

# 程序设计综合实践

信息科学与技术学院

“计算机类”及“电子信息类”平台课程

信息学院程序设计课程教学团队

# 程序设计综合实践

## 小型管理信息系统

多源文件程序设计

程序设计课程教学团队

# 预备知识——预处理与文件包含

C 语言提供三种预处理功能：

- 1.宏定义；
- 2.文件包含；
- 3.条件编译。

分别用宏定义命令、文件包含命令和条件编译命令来实现。这些命令以符号“#”开头，结尾没有分号(;)。该命令的作用域是从定义的地方开始到源文件结束。

在 C 编译系统对程序进行通常的编译之前，先对程序中这些命令进行“预处理”。然后将结果和源程序一起再进行编译处理，最后得到目标代码。

# 1. 宏定义

## (1) 不带参数的宏定义

功能：用一个指定的标识符(宏名)来代表一个字符串。

一般形式：**#define 标识符 任意字符串序列**

例如6.1:

```
#define PI 3.1415926
```

```
main()
```

```
{float l,s,r,v;
```

```
printf("input radius:");
```

```
scanf("%f",&r);
```

```
l=2.0*PI*r;
```

```
s=PI*r*r;
```

```
v=4.0/3*PI*r*r*r;
```

```
printf("l=%f\ns=%f\nv=%f\n",l,s,v);
```

```
}
```



```
main()
```

```
{float l,s,r,v;
```

```
printf("input radius:");
```

```
scanf("%f",&r);
```

```
l=2.0*3.1415926*r;
```

```
s=3.1415926*r*r;
```

```
v=4.0/3*3.1415926*r*r*r;
```

```
printf("l=%f\ns=%f\nv=%f\n",l,s,v);
```

```
}
```

说明:


1.1宏定义是用宏名代替一个字符串，也就是作简单的置换，不作语法检查。如：

```
#define PI 3.1415926p;
```

```
area=PI*r;      预处理后为      area=3.1415926p;*r
```

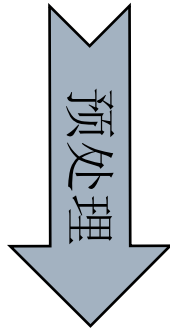
1.2宏名的有效范围为定义命令之后到本源文件结束。可以用**#undef**命令终止宏定义的作用。如：

```
#define G 9.8  
main()  
{  
:  
}  
#undef G  
f1()  
{  
:  
}
```



**1.3**在进行宏定义时，被定义过的宏名可被重新定义，也可以引用已定义的宏名。如例6.2：

```
#define PI 5.6
#define R 3.0
#define PI 3.1415926
#define L 2*PI*R
#define S PI*R*R
main()
{printf("L=%f\nS=%f\n",L,S);
}
```



用双引号括起来的字符串内的字符，即使与宏名相同，也不进行置换。如L,S

```
main()
{
    printf("L=%f\nS=%f\n",2*3.1415926*3.0,3.1415926*3.0*3.0);
}
```

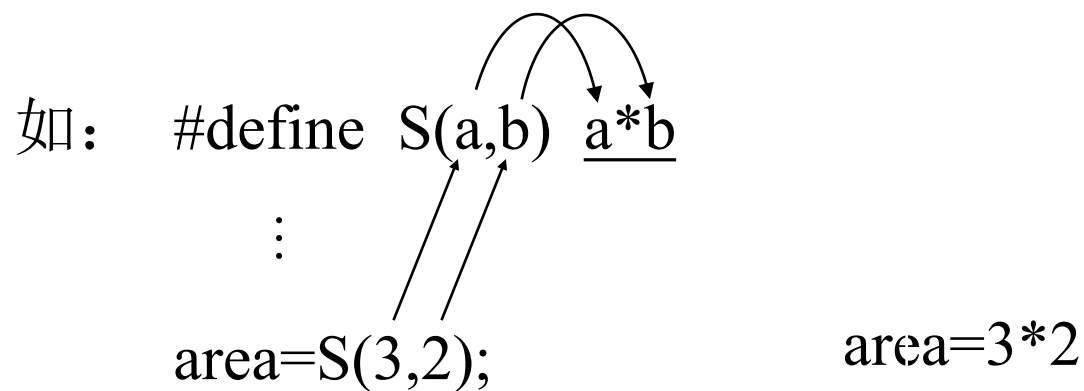
## (2) 带参数的宏定义

进行字符串替换，还要进行参数替换。

定义形式为：**#define 宏名(参数表) 字符串**

包含括弧中指定的参数

如： `#define S(a,b) a*b`  
      :  
      `area=S(3,2);`                      `area=3*2`



注意：不能写成 `#define S (a,b) a*b`

不能有空格

例:

```
#define PI 3.1415926
```

```
#define S(r) PI*r*r
```

```
main()
```

```
{float a,area;
```

```
  a=3.6;
```

```
  area=S(a);
```

```
  printf("r=%f\narea=%f\n",a,area);
```

```
}
```

预处理

```
main()
```

```
{float a,area;
```

```
  a=3.6;
```

```
  area=3.1415926*a*a;
```

```
  printf("r=%f\narea=%f\n",a,area);
```

```
}
```

如果有以下语句,

```
area=S(x+y);
```

预处理后  
结果为?

```
area=3.1415926*x+y*x+y;
```

```
#define S(r) PI*(r)*(r)
```

```
area=3.1415926*(x+y)*(x+y);
```

