



清华大学

腾讯会议 ID: 791 102 390

*Tsinghua University*

# 计算机程序设计基础

## 第7讲 函数

沈瑜 (010-62782951)

shenyu@tsinghua.edu.cn

清华大学电机系

2022.10.25



## 前6讲知识点回顾

- ◆ **main**函数、头文件
- ◆ 整型和实型数据、字符类型
- ◆ 算术表达式、赋值表达式、逗号表达式、逻辑表达式（运算符**operator**、操作数**operand**）
- ◆ 语句（复合语句）
- ◆ **printf**、**scanf**
- ◆ 选择结构、循环结构、跳转语句
- ◆ 一维数组、冒泡法排序

运算的优先级：附录D， **p378**





# 主要内容

➤ 基于函数的C程序设计举例

➤ 函数基本介绍

➤ 典型库函数：数学函数

➤ 自定义函数

✓ 函数的定义

✓ 函数的调用

✓ 函数调用时的数据传递（形参与实参）

✓ 函数调用的过程

✓ 函数的返回值

✓ 函数的声明

➤ 作用域、局部变量和全局变量

➤ 程序的运行机理与调试Debug

重点

参考教材： 第7.1-7.5节、第7.8节、附录F p384  
《C语言程序设计：现代方法(第2版)》第9章





## 7.1 基于函数的C程序设计举例

### ● 例1: 数学函数的使用示例

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

#define PI 3.1415926

int main(void)
{
    printf("sin(PI/6)=%f\ncos(PI/6)=%f\n", sin(PI/6.0), cos(PI/6.0) );
    printf("ln1=%f\nlg2=%f\n", log(1.0), log10(2.0) );
    printf("e^(-1)=%f\n2^32=%.0f\n", exp(-1.0), pow(2.0, 32.0) );
    return 0;
}
```

```
sin(PI/6)=0.500000
cos(PI/6)=0.866025
ln1=0.000000
lg2=0.301030
e^(-1)=0.367879
2^32=4294967296
```





## ● 例2: 用函数找到两数的最大值 （教材p176, 例7.2）

```
#include <stdio.h>

int max(int x, int y)
{
    int z;
    z = x > y ? x : y;
    return z;
}
```

→ 函数定义

```
int main()
{
    int a=30, b=20, c=25, z;
    z = max(a, b);
    z = max(z, c);

    printf("最大的数是: %d\n", z);
    return 0;
}
```

→ 函数调用







## ●例3: 自定义函数进行调试

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

getch在conio.h中定义

```
void My_Print_Debug_Info(int i, int fac)
{
    //这个调试函数版本比较土，简单示意，以后给完美的方案
    printf("[调试]\ti=%d\tfac=%d\t任意键继续调试....\n", i, fac );
    getch();
    return;
}
```

```
int main()
{
    int i, N, fac=1;
    printf("请输入待求阶乘数N: ");
    scanf("%d", &N);
    printf("[调试]\tN=%d\n", N );//输出调试信息
    for(i=1; i<=N; i++)
    {
        fac = fac * i;
        My_Print_Debug_Info(i, fac);//输出调试信息
    }
    printf("1*2*...*%d=%d\n", N, fac);
    return 0;
}
```

未熟练掌握debug技能之前，请多printf重要数据

```
C:\windows\system...
请输入待求阶乘数N: 5
[调试] N=5
[调试] i=1    fac=1    任意键继续调试....
[调试] i=2    fac=2    任意键继续调试....
[调试] i=3    fac=6    任意键继续调试....
[调试] i=4    fac=24   任意键继续调试....
[调试] i=5    fac=120  任意键继续调试....
1*2*...*5=120
```





## ● 例4: 计算成绩的平均值（采用数组/函数实现）

```
1 #include <stdio.h>
2 #define MAX_NUM 200
3 int num=5; //学生人数, 全局变量
4 int ave=0; //N个学生的平均成绩, 全局变量
5 int data[MAX_NUM]={0}; //存放成绩的数组, 全局变量
6 void InputData();
7 void Calculate();
8 void Output();
9
10 int main()
11 {
12     InputData(); //输入数据, 检查输入逻辑
13     Calculate(); //计算 (注意避免被0除)
14     Output();    //输出结果
15
16     return 0;
17 }
```

### 模块化程序设计

