



移动平台应用软件开发 C/C++/JAVA基础

C中的预处理指令

主讲：张齐勋

zhangqx@ss.pku.edu.cn

《移动平台应用软件开发》课程建设小组

北京大学

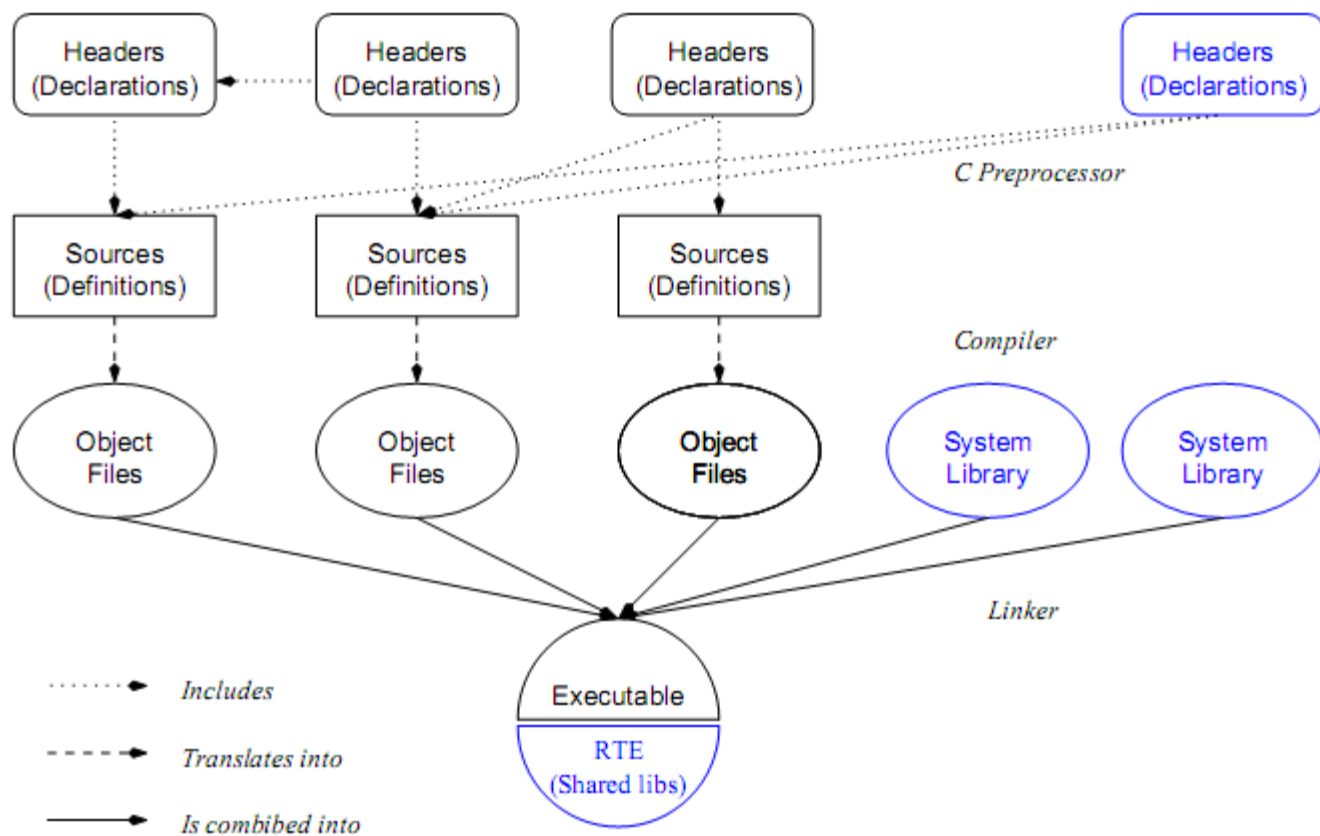
二零一五年



北京大学



预处理





预处理器

- C语言的编译系统分为**编译预处理**和正式编译。
- 预处理作用：对源程序**编译之前**做一些处理, 生成扩展C源程序
- 预处理器的行为是由**预处理指令**控制的。
 - 宏定义 **#define #ifdef #ifndef**
 - 文件包含 **#include**
 - 条件编译 **#if--#else--#endif**
 - 其他宏定义 **#line #error #pragma**

