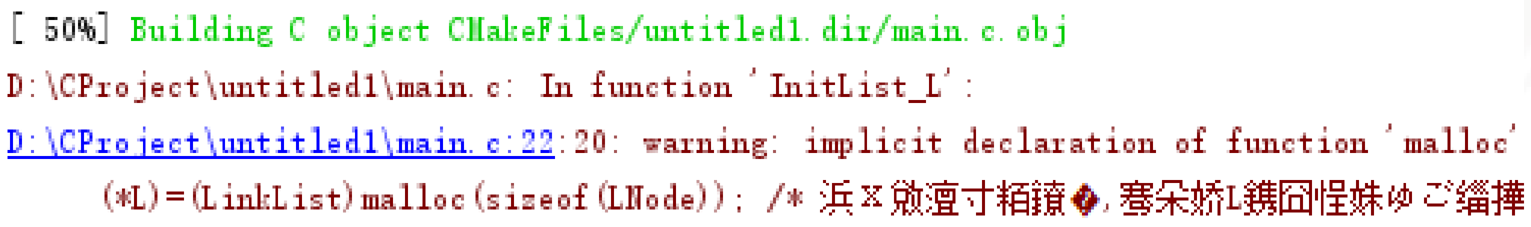
再来说说库文件



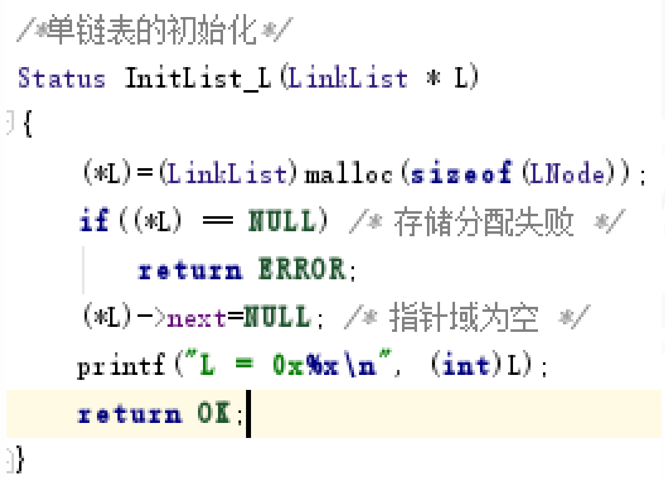
```
#include <stdlib.h>
#include <stdio.h>

#define OK 1
#define ERROR 0

typedef int Status;          /* Status是
typedef int ElemType;        /* ElemType
```



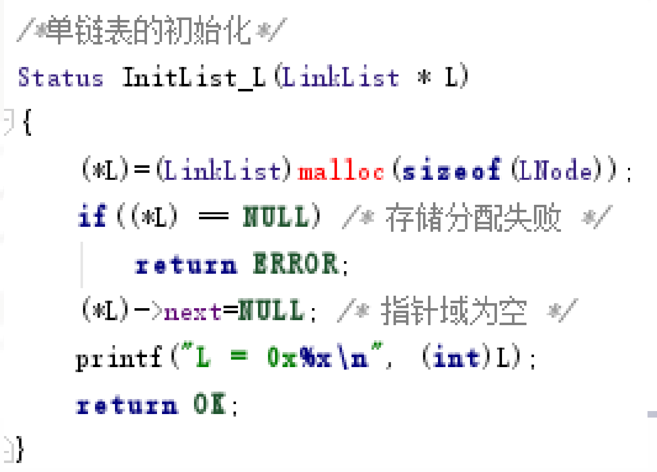
```
//#include <stdlib.h>
#include <stdio.h>

#define OK 1
#define ERROR 0

typedef int Status;          /* Status是
typedef int ElemType;        /* ElemType
```



```
[ 50%] Building C object CMakeFiles/untitled1.dir/main.c.obj
D:\CProject\untitled1\main.c: In function 'InitList_L':
D:\CProject\untitled1\main.c:22:20: warning: implicit declaration of function 'malloc'
     (*L)=(LinkList)malloc(sizeof(LNode)); /* 浜х敓澶寸粨鐐�,骞朵娇L鎸囧悜姝ゅご缁撶
```



```
/*单链表的初始化*/
Status InitList_L(LinkList * L)
{
    (*L)=(LinkList)malloc(sizeof(LNode));
    if((*L) == NULL) /* 存储分配失败 */
        return ERROR;
    (*L)->next=NULL; /* 指针域为空 */
    printf("L = 0x%x\n", (int)L);
    return OK;
}
```



