



预处理、头文件卫士



- 作用：对源程序编译之前做一些处理

- 种类

- 宏定义 **#define**
- 文件包含 **#include**
- 条件编译 **#if--#else--#endif**等

- 格式：

- “#”开头
- 占单独书写行
- 语句尾不加分号



宏定义

■ 种类

- ☐ 带参数的宏
- ☐ 不带参数的宏

■ **#define PI 3.1415926**

■ 宏代换

■ 宏展开

不带参数宏定义

- 一般形式: **#define** 宏名 [宏体]
- 功能: 用指定标识符(宏名)代替字符序列(宏体)

例如

```
#define PI 3.14159
```

```
.....
```

```
printf(“%f\n”,PI*2);
```

宏展开: **printf(“%f\n”,3.14159*2);**

说明:


- 作用域:从定义命令到文件结束
- **#undef**可终止宏名作用域

格式: **#undef** 宏名

例 **#define** **YES** **1**
 void **main**()
 {
 }
 #undef **YES**
 #define **YES** **0**
 max()
 {.....
 }


YES原作用域

YES新作用域

- 
- 宏名在源程序中若用引号括起来，则预处理程序不对其作宏代换。

例

```
#define OK 100  
void main()  
{  
    printf("OK");  
    printf("\n");  
}
```

- 
- 宏定义允许嵌套，在宏定义的字符串中可以使用已经定义的宏名。在宏展开时由预处理程序层层代换。

例如：

```
#define PI 3.1415926
```

```
#define S PI*y*y      /* PI是已定义的宏名*/
```

对语句：

```
printf("%f",S);
```

在宏代换后变为：

```
printf("%f",3.1415926*y*y);
```



带参数宏定义

- C 语言允许宏带有参数。在宏定义中的参数称为形式参数，在宏展开中的参数称为实际参数。
- 对带参数的宏，在代换过程中，不仅要宏展开，而且要用实参去代换形参。

- 一般形式: **#define** 宏名(参数表) 宏体

例 **#define S(a,b) a*b**

.....

area=S(3,2);

宏展开: **area=3*2;**

不能加空格

- 宏体及各形参外一般应加括号 ()
- 宏展开: 形参用实参换, 其它字符保留



例如:

#define M(y) y*y+3*y /*宏定义*/

.....

k=M(5); /*宏展开*/

.....



例


```
#define MAX(a,b) (a>b)?a:b  
void main()  
{  
    int x,y,max;  
    cout<<"input two numbers:  ";  
    cin>>x>>y;  
    max=MAX(x,y);  
    cout<<"max="<<max<<endl;  
}
```



例 用宏定义和函数实现同样的功能

```
#define MAX(x,y) (x)>(y)?(x):(y)
.....
void main()
{ int a,b,c,d,t;
  .....
  t=MAX(a+b,c+d);
  .....
}
```

```
int max(int x,int y)
{ return(x>y?x:y);
}
void main()
{ int a,b,c,d,t;
  .....
  t=max(a+b,c+d);
  .....
}
```