宏定义中的圆括号  
——绝对需要

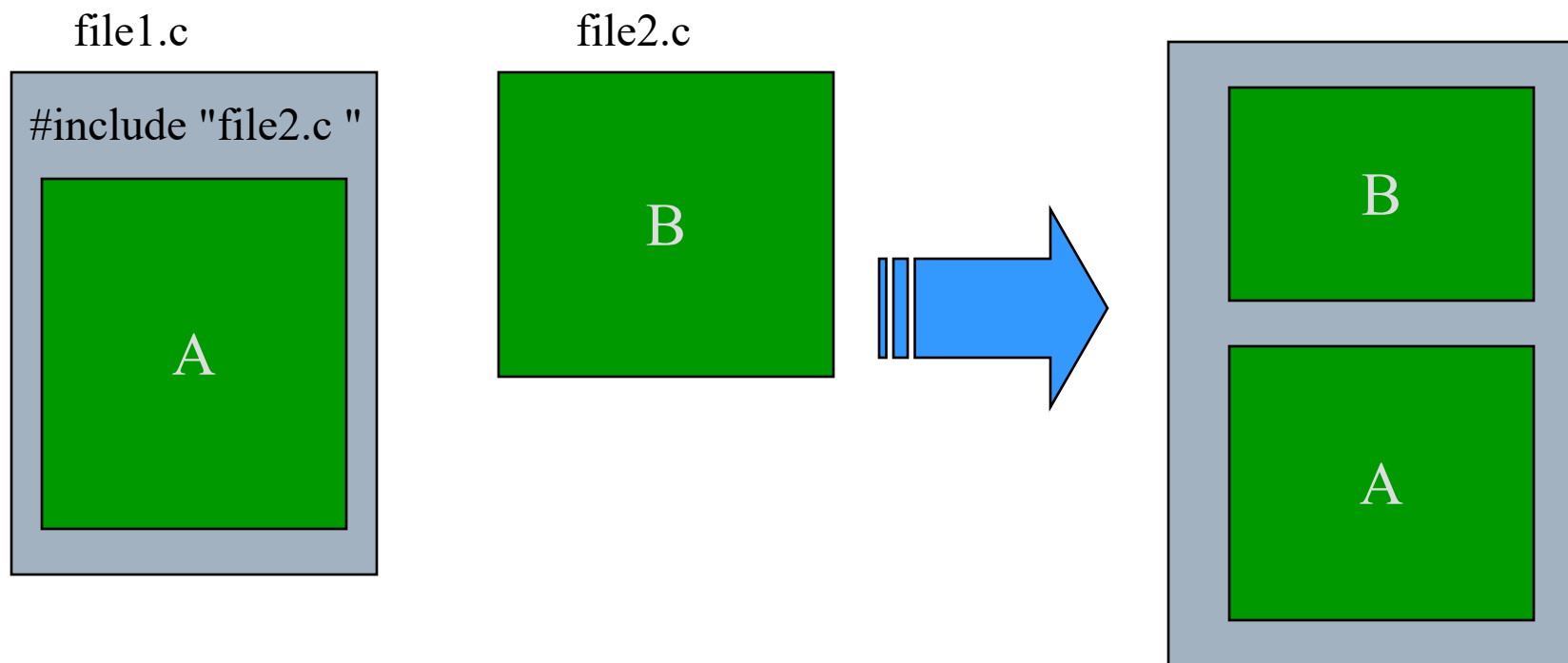


## 2. “文件包含” 处理

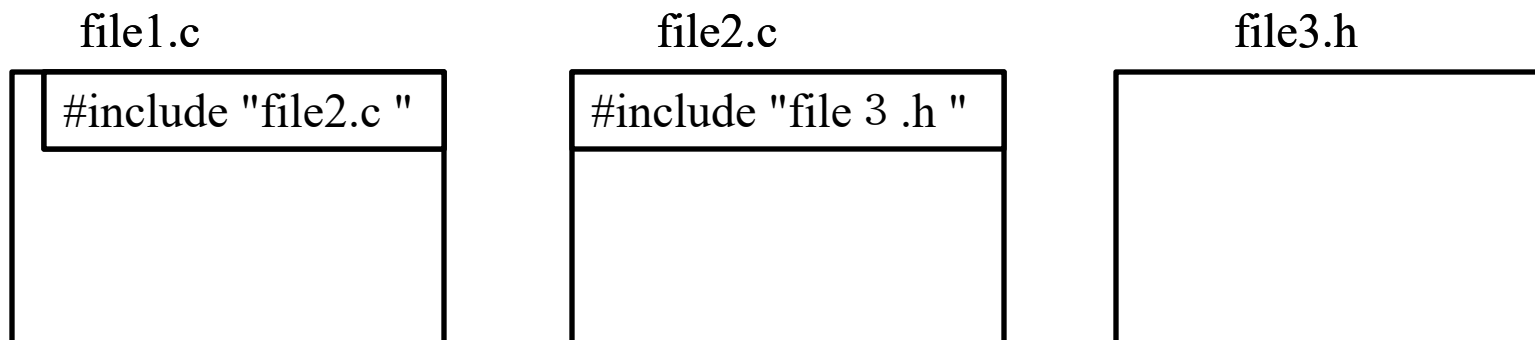
文件包含处理是指一个源文件可以将另一个源文件的全部内容包含进来。

一般形式为：`#include` “文件名”