- “#”开头
- 占单独书写行
- 语句尾不加分号
- 可以使用续行标志 “\”





宏定义

- 预处理模块只是用宏名作简单的替换，不作语法检查，若字符串有错误，只有在正式编译时才能检查出来。
- 没有特殊的需要，一般在预处理语句的行末不必加分号，若加了分号，则连同分号一起替换。
- 使用宏定义可以减少程序中重复书写字符串的工作量，提高程序的可移植性。



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)

```
如  #define YES 1
     #define NO 0
     #define PI 3.1415926
     #define OUT printf("Hello,World");
```

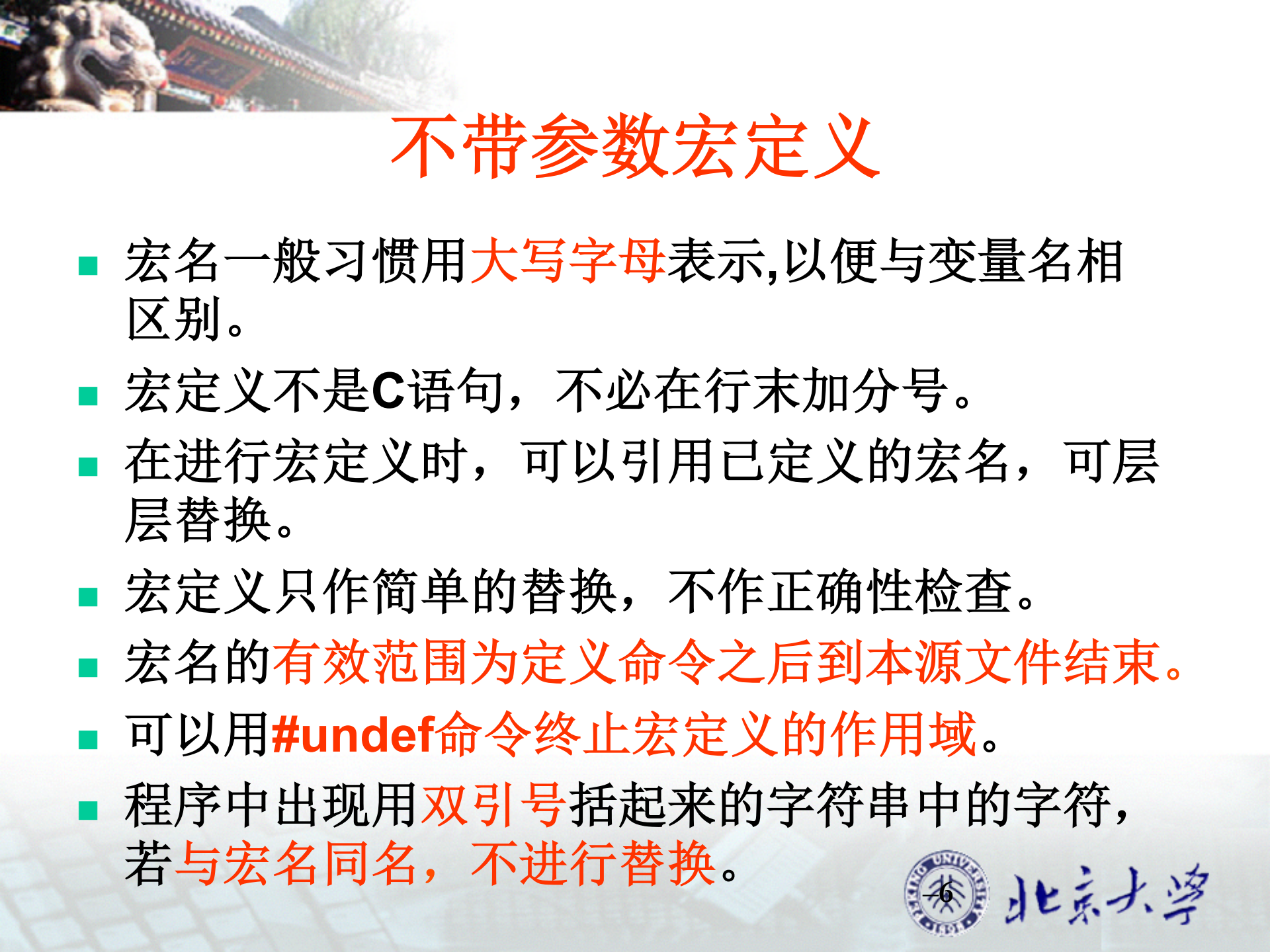
```
例 #define WIDTH 80
     #define LENGTH WIDTH+40
     var=LENGTH*2;
宏展开: var= 80+40 *2;
```

```
例 #define WIDTH 80
     #define LENGTH (WIDTH+40)
     var=LENGTH*2;
宏展开: var= (80+40) *2;
```

```
例 #define MAX MAX+10 (×)
```