```
19 void InputData()
20 {
21     printf("请输入学生人数: \n");
22     scanf("%d", &num);
23     if( num < 0 ) num = 0; //若输入负数, 强制为0, 否则后续程序出错
24     if( num > MAX_NUM ) num = MAX_NUM; //输入过大, 则强制为MAX_NUM
25
26     printf("请输入%d个学生的成绩\n", num);
27     for(int i=0; i<num; i++)
28     {
29         scanf("%d", &data[i]);
30     }
31 }
32
33 void Calculate()
34 {
35     int sum = 0;
36     for(int i=0; i<num; i++)
37     {
38         sum = sum + data[i];
39     }
40
41     if( num>0 ) //防止num=0的情况发生. "马路警察, 各管一段"
42         ave = sum/num;
43 }
44
45 void Output()
46 {
47     printf("%d个学生的平均成绩为%d\n", num, ave);
48 }
```



## ● 例5:成绩排序: 冒泡法 (采用数组/函数实现)

```
1 #include <stdio.h>
2
3 #define N 5
4 int a[N];
5
6 void swap(int m, int n); // 交换m和n的值
7
8 int main()
9 {
10     int i, j;
11     printf("input %d integer numbers:\n", N);
12     for(i=0; i<N; i++)
13         scanf("%d", &a[i]); // 输入N个整数
14     printf("\n");
15     for(j=1; j<N; j++) // 第j趟比较
16         for(i=0; i<N-j; i++) // 第j趟中两两比较N-j次
17             if (a[i] > a[i+1]) // 交换两个数的值
18                 swap(i, i+1);
19
20     printf("the sorted numbers:\n");
21     for(i=0; i<N; i++)
22         printf("%d\n", a[i]);
23     return 0;
24 }
```

全局变量

```
26 void swap(int m, int n)
27 {
28     int temp;
29     temp = a[m];
30     a[m] = a[n];
31     a[n] = temp;
32 }
```

C:\Windows\system32\cmd.exe

input 5 numbers:

3 5 2 4 1

the sorted numbers:

1

2

3

4

5

请按任意键继续. . .





## 7.2 函数基本介绍

### 1. 函数（function）的用途

- 重复执行相同或类似的代码

max    My\_Print\_Debug\_Info

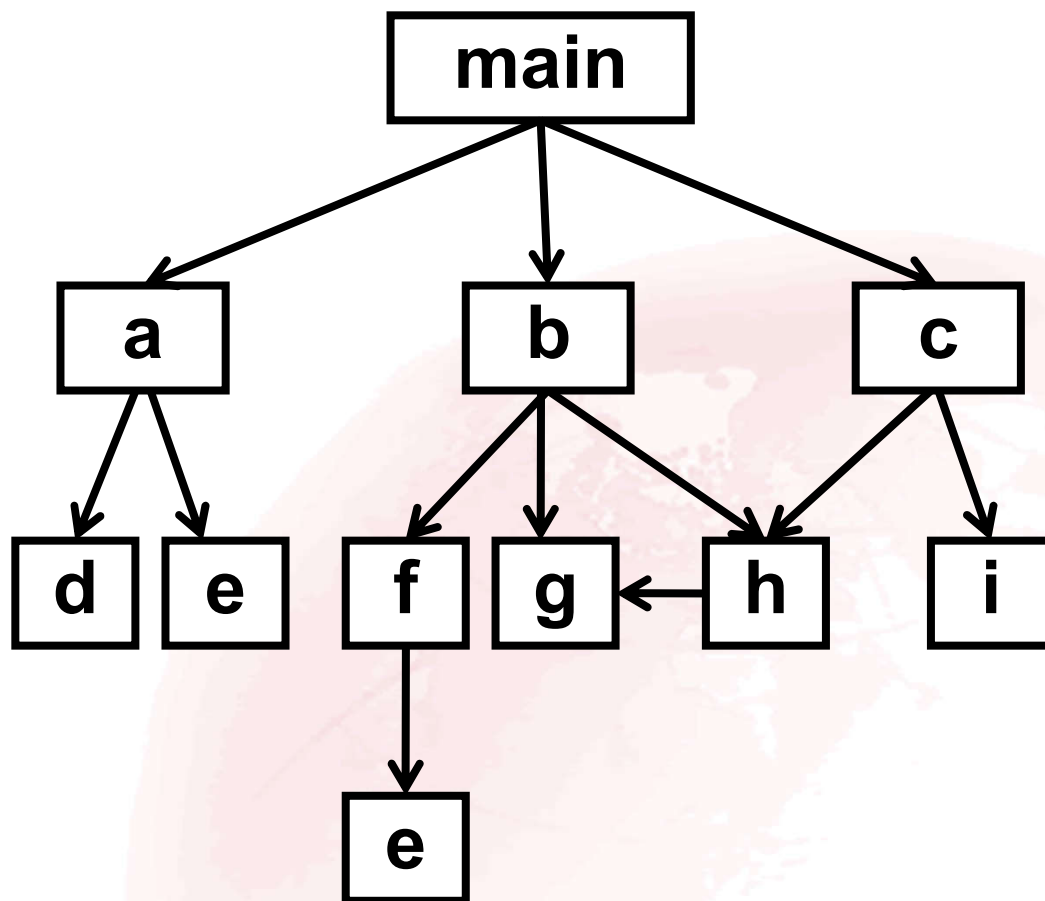
- 预制若干代码模块，实现编程分工

比如数学函数

- 模块化编程，有利于程序阅读、修改、完善



# 模块化程序示例