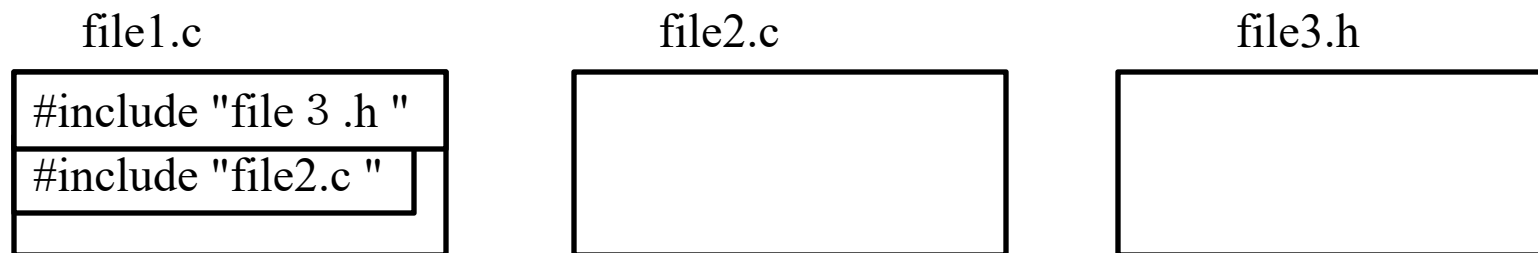
( “ ” 和 `< >` 指查找路径不同 )



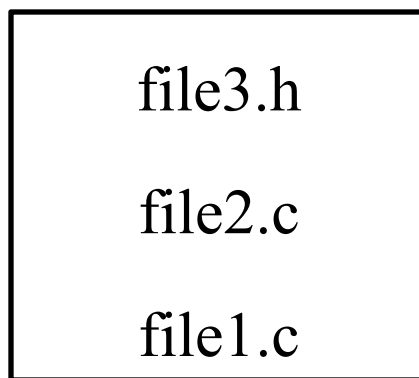
文件包含是可以嵌套的：



也可按如下方式：



编译后  
文件1为



file3中的全局变量，file2、file1可直接使用，不必用extern说明。

另：一个#include只能包含一个文件。

### 3. 条件编译（标准头文件结构常用）

条件编译是指让编译程序根据条件有选择地对源程序的一部分进行编译。

C 语言提供三组条件编译命令，其形式如下：

★ #ifdef 标识符

程序段1

#else

程序段2

#endif

若标识符存在编译  
程序段1，否则编译  
程序段2

★ #ifndef 标识符

程序段1

#else

程序段2

#endif

若标识符不存在编译  
程序段1，否则编译  
程序段2



`#if` 表达式  
程序段1

`#else`  
程序段2

`#endif`

若表达式为真编译  
程序段1，否则编译  
程序段2

例:6.4 `#define LETTER 1`  
`main()`  
`{char str[20]="C Language",c;`  
`int i=0;`  
`while ((c=str[i])!='\0')`  
`{i++;`  
`#if LETTER`  
`if (c>='a'&&c<='z')`  
`c=c-32;`  
`#else`  
`if (c>='A'&&c<='Z')`  
`c=c+32`  
`#endif`  
`printf("%c",c);`  
`}}`

预处理后

`main()`  
`{char str[20]="C Language",c;`  
`int i=0;`  
`while ((c=str[i])!='\0')`  
`{i++;`  
`if (c>='a'&&c<='z')`  
`c=c-32;`  
`printf("%c",c);`  
`}`  
`}`

# 管理信息系统对多源文件程序设计的需求

**为什么需要多源文件：**

**思考：**

- **main() 里的代码太长了适合分成几个函数**
- **一个源代码文件太长了适合分成几个文件**
- **两个独立的源代码文件不能编译形成可执行的程序**