北京大學



不带参数宏定义

- 宏名一般习惯用**大写字母**表示,以便与变量名相区别。
- 宏定义不是**C**语句,不必在行末加分号。
- 在进行宏定义时,可以引用已定义的宏名,可层层替换。
- 宏定义只作简单的替换,不作正确性检查。
- 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域。
- 程序中出现用**双引号**括起来的字符串中的字符,若**与宏名同名**,不进行替换。



带参数宏定义

- 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;**

- 宏展开: 形参用实参换, 其它字符保留

例 **#define S (r) PI*r*r**

相当于定义了不带参宏**S**,代表字符串“(r) PI*r*r”

不能加空格

- 注意宏体的括号

例 **#define POWER(x) x*x**

x=4; y=6;

z=POWER(x+y);

宏展开: **z=x+y*x+y;**

一般写成: **#define POWER(x) ((x)*(x))**

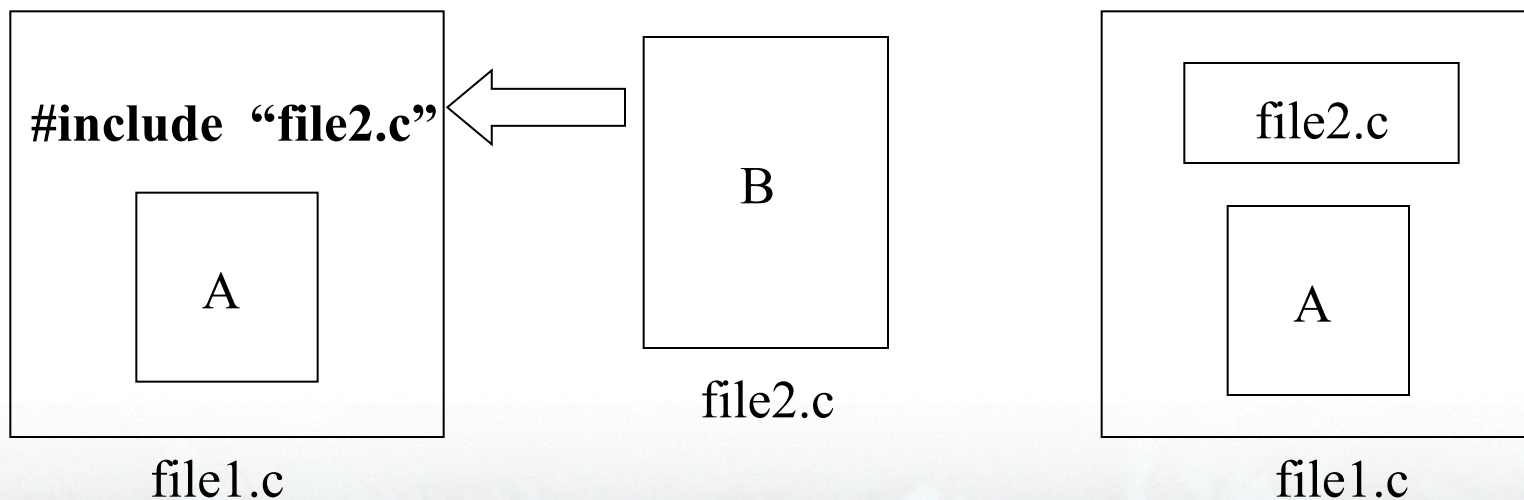
宏展开: **z=((x+y)*(x+y));**



北京大学

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

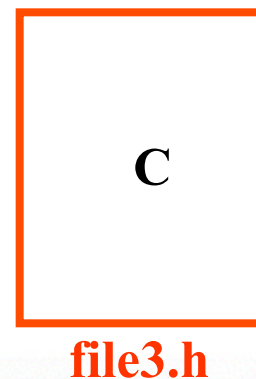
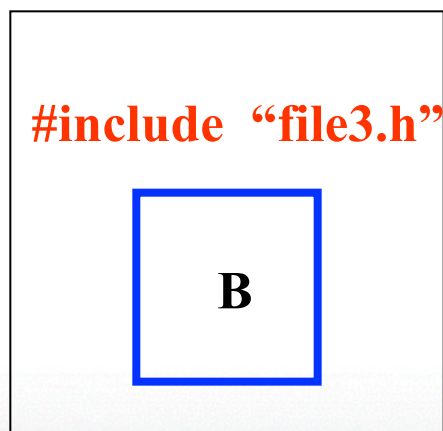