```
int a() { return 0; }  
int b() { return 0; }  
int c() { return 0; }
```

```
int main()  
{  
    a();  
    b();  
    c();  
    return 0;  
}
```



## 2. C语言中的函数

### ◆ C标准库函数

如: `printf, scanf, sin, cos, ...`

### ◆ 自定义函数

➤ 函数本质是自带声明和语句的小程序

➤ 函数调用也是表达式 → 函数返回值

`f_a = 2.0+sin(0.01);`

- 函数调用运算符 ()

- 参数

- 返回值

➤ 函数调用, 可改变变量值、完成输入输出操作

- 赋值语句; 输入输出





## 7.3 典型库函数：数学函数

### 常用数学函数

**sin, cos, tan, asin, acos, atan**

**sinh, cosh, tanh**

**sqrt, abs**

**exp, pow**

**log, log10**





# 数学函数举例

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

#define PI 3.1415926

int main(void)
{
    printf("sin(PI/6)=%f\nln1=%f\n", sin(PI/6.0), log(1.0));
    return 0;
}
```

函数调用也是表达式 → 函数返回值  
函数调用运算符 ()      参数





## 7.4 自定义函数

### 1. 自定义函数的要求

- C语言要求，在程序中用到的所有函数，必须“先定义，后使用”
- 指定函数名字、函数返回值类型、参数的个数与类型以及函数实现的功能，将这些信息通知编译系统。
- 函数名字必须是标识符，与变量等不得重名



## 2. 函数的定义

- 定义无参函数的一般形式为:

指定函数值的类型

```
类型名 函数名()  
{  
    函数体  
}
```

- ◆ **void**表示没有参数
- ◆ 如果类型名为**void**, 表示没有返回值

```
类型名 函数名(void)  
{  
    函数体  
}
```

包括声明部分和语句部分



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

```
void print_star()
```

```
{ printf("*****\n"); }
```

无参函数定义

```
void print_message()
```

```
{ printf(" How do you do!\n"); }
```

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

```
int main()
```

```
{
```

```
    print_star();
```

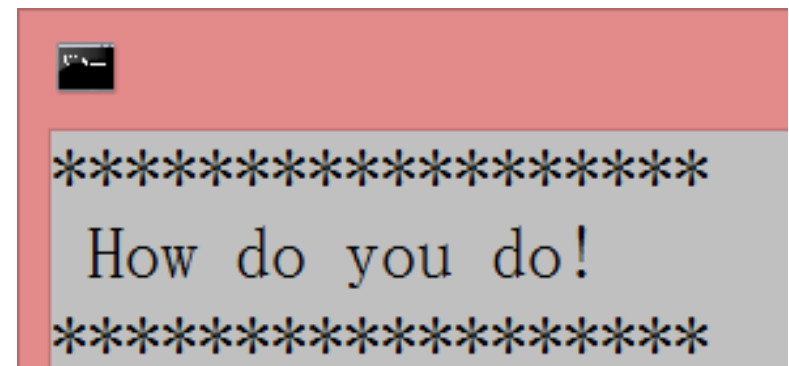
```
    print_message();
```

```
    print_star();
```

```
    return 0;
```

```
}
```

