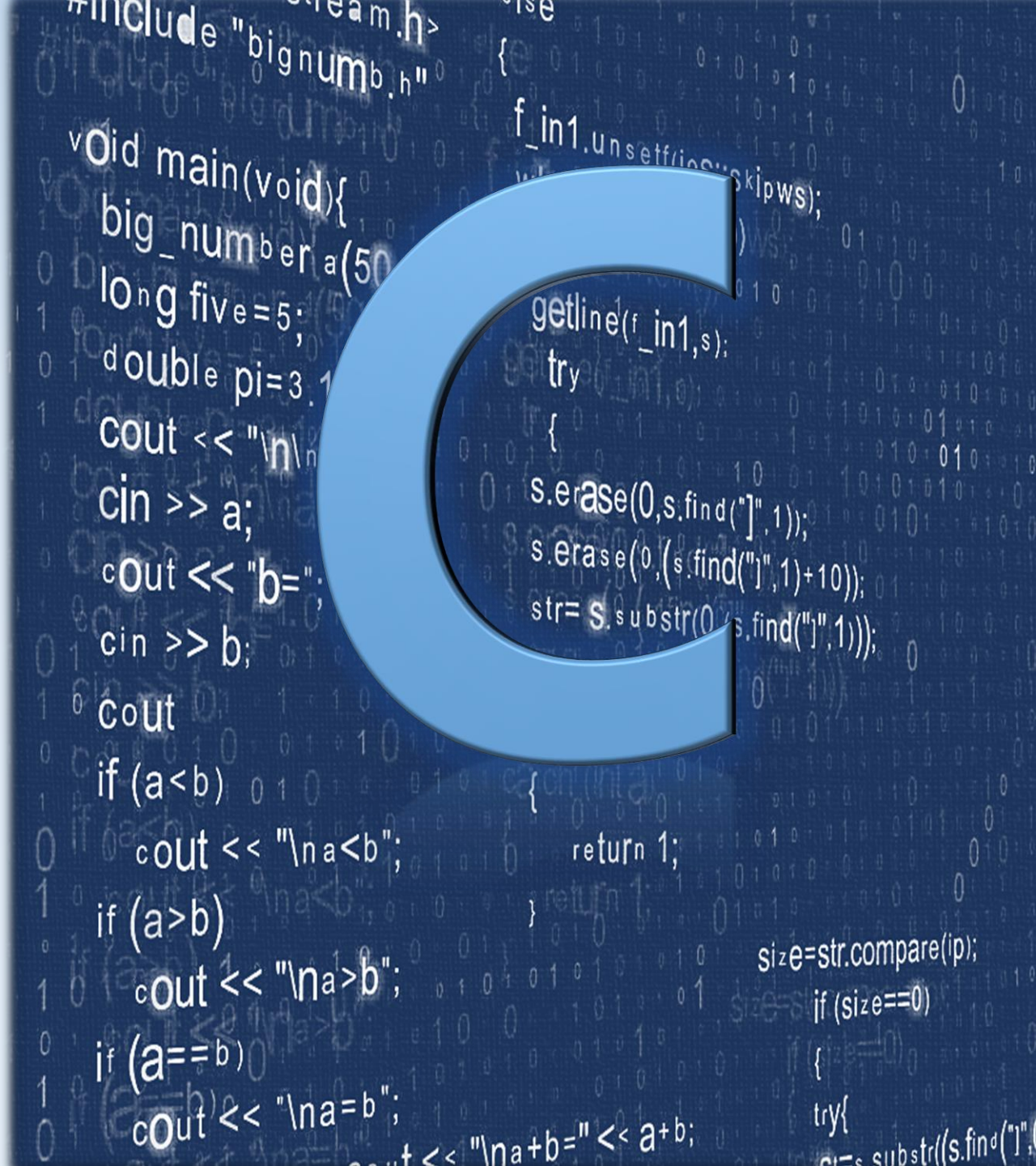


《C语言程序设计》

C语言课程组



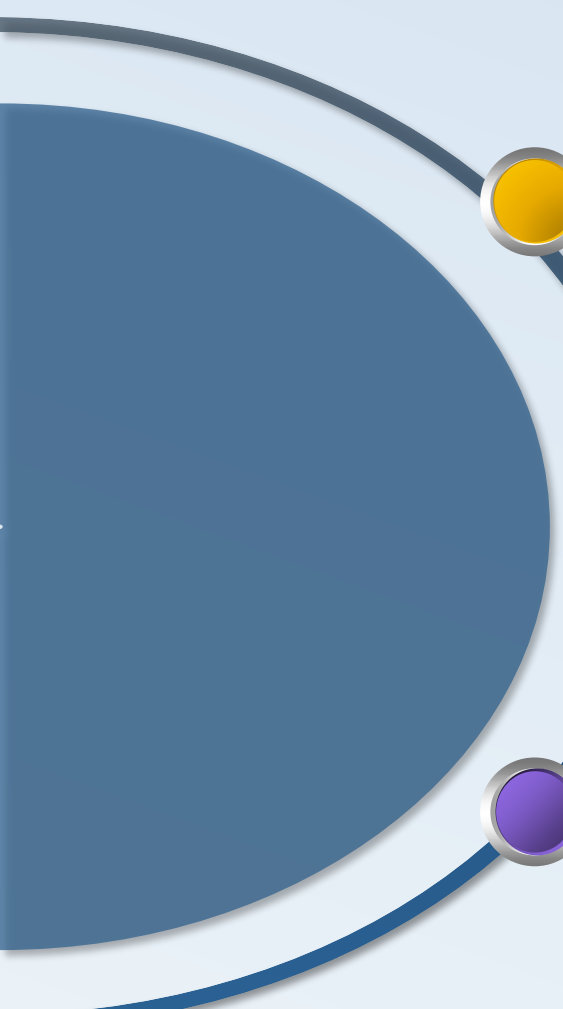
上一讲知识复习

- ◆ 结构体的定义
- ◆ 结构体变量的声明
- ◆ 结构体指针的声明
- ◆ 通过结构体变量名来引用其分量
- ◆ 通过指向结构体的指针来应用其分量
- ◆ 结构体数组
- ◆ 链表
- ◆ 共同体、枚举

本讲教学目标

- ◆理解预处理指令的作用及给程序员带来的好处。
- ◆重点掌握#include、#define预处理指令。
- ◆掌握宏定义的方法,学会分析宏替换的详细过程。
- ◆掌握条件编译指令的作用及给程序移植、调试等带来的好处。
- ◆了解预定义宏，学会使用预定义宏。

本讲授课程内容



预处理器与预处理指令



文件包含

宏定义与宏替换

条件编译

预处理器与预处理指令

```
#include <stdio.h>

void sayHello(void)
{
    printf("Hello World!\n");
}

int main(void)
{
    sayHello();
    return 0;
}
```


预处理器与预处理指令

❖ 预处理是在编译前所作的一项工作，一般预处理命令行都是以#开头的。

预处理指令	功能描述
<code>#include</code>	插入另一文件中的文本
<code>#define</code>	定义预处理器宏
<code>#undef</code>	取消预处理器宏定义
<code>#if</code>	根据常量表达式值有条件地包括一些文本
<code>#ifdef</code>	根据是否定义宏名有条件地包括一些文本
<code>#ifndef</code>	根据与 <code>#ifdef</code> 相反的测试有条件地包括一些文本
<code>#else</code>	上述 <code>#if</code> 、 <code>#ifdef</code> 、 <code>#ifndef</code> 或 <code>#elif</code> 测试失败时包括一些文本
<code>#endif</code>	终止条件文本
<code>#elif</code>	上述 <code>#if</code> 、 <code>#ifdef</code> 、 <code>#ifndef</code> 或 <code>#elif</code> 测试失败时根据另一常量表达式的值包括一些文本
<code>#line</code>	提供编译器消息的行号
<code>#pragma</code>	对编译器指定实现相关信息
<code>#error</code>	将编译错误换成指定信息