# 管理信息系统对多源文件程序设计的需求

## □ 多人开发，需要多个程序源文件

- 独立设计和编辑
- 独立编译、调试、测试
- 团队开发、联合调试和测试

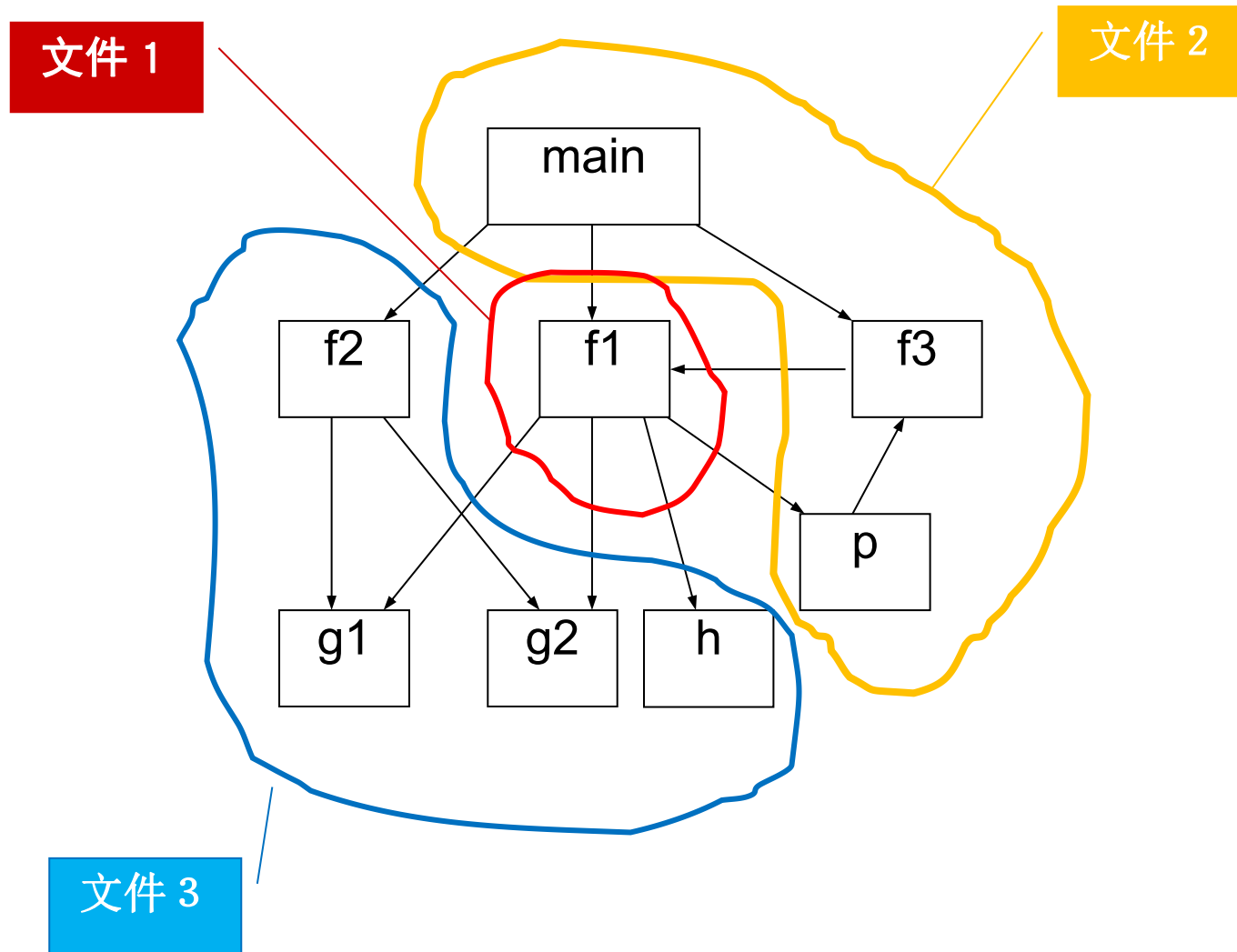
## □ 多源文件程序设计先修知识

- 变量的作用域和生存期
- 编译预处理
- 如忘记可看上学期的参考课件

# 多源文件的程序结构

- ❑ 一个工程包含有多个源文件
- ❑ 每个源文件包含若干个函数（只有一个main）
- ❑ 每个源文件包含若干个变量（局部、全局）
- ❑ 不同源文件的函数之间可以互相调用
- ❑ 使用头文件支持函数调用和隐藏

# 多源文件程序的结构示例






# 多源文件程序的结构

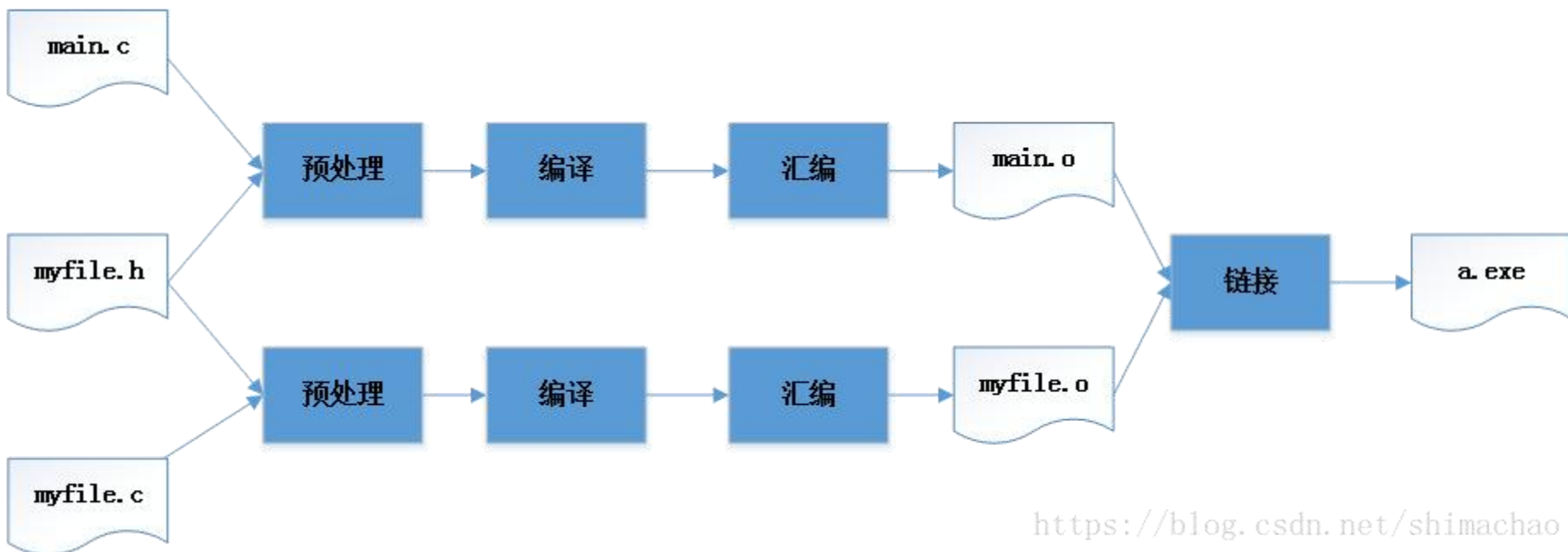
## C 程序中多源文件的程序包括:

- ❑ 头文件(header files) .h 或 .hpp
- ❑ 定义文件(definition files).c 或 .cpp
- ❑ 项目 (或工程) 组织文件 .dev
  - ✓ .layout ---自动生成的布局文件
  - ✓ Makefile.win--自动生成的编译配置文件
  - ✓ .o---经过编译之后的目标文件
  - ✓ .exe 可执行文件, 双击直接运行, 不同操作系统不能混用