函数调用





- 定义有参函数的一般形式为:

类型名 函数名(形式参数表列)

{

函数体

}

```
int max(int x, int y)
{
    int z;
    z = x > y ? x : y;
    return z;
}
```

- 定义空函数的一般形式为:

```
类型名 函数名()  
{ }
```

- 先用空函数占一个位置，以后逐步扩充
- 好处：程序结构清楚，可读性好，以后扩充新功能方便，对程序结构影响不大



### 3. 函数的调用

函数调用的一般形式为：

函数名（实参表列）

- ◆如果是调用无参函数，则“实参表列”可以没有，但括号不能省略
- ◆如果实参表列包含多个实参，则各参数间用逗号隔开



## 3种函数调用方式

### (1) 函数调用语句:

把函数调用单独作为一个语句

如 **print\_star();**

这时不要求函数带返回值，只要求函数完成一定的操作

### (2) 函数表达式

- 函数调用出现在另一个表达式中

如 **c=max(a,b);**

- 这时要求函数带回一个确定的值以参加表达式的运算





### (3) 函数参数

- 函数调用作为另一函数调用时的实参

如  $m = \max(a, \max(b, c));$

- 其中  $\max(b, c)$  是一次函数调用，它的值作为  $\max$  另一次调用的实参



## 4. 函数调用时的数据传递（形参与实参）

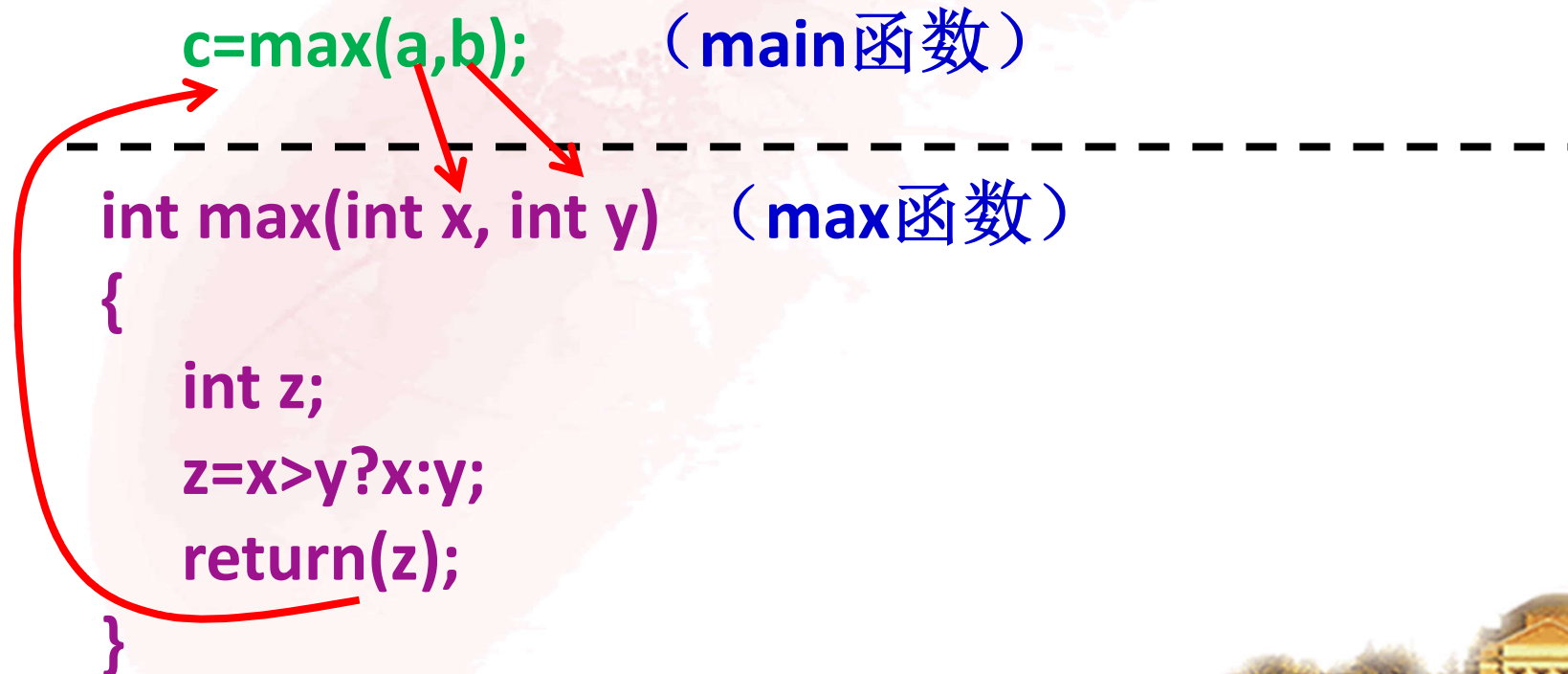
### 形式参数和实际参数

- 在调用有参函数时，主调函数和被调用函数之间有数据传递关系
- 定义函数时函数名后面的变量名称为“形式参数”（简称“形参”）
- 主调函数中调用一个函数时，函数名后面参数称为“实际参数”（简称“实参”）
- 实际参数可以是常量、变量或表达式
  - 一般传递的是值



# 实参和形参间的数据传递

- 在调用函数过程中，系统会把实参的**值**传递给被调用函数的形参（**单向传递**）
- 或者说，形参从实参得到一个值
- 该值在函数调用期间有效，可以参加被调函数中的运算



```
□ #include <stdio.h>
|
|
□ int max(int x, int y)
| {
|   int z;
|   z=x>y?x:y;
|   return z;
| }
```