本讲授课程内容



预处理器与预处理指令

文件包含

宏定义与宏替换

条件编译



文件包含

❖ #include 预处理指令在标准C语言中的形式

- ◆ #include <文件名>

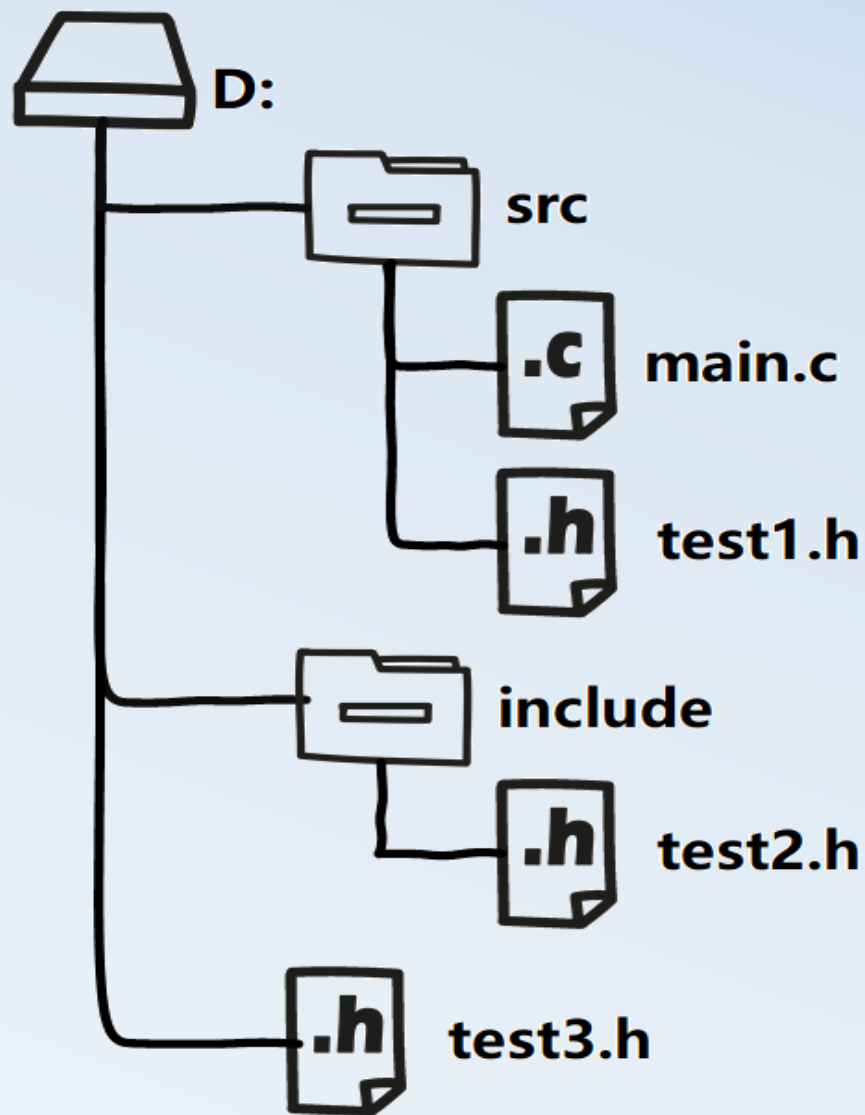
- ◆ #include "文件名"

❖ 两种形式的使用区别：

- ◆ #include <stdio.h> 将导致预处理器根据实现定义的搜索规则从某个指定的文件夹寻找stdio.h。

- ◆ #include "stdio.h" 将导致预处理器先搜索本地文件夹，如果未找到的话，再去系统路径查找。

文件包含



注意：“”形式引用编程人员自己写的头文件
而 <> 形式引用标准实现文件.

注意：在<>或“”中，有时会出现 .. 或 . ,
.. 表示当前文件所在目录的上一级目录；
. 则表示当前文件所在的目录.