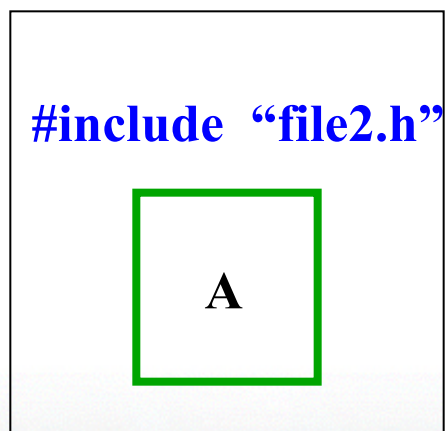
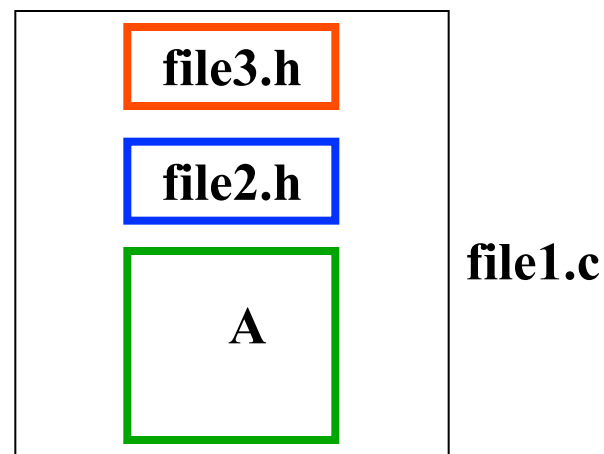




被包含文件内容
源文件(*.c)
头文件(*.h)

文件包含可嵌套

宏定义
数据结构定义
函数说明等



北京大学



■ 例子

```
/* powers.h */  
#define sqr(x)  ((x)*(x))  
#define cube(x) ((x)*(x)*(x))  
#define quad(x) ((x)*(x)*(x)*(x))
```

```
#include <stdio.h>  
#include "powers.h"  
#define MAX_POWER 10  
void main()  
{ int n;  
  printf("number\t exp2\t exp3\t exp4\n");  
  printf("----\t----\t-----\t-----\n");  
  for(n=1;n<=MAX_POWER;n++)  
    printf("%2d\t %3d\t %4d\t %5d\n",n,sqr(n),cube(n),quad(n))  
}
```



条件编译

- C语言的编译预处理器还提供了条件编译能力，使得可以对源程序的一部分内容进行编译，即不同的编译条件产生不同的目标代码。
 - **#ifdef.....#else..... #endif**
 - **#ifndef.....#else..... #endif**
 - **#undef**
 - **#if.....#else..... #endif**