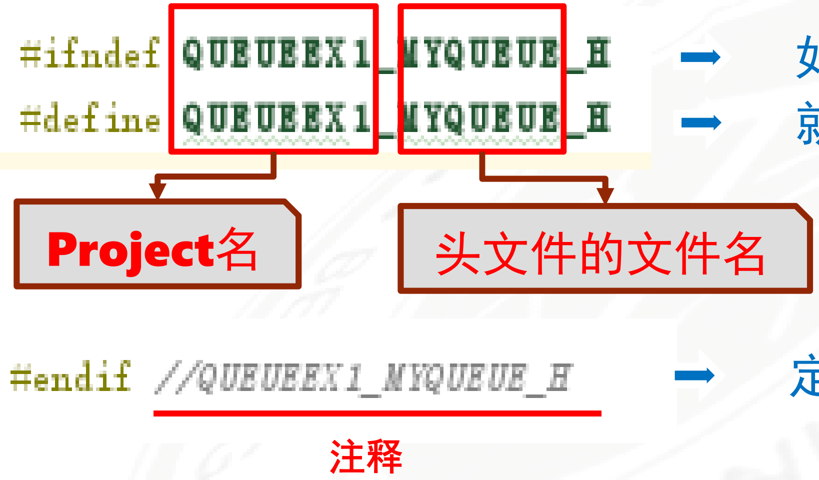
```
/*单链表的初始化*/
Status InitList_L(LinkList * L)
{
    (*L)=(LinkList)malloc(sizeof(LNode));
    if((*L) == NULL) /* 存储分配失败 */
        return ERROR;
    (*L)->next=NULL; /* 指针域为空 */
    printf("L = 0x%x\n", (int)L);
    return OK;
}
```

再来说说库文件

- 是否可以猜测库文件的内容？
  - 库文件应该包括函数的定义和声明；
  - 库文件还可能包含宏定义、全局变量定义和全局结构体的生命和定义。
- 如果有一个库包含队列的基本操作
  - 那么只需要include这个库，就可以直接调用这些函数。
- 能否自己编写库文件/头文件？
  - 库文件：函数的定义
  - 头文件：函数的申明
  - 库文件通过头文件向外导出接口。用户通过头文件找到库文件中的函数定义

库文件的编写

- 包括一个.h文件（头文件）和一个或多个.c文件
- 头文件有固定的格式
  - Clion自动生成头文件的固定格式。



```
#ifndef QUEUEEX1_MYQUEUE_H
#define QUEUEEX1_MYQUEUE_H

#endif //QUEUEEX1_MYQUEUE_H
```

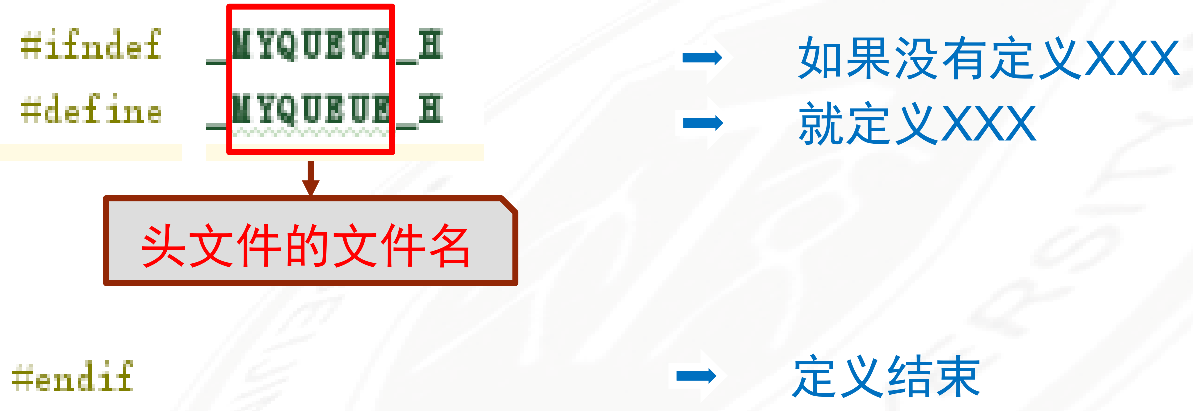
➡ 如果没有定义XXX

➡ 就定义XXX

➡ 定义结束

库文件的编写

- 包括一个.h文件（头文件）和一个**或多个**.c文件
- 头文件有固定的格式
  - VC6不能自动生成，但格式类似。



```
#ifndef  _MYQUEUE_H
#define  _MYQUEUE_H


#endif
```