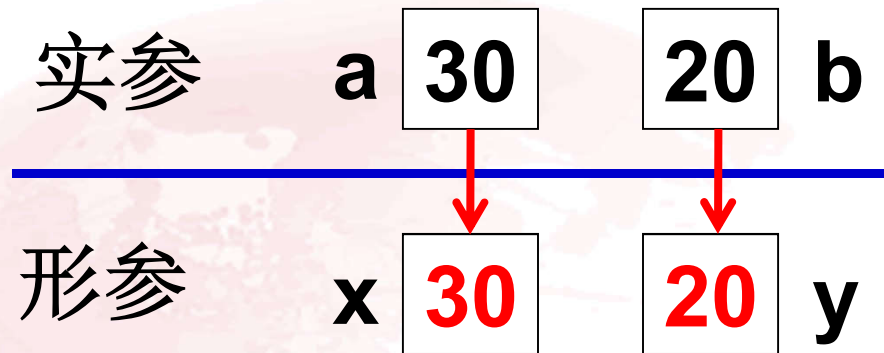
```
□ int main()
| {
|   int a=30, b=20, c=25, z;
|   z = max(a, b);
|   z = max(z, c);
|
|   printf("最大的数是: %d\n", z);
|   return 0;
| }
```





## 5. 函数调用的过程

- 在定义函数中指定的形参，在未出现函数调用时，它们并不占内存中的存储单元。在发生函数调用时，函数**max**的形参被临时分配内存单元。



- 调用结束，形参单元被释放
- 实参单元仍保留并维持原值，没有改变
- 如果在执行一个被调用函数时，形参的值发生改变，不会改变主调函数的实参的值



思考下段程序的输出： 交换a与b的值的函数？

```
void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

错误！

```
int main()
{
    int a = 10;
    int b = 11;
    printf("a=%d, b=%d\n", a, b);
    swap(a, b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

a=10, b=11  
a=10, b=11

在调用函数过程中，  
系统会把实参的**值**  
传递给被调用函数  
的形参（**单向传递**）

## 6. 函数的返回值

➤ 通常，希望通过函数调用使主调函数能得到一个确定的值，这就是**函数值**(函数的返回值)

(1) 函数的返回值是通过函数中的**return语句**获得的。

◆ 一个函数中可以有一个以上的**return**语句，执行到哪一个**return**语句，哪一个就起作用

◆ **return**语句后面的括号可以不要

(2) 函数值的类型。应当在定义函数时指定函数值的类型。

(3) 在定义函数时指定的函数类型一般应该和**return**语句中的表达式类型一致

◆ 如果函数值的类型和**return**语句中表达式的值不一致，则以函数类型为准，将发生**强制类型转换**



## 7. 函数的声明

- 此前的例子中，函数定义总是在调用点的上面
- 如果函数定义在调用点之后，可在调用之前声明(**function declaration**)以完成正确编译
- 通常，声明有参函数的一般形式为：