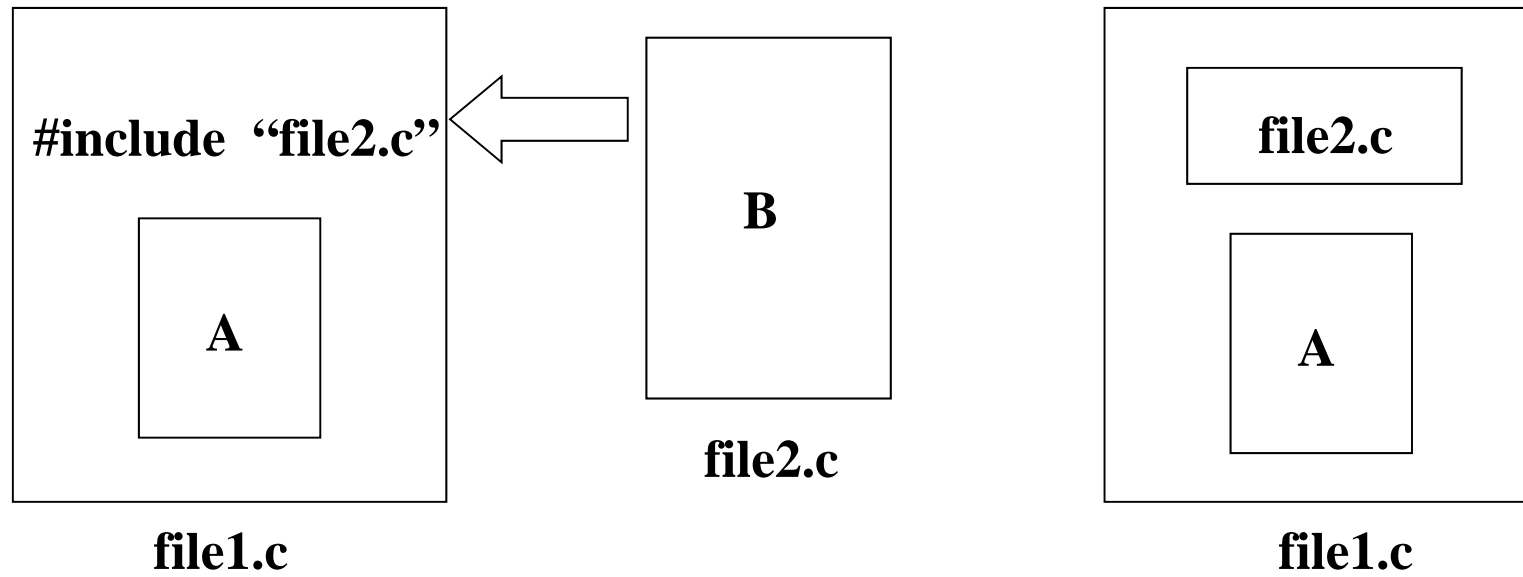
■ 带参的宏与函数区别

| | 带参宏 | 函数 |
|------|------------------|---------------------|
| 处理时间 | 编译时 | 程序运行时 |
| 参数类型 | 无类型问题 | 定义实参,形参类型 |
| 处理过程 | 不分配内存 简单的字符置换 | 分配内存 先求实参值,再代入形参 |
| 程序长度 | 变长 | 不变 |
| 运行速度 | 不占运行时间 | 调用和返回占时间 |



文件包含

- 功能：一个源文件可将另一个源文件的内容全部包含进来。
- 一般形式：**#include** “文件名”
或 **#include** <文件名>
- 处理过程：预编译时,用被包含文件的内容取代该预处理命令，再对“包含”后的文件作一个源文件编译。



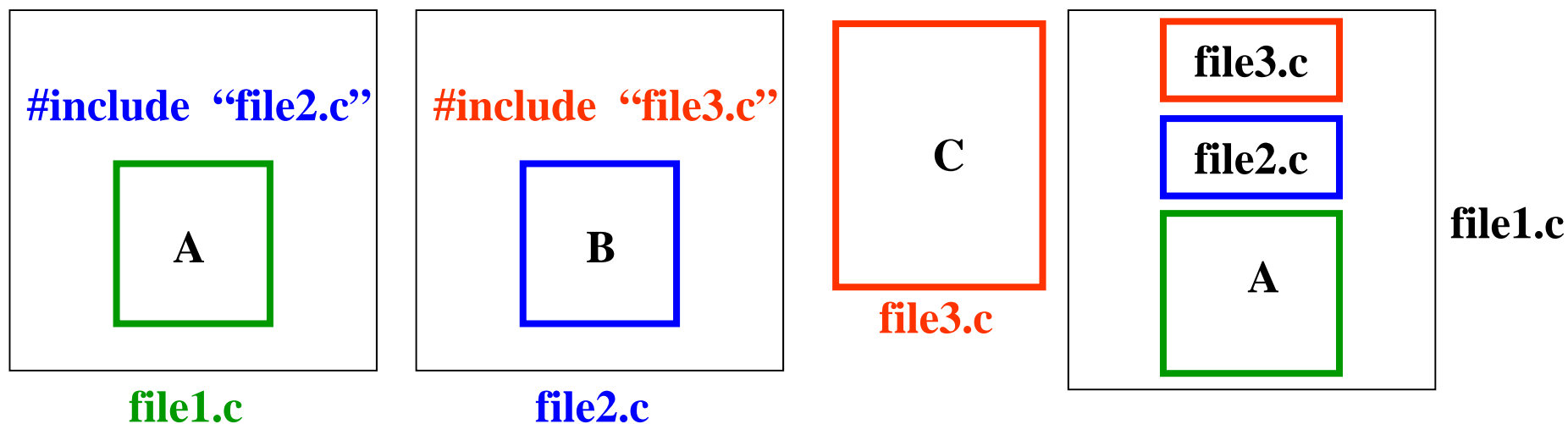
□ 被包含文件内容

- 源文件(*.c)

- 头文件(*.h)

宏定义
数据结构定义
函数说明等

□ 文件包含可嵌套





头文件卫士

- 如果一个.h在包含了
 - 宏、结构体的定义等
- 多个.cpp都文件文件包含了该.h
- 或者该.h被同一个文件两次文件包含
- 编译会出错，报告：XX重复定义
- 解决办法头文件卫士



条件编译

(1) **#ifdef ~ #endif**

□ 一般格式:

ifdef 标识符

程序段1;

[# else

程序段2;]

endif

- 功能：当“标识符”已经被**#define**命令定义过，则编译程序段1，否则编译程序段2。



例

```
#include <cstdio>  
#define DEBUG  
void main()  
{  
    #ifdef DEBUG  
        cout<<"yes\n";  
    #else  
        cout<<"no\n";  
    #endif  
}
```



(2) **#ifndef ~ #endif**

□ 一般格式:

ifndef 标识符
程序段1;

[# else
程序段2;]

endif

- 功能：当“标识符”没有被**#define**命令定义过，
则编译程序段1，否则编译程序段2。



(3) **#if ~ #endif**

□ 一般格式:

if 常量表达式

程序段**1**;

[# else

程序段**2**;]

endif

- 功能：当表达式为非**0**（“逻辑真”）时，编译程序段**1**，否则编译程序段**2**。




例

```
#define R 1
void main()
{
    float c,r,s;
    cout<<"input a number: ";
    cin>>c;
    #if R
        r=3.14159*c*c;
    #else
        r=3.14*c*c;
    #endif
    cout<<"area of round is: "<<r<<endl;
}
```



Debug与Release

- 调试模式
 - 大
- 发行模式
 - 小

- 
- 文件包含是预处理的一个重要功能，它可用来把多个源文件连接成一个源文件进行编译，结果将生成一个目标文件。
 - 条件编译允许只编译源程序中满足条件的程序段，使生成的目标程序较短，从而减少了内存的开销并提高了程序的效率。
 - 使用预处理功能便于程序的修改、阅读、移植和调试，