本讲授课程内容



预处理器与预处理指令

文件包含

宏定义与宏替换

条件编译



宏定义与宏替换

- ❖ #define 预处理指令把标识符**定义为宏**，出现宏的地方被视为对**宏的调用**。
- ❖ #define 可以定义的两类宏：
 - ◆ 对象宏（无参宏）
 - ◆ 函数式宏（有参宏）

宏定义与宏替换

❖ 简单对象式宏定义的格式如下：

#define 宏名 宏体

❖ 注意：

- ◆ 把源程序中相应宏名用宏体予以替换，仅是简单的字符串替换，没有任何语法正确性检查
- ◆ 宏名和宏体之间有空格，宏体最好用括号括起来
- ◆ 宏定义与main函数在同一层，行末没有分号，如有分号将会被一起替换

❖ #undef 命令则可以取消宏定义，其形式如下：

#undef 宏名

宏定义与宏替换

❖ 阅读下面的宏定义语句，指出宏名与宏体

◆ #define UNIVERSITY "He Bei Normal University"

◆ #define PI 3.145926f

◆ #define ERRMSG "Error: %s\n"

◆ #define BLOCK_SIZE 0x100

◆ #define TRACK_SIZE (16*BLOCK_SIZE)

◆ #define AREA "2*PI*r"

- 宏定义语句**可以嵌套**但**不能递归**
- 宏名一般**习惯大写**，但这不是语法规定
- 宏**不替换双引号里面**的内容

宏定义与宏替换

```
//宏定义的作用域
#define A "This is the first macro"
void f1(void)
{
    printf( "A\n" );
}
#define B "This is the second macro"
void f2(void)
{
    printf( B );
}
#undef B
int main(void)
{
    f1( );
    f2( );
    return 0;
}
```

A 的有效范围

B 的有效范围

宏定义与宏替换

❖ 函数式宏定义的常用格式如下：

#define 宏名(参数表) 宏体

❖ 注意

- ◆ 宏名和括号之间的**不能有空格**，参数表可为空
- ◆ 参数表里的参数**无类型**，多个参数用 “,” 分开
- ◆ 宏体最好使用**小括号**括起，以防歧义产生

❖ 请分析下面的语句

```
#define product(x, y) ((x) * (y))  
double x = product(3, 4);
```

宏定义与宏替换

❖ 阅读下面的程序，体会函数式宏调用的扩展及宏扩展的过程。

```
#include <stdio.h>
#define ARRAY_SIZE 10
#define LOOP_PRINT(i, startPos, endPos)
    for((i) = (startPos); (i) < (endPos); (i)++)
#define PRINT_CONTROL "%d "
int main(void)
{
    int a[ARRAY_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int j = 0;
    LOOP_PRINT(j, 0, ARRAY_SIZE)
        printf(PRINT_CONTROL, a[j]);
    printf("\n");
    return 0;
}
```

宏定义与宏替换

❖ 阅读下面的程序段，给出输出结果

```
#define F(x)  x - 2
#define D(x)  x * F(x)
int main(void)
{
    printf("%d,%d", D(3), D(D(3))) ;
    return 0;
}
```

❖ 定义函数宏

- 交换两个数的值
- 定义宏LOWCASE，判断字符c是否为小写字母
- 定义宏CTOD将数字字符（ '0' ~ '9' ）转换为相应的十进制整数，-1表示出错。

宏定义与宏替换

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

宏展开: $t=(a+b)>(c+d)?(a+b):(c+d);$

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

- ❖ 要求**功能简单，代码短小**，则可以使用函数宏
- ❖ 如果要求**效率较高**，可以采用函数宏，函数宏没有函数调用的代价。

宏定义与宏替换

- ❖ 标准C语言指定了某些对象式宏，这些宏不能被取消定义或由编程人员重新定义，含义也是固定的

预定义宏	含义
<code>__LINE__</code>	当前源程序行的行号，表示为十进制整型字面值
<code>__FILE__</code>	当前源文件名，表示为字符串字面值
<code>__DATE__</code>	转换的日历日期，表示为“Mm dd yyyy”形式的字符串字面值
<code>__TIME__</code>	转换的时间，表示为“hh: mm: ss”形式的字符串字面值
<code>__STDC__</code>	编译器为 ISO 兼容实现时为十进制整型字面值 1
<code>__STDC_VERSION__</code>	若实现符合 C89 增补 1，则值为 199409L；若实现符合 C99，则值为 199901L；否则值未定义
<code>__STDC_EOSTED__</code>	(C99)。实现为宿主实现时为 1，独立实现时为 0
<code>__STDC_IEC_559__</code>	(C99)。浮点数实现符合 IEC 50559 标准时定义为 1，否则数值未定义
<code>__STDC_IEC_559_COMPLEX__</code>	(C99)。复数实现符合 IEC 50559 标准时定义为 1，否则数值未定义
<code>__STDC_ISO_10646__</code>	(C99)。定义为长整型常量，yyyymmL 表示 <code>wchar_t</code> 值符合 ISO 10646 标准及其指定年月的修订补充，否则数值未定义