类型名 函数名(形式参数表列);

函数原型 **prototype**



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

```
int main()
```

```
{ float add(float x, float y);
```

只多一个分号

声明**add**函数

```
float a,b,c;
```

```
printf("Please enter a and b:");
```

```
scanf("%f,%f",&a,&b);
```

```
c=add(a,b);
```

调用**add**函数

```
printf("sum is %f\n",c);
```

```
return 0;
```

```
}
```

定义**add**函数

```
float add(float x,float y)
```

```
{ float z;
```

```
z=x+y;
```

```
return(z);
```

```
}
```

## 函数原型

- 函数原型的一般形式有两种：

如 **float add(float x, float y);**

**float add(float, float);**

- 原型说明可以放在文件的开头，这时所有函数都可以使用此函数





```
void fun(int a, int b, int c)
{ c=a+b;}
```

以下程序的输出是

```
int c;
fun(2,3,c);
printf("%d\n",c);
```

☐ A 2

☐ B 3

☐ C 5

☒ D 无定值

提交





## 7.5 作用域、局部变量和全局变量

### 1. 作用域与生存期

➤ 作用域：编译相关的概念

◆ 变量起作用的空间范围

➤ 生存期：运行相关的概念

◆ 变量起作用的时间范围

● 一个变量，何时被定义（分配了空间），何时消亡（定义的空间被收回）



## 2. 局部变量

➤ 定义在函数之中的变量

➤ 作用域:

◆ 在被定义的函数中

➤ 生存期:

◆ 函数调用开始时开始

● 为局部变量分配空间

◆ 函数调用结束时结束

● 局部变量分配空间被收回

```
int max(int x, int y)
{
    int z;
    z = x > y ? x : y;
    return z;
}
```



```
void printTime(int hour, int minute) {  
    printf("time is %2d:%2d\n",hour,minute);  
}  
  
void main() {  
    int hour = 23;  
    int minute = 59;  
    printTime(hour,minute);  
}
```

变量名尽量不要重名

**main**和**printTime**中分别定义的  
**hour**、**minute**，是不同的局部变量



## 推荐的编程习惯

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

```
    //int hour=1;
```

```
    //int minute = 10;
```

变量名尽量不要重名

```
void printTime(int theHour, int theMinute)
```

```
{
```

```
    printf( "Time is %2d:%2d\n", theHour, theMinute );
```

```
}
```

```
void main()
```

```
{
```

```
    int hour=23;
```

```
    int minute = 59;
```

```
    printTime(hour, minute);
```

```
}
```



### 3. 全局变量

➤ 定义在函数外部的变量

➤ 作用域:

- ◆ 从定义位置，直到被定义的文件结束

- ◆ 如果函数中有重名的局部变量，局部变量优先作用，全局变量被屏蔽

➤ 生存期:

- ◆ 程序运行开始时开始

- ◆ 程序运行结束时结束





```
int hour = 1 ;  
int minute = 10 ;
```

```
void printTime(int hour, int minute) {  
    printf("time is %2d:%2d\n",hour,minute);  
}
```

全局变量被屏蔽

被打印的  
是什么值?

```
void main() {  
    int hour = 23;  
    int minute = 59;  
    printTime(hour,minute);  
}
```

全局变量被屏蔽



## 7.6 程序的运行机理与调试Debug

### 1. 程序的运行

- 从main函数开始执行
- 一般情况下，逐行顺序执行
- 若有选择/循环/跳转语句，则改变执行顺序
- 若有函数，则跳转到函数内部执行，函数执行完毕后返回主调函数，执行下一条语句