- ComputeAndSort.h
- ConstDefine.h
- FileRelated.h
- FindByQuery.h
- InfoManagement.h
- SelectMenu.h
- ComputeAndSort.cpp
- FileRelated.cpp
- FindByQuery.cpp
- InfoManagement.cpp
- SelectMenu.cpp
- StudentMain.cpp
- StudentScore.dev
- StudentScore.layout
- ComputeAndSort.o
- FileRelated.o
- FindByQuery.o
- InfoManagement.o
- SelectMenu.o
- StudentMain.o
- Makefile.win
- studentlist.txt
- StudentScore.exe

# 多源文件程序的结构-可执行文件生成



- ❑ **预处理：**宏替换、文件包含等事务的预先处理
- ❑ **编译：**将预处理后的源文件转换成汇编代码
- ❑ **汇编：**将上一阶段生成的汇编代码编译成二进制中间文件.o
- ❑ **链接：**将中间文件链接到一起，生成可执行文件.exe

# 多源文件程序的结构

## 1. 头文件 (.h 或 .hpp文件)

- ❑ 作为一种包含功能函数、数据接口声明的载体文件，用于保存程序的**声明**(只有声明可以被放在头文件中)
- ❑ 对程序所需的数据类型、常量、全局变量进行声明
- ❑ 对各个被调用函数进行声明
  - ❑ 头文件不包含程序的逻辑实现代码，只起描述性作用，告诉应用程序通过相应途径寻找相应功能函数的真正逻辑实现代码（定义文件）

```
#ifndef FINDQUERY
#define FINDQUERY
// 函数的声明 */
#include "ConstDefine.h"
/* 查询相关算法 */
/* 按学号查找 */
void FindByNumber(struct student stu[], char szCourseName[][STRLENGTH], int nStudentNum) ;
/* 按姓名查找 */
void FindByName(struct student stu[], char szCourseName[][STRLENGTH], int nStudentNum) ;
////////////////////////////////////
#endif
```

# 多源文件程序的结构

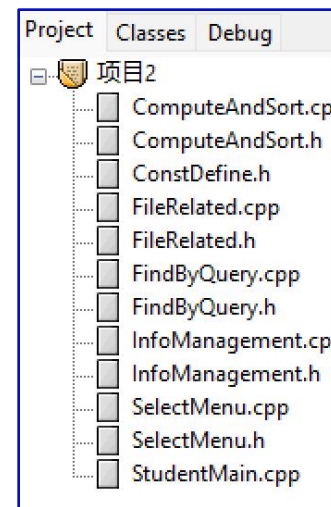
## 2. 定义文件 (.c 或 .cpp文件)

- ❑ 编译器从库中提取相应的实现代码
- ❑ 用于保存程序的实现的具体细节
- ❑ 包括数据结构、算法等具体实现代码

```
int main()
{
    //定义学生结构体数组
    struct student stu[STUDENTNUM];
    int nStudentNum=0; //存储学生个数
    //定义课程名称
    char szCourseName[COURSENUM][STRLength]={"语文",
    //改变背景前景色
    system("color 5e");
    //打开学生数据文件
    FILE * fp=fopen(FILENAME,"r");
    if(fp==NULL) //若数据文件不存在, 新建数据文件
```

## 3. 工程或项目文件

- ❑ 各文件之间传递参数和控制
- ❑ 把各个文件链接一个可执行的整体



# 多源文件程序的结构-头文件

## 声明和定义

### 声明

- 函数原型
- 变量声明
- 结构声明
- 宏声明
- 枚举声明
- 类型声明
- inline函数

### 定义

- 函数
- 全局变量

### 全局变量与全局变量的声明

- `int i;`是变量的定义
- `extern int i;`是变量的声明

# 多源文件程序的结构-头文件

## #include

`#include`是一个编译预处理指令，和宏一样，在编译之前就处理了

- 它把那个文件的全部文本内容原封不动地插入到它所在的地方
- 所以也不是一定要在.c文件的最前面#include

“” 还是<>?

“” 要求编译器首先在当前目录（.c文件所在的目录）寻找这个文件，如果没有，到编译器指定的目录去找

<>让编译器只在指定的目录去找

- 编译器自己知道自己的标准库的头文件在哪里
- 环境变量和编译器命令行参数也可以指定寻找头文件的目录

# 多源文件程序的结构-头文件

## 标准头文件结构 —— 避免重复声明

同一个编译单元里，同名的结构不能被重复声明

- 如果你的头文件里有结构的声明，很难这个头文件不会在一个编译单元里被#include多次

- 所以需要“标准头文件结构”

“标准头文件”

运用条件编译和宏，保证这个头文件在一个编译单元中只会被#include一次

```
#ifndef __ADD_H  
#define __ADD_H
```

```
int Add(int num1,int  
num2);  
struct stu
```

```
{
```

```
    int num;  
    char name[10];
```

```
};
```

```
#endif
```

# 多源文件程序的结构-头文件

## 1. 原始头文件

作为共同开发的项目，为了共享彼此的过程资源（函数），将全体函数声明放在一个共用的头文件中

```
// abc.h
typedef struct Student
{ char name[10];
  float score;
}Stu;
void f1 ();
void f2 ();
void f3 ();
void g1 ();
void g2 ();
void p ();
void h ();
```

```
// a1.cpp
#include "abc.h"
void f1()
{
    if(...){
        p();
        g1(); }
    else{
        g2();
        h(); }
}
```

```
// a2.cpp
#include "abc.h"
int main(){
    f1();
    f2();
    f3();
}
void f3(){
    f1();
}
void p(){
    f3();
}
```