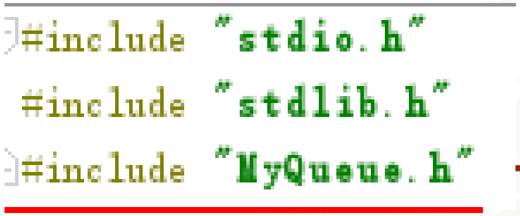
头文件的文件名

➡ 如果没有定义XXX
➡ 就定义XXX
➡ 定义结束

- 头文件的内容
  - 宏定义
  - 全局变量声明和定义
  - 全局结构体声明和定义
  - 函数声明

库文件的编写

- 包括一个.h文件（头文件）和一个**或多个**.c文件
- .c文件的格式要求
  - 文件名必须与.h文件名相同。
  - 必须包含同名头文件



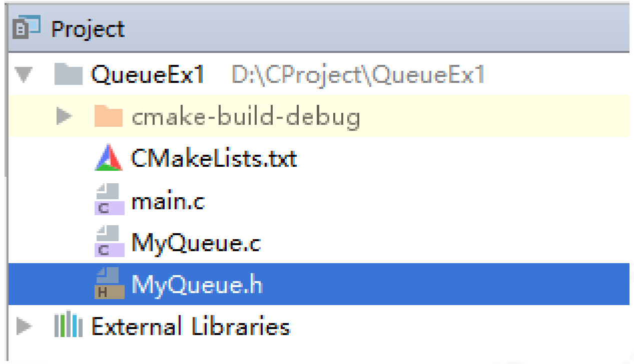
```
#include "stdio.h"
#include "stdlib.h"
#include "MyQueue.h"
```

同名头文件

  - 没有main()函数

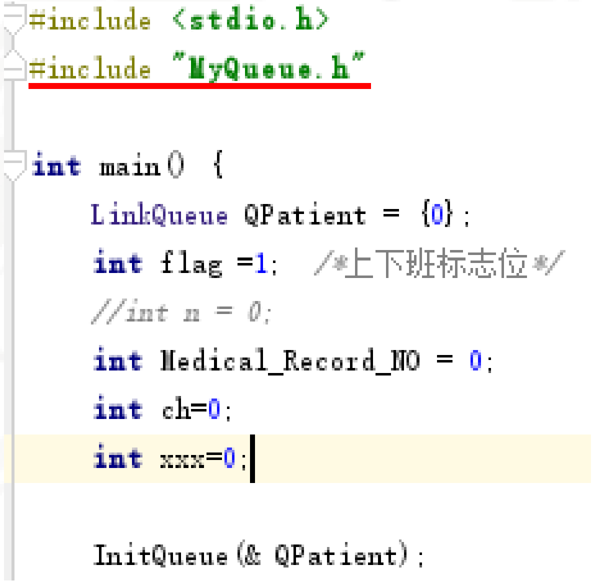
库文件的使用

- 与main()函数放入同一工程目录



```
Project
▼ QueueEx1   D:\CProject\QueueEx1
  ▶ cmake-build-debug
    CMakeLists.txt
    main.c
    MyQueue.c
    MyQueue.h
▶ External Libraries
```

- 包含自己定义的头文件



```c
#include <stdio.h>
#include "MyQueue.h"

int main() {
    LinkQueue QPatient = {0};
    int flag =1;  /*上下班标志位*/
    //int n = 0;
    int Medical_Record_NO = 0;
    int ch=0;
    int xxx=0;

    InitQueue(& QPatient);
```

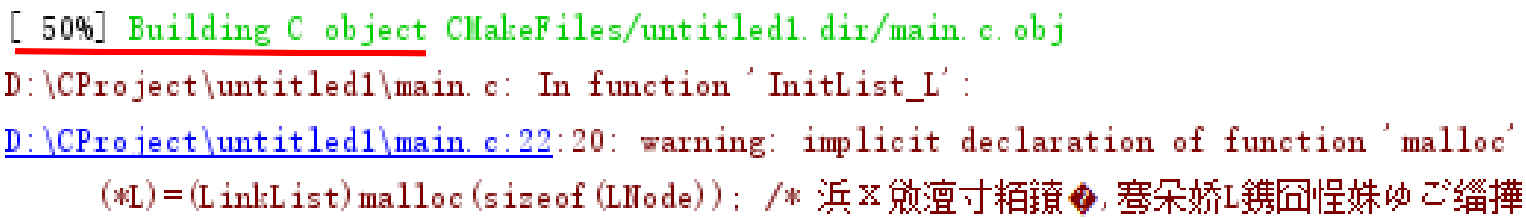
关于编译（compile）和链接（Build）

- 一个C程序分为若干程序文件，每个程序文件单独编

compile只生成.obj文件。

build先compile出.obj文件，然后link出.exe文件。

- 最后 和变量符 文件



```
[ 50%] Building C object CMakeFiles/untitled1.dir/main.c.obj
D:\CProject\untitled1\main.c: In function 'InitList_L':
D:\CProject\untitled1\main.c:22:20: warning: implicit declaration of function 'malloc'
         (*L)=(LinkList)malloc(sizeof(LNode)); /* 浜ㄨ敓潰寸粨鐨�,蹇朵娇L鎸囧悜鐨ゅご缁撴
```

  - 如
那些模块，用到了其中的变量或函数，要把那些目标文件链接进来，才能正确编译。
  - 为了解决这个问题，C程序引入了库的概念，所谓库就是一组已经编译好的目标文件集合，链接的时候只要把这个库文件指示给链接程序，链接程序会自动从文件中查找符号要求的函数和变量进行链接，整个查找过程根本不需要我们担心。