宏定义与宏替换

❖ 阅读下面程序体会预定义宏的作用。

```
#include <stdio.h>
void foo(void)
{
    printf( "foo()所在的文件为: %s.\n", __FILE__ );
    printf( "当前行号: %d.\n", __LINE__ );
}
int main(void)
{
    printf( "main()所在的文件为: %s.\n", __FILE__ );
    printf( "编译的日期为: %s.\n", __DATE__ );
    printf( "编译的时间为: %s.\n", __TIME__ );
    printf( "当前行号: %d.\n", __LINE__ );
    foo();
    return 0;
}
```


宏定义与宏替换

❖注意：

- ❖宏替换在程序**编译前**进行；
- ❖宏替换不是C语句，**不用以分号结尾**；
- ❖宏替换只是简单的字符串替换，**不做语法检查**；
- ❖宏替换不替换双引号内的字符串；
- ❖宏定义可以嵌套，但不允许递归；
- ❖函数宏定义时，参数最好加括号，尽量多使用括号

本讲授课程内容



预处理器与预处理指令

文件包含

宏定义与宏替换

条件编译



条件编译

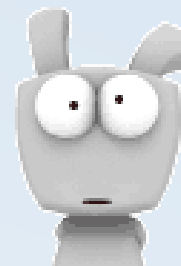
❖ 预处理器条件指令（**#if**、**#else**、**#elif**、**#endif** 等）允许预处理器根据计算条件处理和替换宏。

❖ 处理形式：

```
#if 常量表达式1
    语句块1
#elif 常量表达式2
    语句块2
...
#else
    最后一个语句块
#endif
```

问题：

• 能否根据条件决定哪些宏需要处理和替换呢？



条件编译

❖ 阅读下面的程序，比较 #if 与 if 语句的相同与不同

```
#include <stdio.h>
#include <stdlib.h>
#define DEBUG 0
int main(void)
{
    int x = 1;
    #if DEBUG
        printf("%d\n", x);
    #endif
    system("PAUSE");
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define DEBUG 0
int main(void)
{
    int x = 1;
    if(DEBUG)
        printf("%d\n", x);
    system("PAUSE");
    return 0;
}
```

条件编译

- ❖ 设有一个C源程序将分别在Windows、Linux平台上编译，并设有两个函数w_foo()、l_foo()。

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

```
#define WINDOWS 1
```

```
#define LIUNIX 0
```

```
void w_foo()
```

```
{
```

```
    printf("w_foo() is called!\n");
```

```
}
```

```
void l_foo()
```

```
{
```

```
    printf("l_foo() is called!\n");
```

```
}
```

```
int main(void)
```

```
{
```

```
    #if WINDOWS
```

```
        w_foo();
```

```
    #elif LIUNIX
```

```
        l_foo();
```

```
    #endif
```

```
    return 0;
```

```
}
```

条件编译

❖ #ifdef、#ifndef 用于测试一个名称是否被定义为预处理宏。

◆ 主要用来解决头文件重复包含

//#ifdef 的基本形式为：

#ifdef 宏名

语句块

#endif

//#ifndef 的基本形式为：

#ifndef 宏名

语句块

#endif

条件编译

- ❖ 设有a.h、b.h、a.c，其文件内容大致如下，请分析编译a.c时将会产生什么样的编译错误，并利用#ifdef、#define、#endif解决这一问题。

```
/* 头文件 a.h */  
int x = 10;  
void foo();
```

```
/* 头文件 b.h */  
#include "a.h"
```

```
/* 源文件 a.c */  
#include "a.h"  
#include "b.h"  
  
int main(void)  
{  
    return 0;  
};
```

Thank You !