# 多源文件程序的结构-头文件

## 2. 界面头文件

提供给他人使用的调用资源声明（函数，数据，类型等）

```
// a1.h  
void f1();
```

```
// a2.h  
void p();
```

```
// a3.h  
void g1();  
void g2();  
void f2();  
void h();
```

```
// a1.cpp  
#include "a1.h"  
#include "a2.h"  
#include "a3.h"  
void f1()  
{  
    if(...) {  
        p();  
        g1();  
    } else {  
        g2();  
        h();  
    }  
}
```

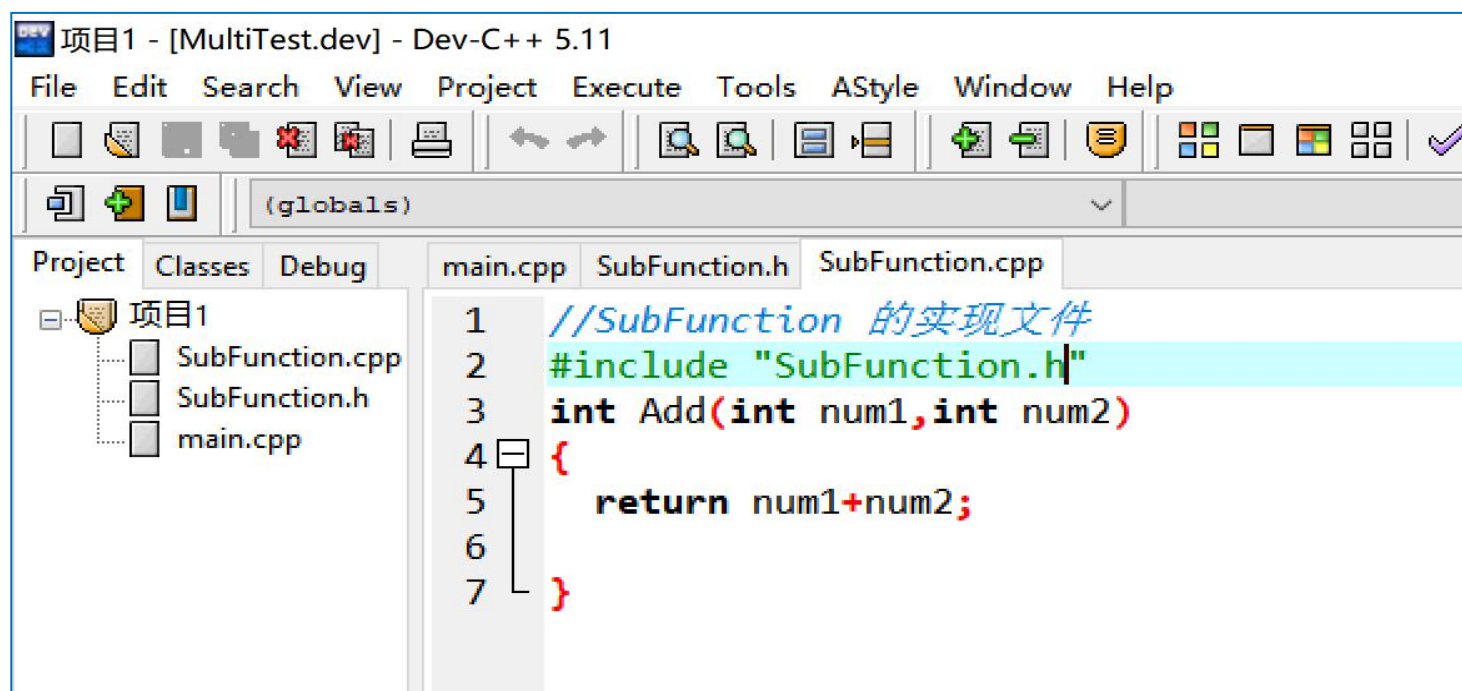
```
// a2.cpp  
#include "a2.h"  
#include "a1.h"  
#include "a3.h"  
static void  
    f3();  
int main() {  
    f1();  
    f2();  
    f3();  
}  
void f3() {  
    f1();  
}  
void p() {  
    f3();  
}
```

```
// a3.cpp  
#include "a3.h"  
void h() { ... }  
void f2()  
{  
    g1();  
    g2();  
}  
void g1() { ... }  
void g2() { ... }
```

# 多源文件程序的结构-课堂练习

## 1. 观察、编译、运行实例：MultiFilesTest

“项目1”是我们创建的第一个多文件程序。里面有3个文件其中有1个头文件（界面头文件），两个定义文件。



**未来自己创建项目的时候，给项目起适当的名字**

# 多源文件程序的结构-课堂练习

## 2. 给已有的源文件里添加函数

照着“加”函数的例子，给MultiFiles 项目里添加“减”函数，分别添加到SubFunction.h 和SubFunction.cpp中

```
pp  [*] SubFunction.h  SubFunction.cpp
//加函数，求两个整数的和
int Add(int,int);
int Subtract(int,int);
```