#ifdef.....#else..... #endif

```
#ifdef 标识符  
    程序段1  
#else  
    程序段2  
#endif
```

若标识符已经被定义过（一般用**#define**命令定义），那么**程序段1**参加编译，否则**程序段2**参加编译，其中**#else**部分可以省略

```
#ifdef 标识符  
    程序段1  
#endif
```



#ifndef.....#else..... #endif

```
#ifndef 标识符  
    程序段1  
#else  
    程序段2  
#endif
```

若标识符没有定义，程序段1参加编译，否则程序段2参加编译，其中**#else**部分可以省略

```
#ifndef 标识符  
    程序段1  
#endif
```





#undef

- 将已定义的标识符变为未定义的。例如：

若**#undef DEBUG**

则语句：

#ifdef DEBUG //为假（0）

而语句：

#ifndef DEBUG //为真（非0）





#if.....#else..... #endif

```
#if 表达式  
    程序段1  
#else  
    程序段2  
#endif
```

若表达式为“真”（非0），程序段1参加编译，否则程序段2参加编译，其中**#else**部分可以省略。

```
#if 表达式  
    程序段1  
#endif
```





使用技巧

防止一个头文件
被重复包含

```
#ifndef COMDEF_H
#define COMDEF_H
    //头文件内容
#endif
```

求最大值
和最小值

```
#define MAX( x, y ) ( ((x) > (y)) ? (x) : (y) )

#define MIN( x, y ) ( ((x) < (y)) ? (x) : (y) )
```

计算数组元
素的个数

```
#define ARR_SIZE( a ) ( sizeof( (a) ) /  
sizeof( (a[0]) ) )
```



其他预处理指令

- **#error**
- 预定义宏
- **#line**
- **#**和**##**运算符





#error

- **#error**指令强制编译程序停止编译，
- **#error**指令的一般形式是
 - **#error error-message**
- 注意：
 - 宏串**error-message**不用双引号包围
 - 可以自定义错误内容
 - 不同于**printf**，在编译期间输出信息
 - 通常与**#if**等，配合使用
- 主要用于程序调试





预定义宏

- **__DATE__** 进行预处理的日期（“Mmm dd yyyy”形式的字符串文字）
- **__FILE__** 代表当前源代码文件名的字符串文字
- **__LINE__** 代表当前源代码中的行号的整数常量
- **__TIME__** 源文件编译时间，格式为“hh: mm: ss”
- **__func__** 当前所在函数名





```
#include <stdio.h>
#include <stdlib.h>
void why_me();
int main()
{
    printf( "The file is %s.\n", __FILE__ );
    printf( "The date is %s.\n", __DATE__ );
    printf( "The time is %s.\n", __TIME__ );
    printf( "This is line %d.\n", __LINE__ );
    printf( "This function is %s.\n",
__func__ );
    why_me();
    return 0;
}
```

```
void why_me()
{
    printf( "This function is %s\n",
__func__ );
    printf( "The file is %s.\n", __FILE__ );
    printf( "This is line %d.\n", __LINE__ );
}
```





#line

- #line指令改变__LINE__和__FILE__的内容
- #line的一般形式
 - #line number “filename”
- 主要用于调试和特殊应用





#和##

- 宏#运算符，通常称为字符串化的操作符，将宏的一个参数转换为字符串。

```
#include <stdio.h>
#define mkstr(s) #s
int main(void)
{
    printf(mkstr(I like C));
    return 0;
}
```

预处理程序把以下的语句：
 printf(mkstr(I like C));
处理为
 printf("I like C");

—宏#运算符仅允许出现在带参数的宏的替换列表中。



#和##

- 宏##运算符，将两个记号（如标识符）“粘合”在一起，成为一个记号。

```
#include<stdio.h>
#define SORT(X) sort_function ## X
void main(void)
{
    char *array;
    int elements, element_size;
    SORT(3) (array, elements,
    element_size);
}
```

结果:

SORT(3) 转化为sort_function3

SORT(3) (array, elements, element_size)转化为

sort_function3(array, elements, element_size)



清华大学



Q&A

本讲结束！



北京大学