可能有输入输出

变量的值被改变



## 2. 利用printf输出变量的值

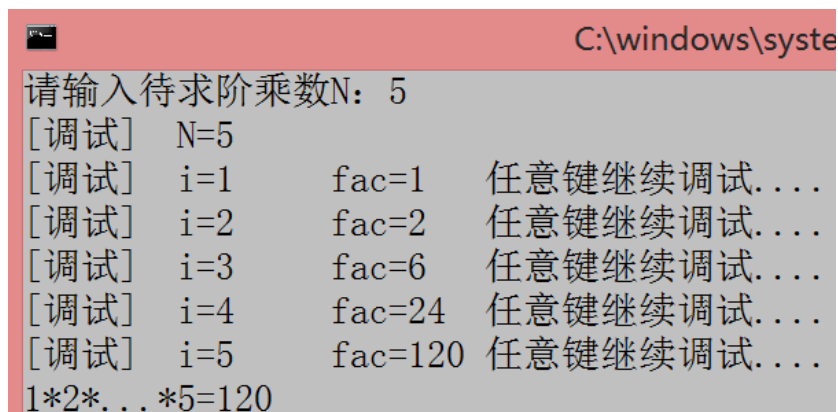
```
#include <stdio.h>
#include <conio.h>
```

getch在conio.h中定义

```
void My_Print_Debug_Info(int i, int fac)
{
    //这个调试函数版本比较土，简单示意，以后给完美的方案
    printf("[调试]\ti=%d\tfac=%d\t任意键继续调试...\n", i, fac );
    getch();
    return;
}
```

```
int main()
{
    int i, N, fac=1;
    printf("请输入待求阶乘数N: ");
    scanf("%d", &N);
    printf("[调试]\tN=%d\n", N );//输出调试信息
    for(i=1; i<=N; i++)
    {
        fac = fac * i;
        My_Print_Debug_Info(i, fac);//输出调试信息
    }
    printf("1*2*...*%d=%d\n", N, fac);
    return 0;
}
```

未熟练掌握debug技能之前，请多printf重要数据



```
C:\windows\system32\cmd.exe
请输入待求阶乘数N: 5
[调试] N=5
[调试] i=1 fac=1 任意键继续调试...
[调试] i=2 fac=2 任意键继续调试...
[调试] i=3 fac=6 任意键继续调试...
[调试] i=4 fac=24 任意键继续调试...
[调试] i=5 fac=120 任意键继续调试...
1*2*...*5=120
```

调试结束后，可注释掉

### 3. Debug手段

通过运行跟踪，理解函数调用

Visual Studio中：

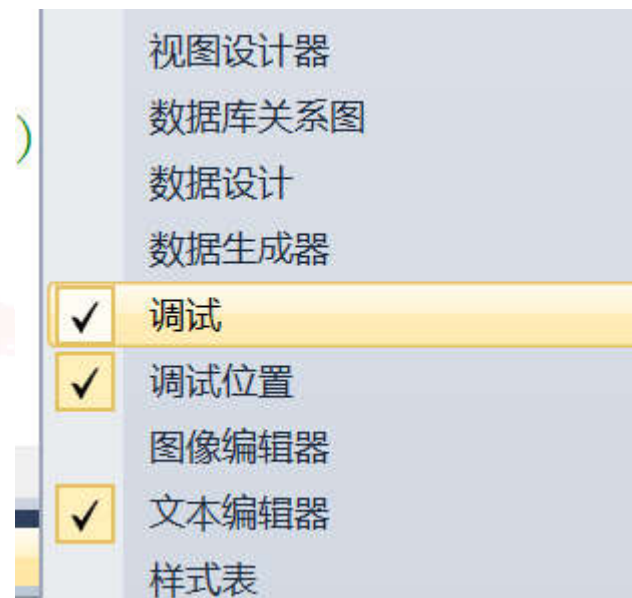
◆ 菜单

◆ 工具条



## 如果找不到工具条？

- 可在菜单栏或工具栏处点击鼠标右键
  - 系统将弹出“菜单”
  - 控制显示哪些工具栏
  - 将“调试”、“调试位置”打开



## 跟踪调试过程

- 1 设置断点: **F9**

- 2 开始调试: **F5**



- 3 通过Debug工具条上的“进入”、“单步”、“跳出”、“当前位置”等功能，跟踪函数执行过程



- **F10, Ctrl F10**
  - **F11, Shift F11**
- 4 停止: **Shift F5**





## 查看、跟踪变量的值



```
int i=0;  
int b[10];  
for(int i=0;i<10;i++)  
    b[i]=i;  
printf("i= %d\n",++i);
```

以上代码运行后的输出是：

A 10

B 11

C 1

D 0

for循环的作用域

提交