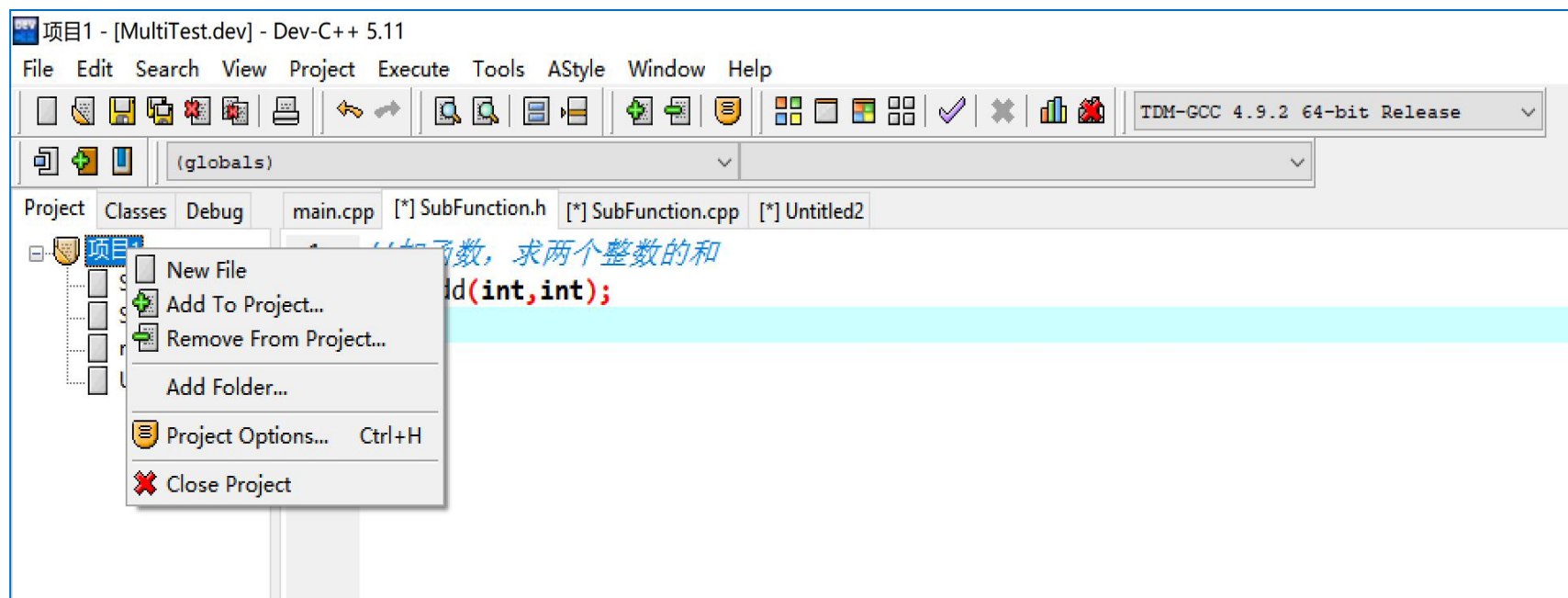
```
cpp  [*] SubFunction.h  [*] SubFunction.cpp
//SubFunction 的实现文件
#include "SubFunction.h"
int Add(int num1,int num2)
{
    return num1+num2;
}

int Subtract(int num1,int num2)
{
    return num1-num2;
}
```

# 多源文件程序的结构-课堂练习

## 3. 添加Times.h 和Times.cpp, 里面放“乘”函数

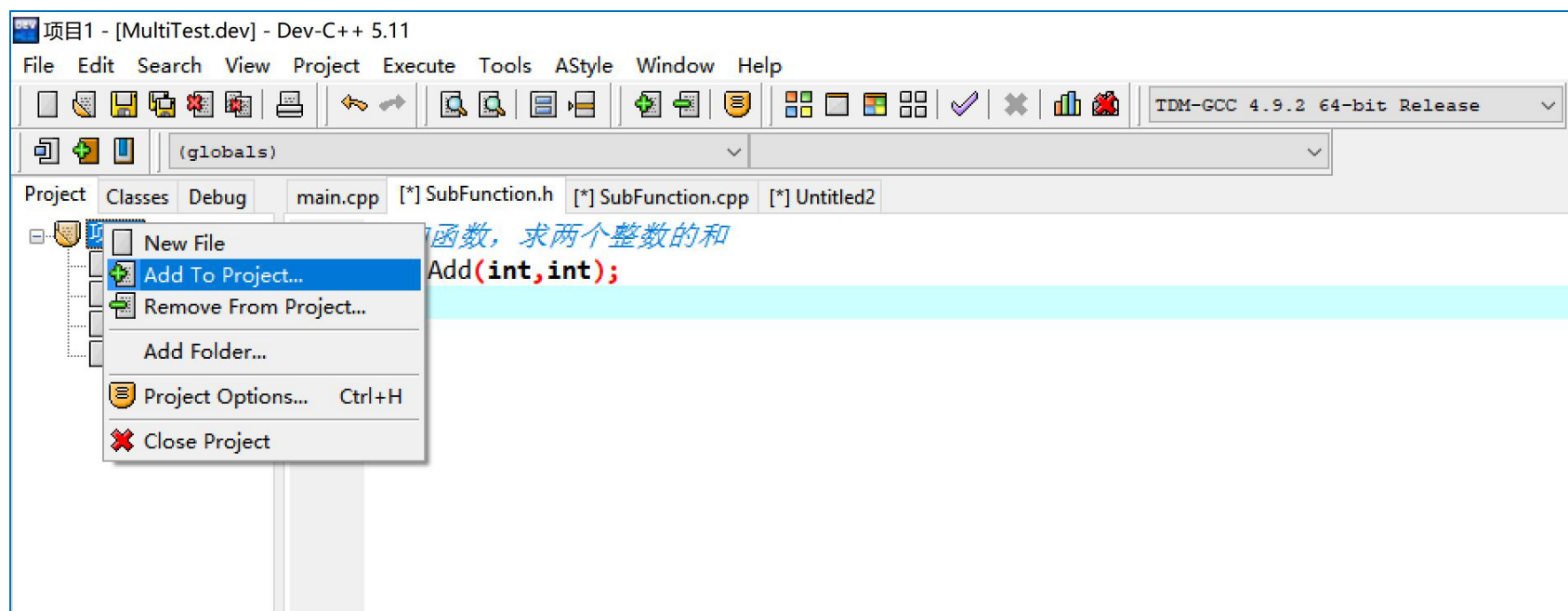
点击项目1，按鼠标右键，选择“New File（新文件）”  
给新文件起名（保存）为Times.h/ Times.cpp



# 多源文件程序的结构-课堂练习

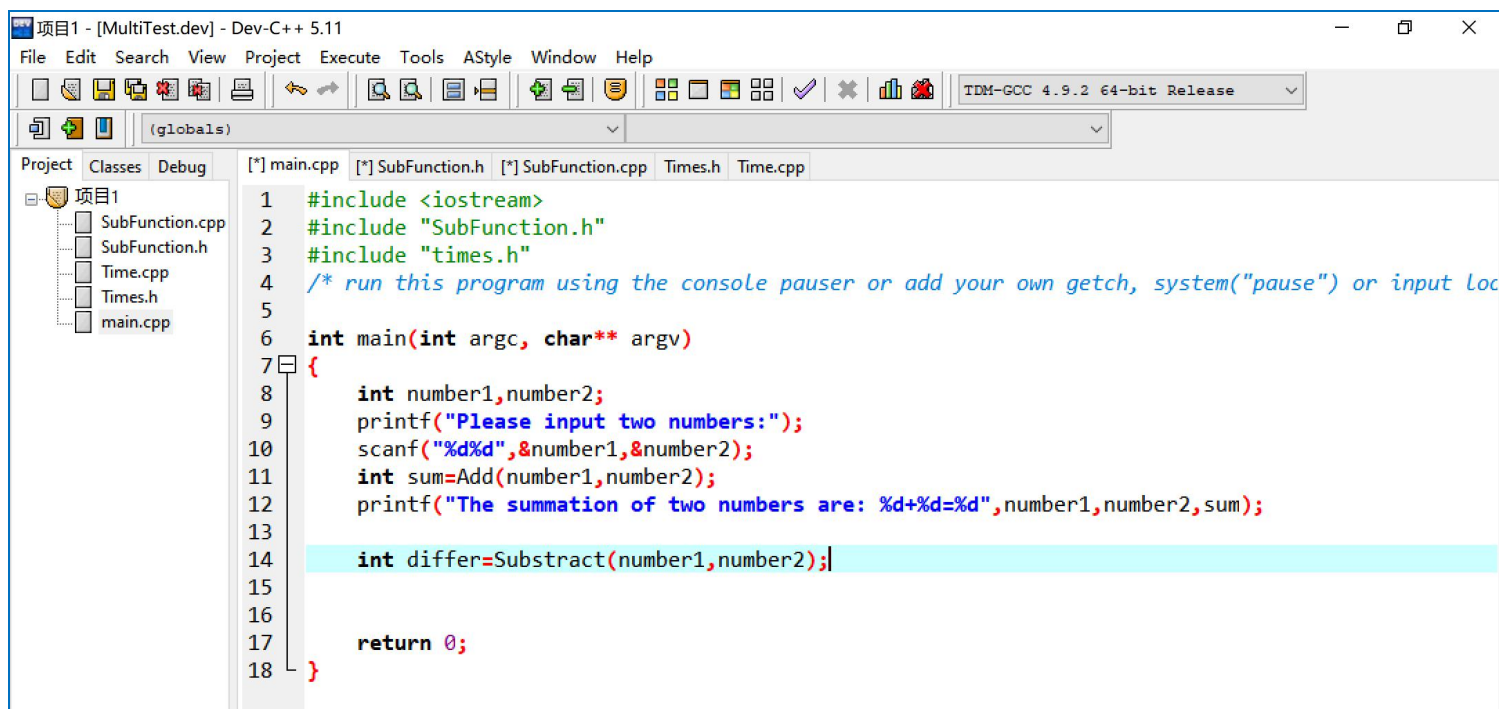
## 3. 添加Times.h 和Times.cpp, 里面放“乘”函数

或在刚才的文件夹下，复制SubFunction.h 和 SubFunction.cpp 到同一文件夹下，改名为Times.h和 Times.cpp，再用“Add to Project（添加到项目）”将两个文件添加到该项目中。



# 多源文件程序的结构-课堂练习

## 3. 添加Times.h 和Times.cpp, 里面放“乘”函数 在main.cpp中一并调用



```
1 #include <iostream>
2 #include "SubFunction.h"
3 #include "times.h"
4 /* run this program using the console pauser or add your own getch, system("pause") or input loc
5
6 int main(int argc, char** argv)
7 {
8     int number1,number2;
9     printf("Please input two numbers:");
10    scanf("%d%d",&number1,&number2);
11    int sum=Add(number1,number2);
12    printf("The summation of two numbers are: %d+%d=%d",number1,number2,sum);
13
14    int differ=Subtract(number1,number2);|
15
16
17    return 0;
18 }
```

# 第二周课后作业

## 1. PTA互评作业

### C 程序设计综合实践实验：多源文件程序设计

#### 一、实验目的

1. 学会多源文件程序的组织结构和方法，体会文件包含含义
2. 学会宏定义和条件编译方法
3. 练习变量的作用域和生存期

## 2. 完成分组

- ☐ 6-7人一组将组队完成一个管理信息系统作品
- ☐ 必须都在同一个选课班
- ☐ 选一位组长，起一个组名