



福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

高级语言程序设计 (C语言与数据结构)

杨雄

83789047@qq.com



第七章 函数

7.1 函数概述

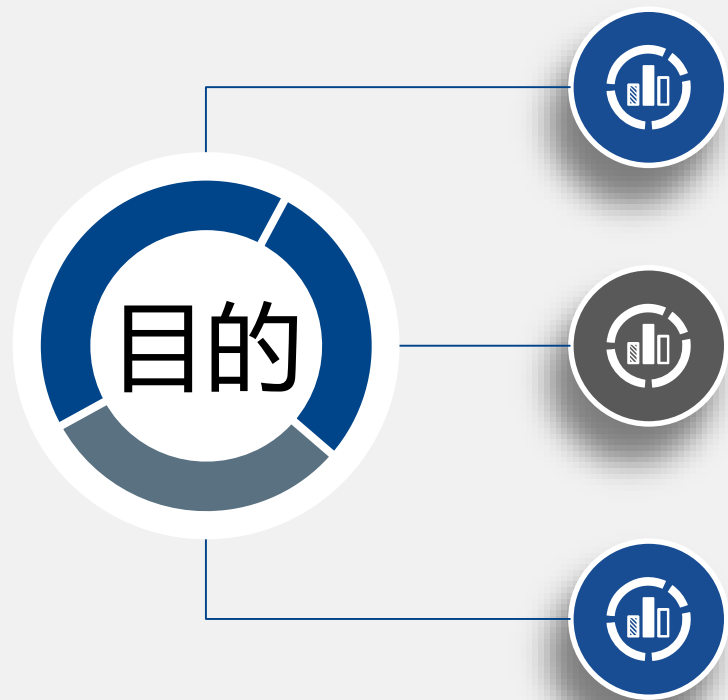
7.2 函数的分类与定义

7.3 函数调用

7.4 函数的嵌套调用和递归调用

7.5 变量的作用域

学习目的



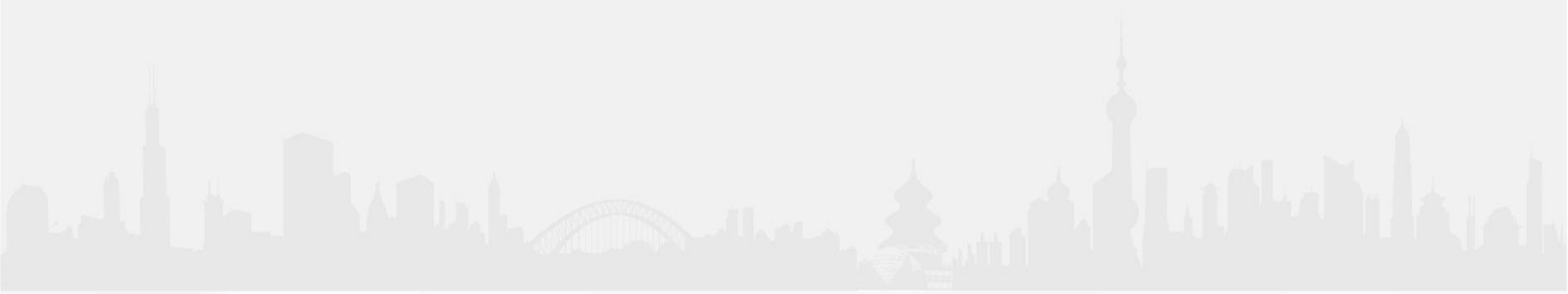
正确理解函数在C语言程序设计中的作用和地位

熟悉函数的定义、原型声明和调用的方法

**掌握值传递和地址传递的区别
掌握全局变量和局部变量区别**

Part.1

7.1 函数概述



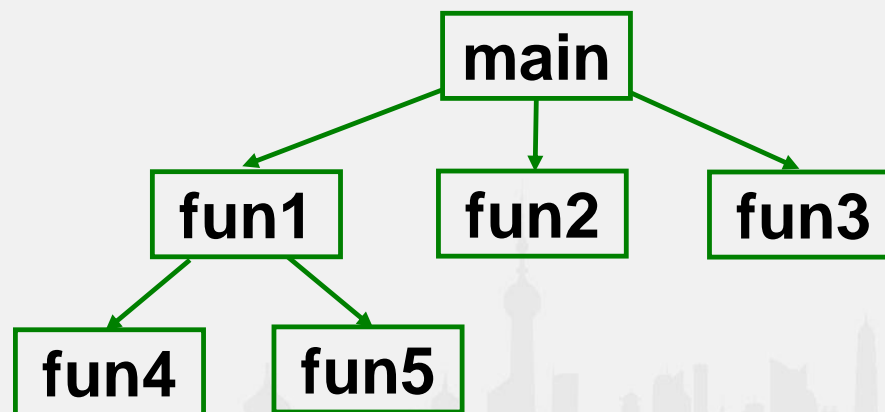
7.1 函数概述

 C语言的程序模块称为函数，利用函数可建立模块化程序。

- 函数是一个独立的具有**特定功能**的程序**模块**。
- 程序的“细化”开发方式使程序**更易于管理**，提高程序的**可读性**。
- 软件的**可重用性**，避免了在程序中使用重复的代码。

7.1 函数概述

- 一个C程序可由**一个主函数**和**若干个其他函数**组成
- 程序执行时从**main函数**开始, 根据需要, main函数调用其他函数, 其他函数也可以互相调用。
- 同一个函数可以被一个或多个函数**调用任意多次**。最后由**main函数结束**程序的运行。
- 不能调用**main函数**



7.1 函数概述

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

```
int max( int x, int y )
```

```
{ if (x>y) return x ;
```

```
  else return y ;
```

```
}
```

函数定义

//使用return把结果返回主调函数

```
void main( )
```

```
{ int a, b, c ;
```

```
  printf(" input two numbers:\n" );
```

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

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

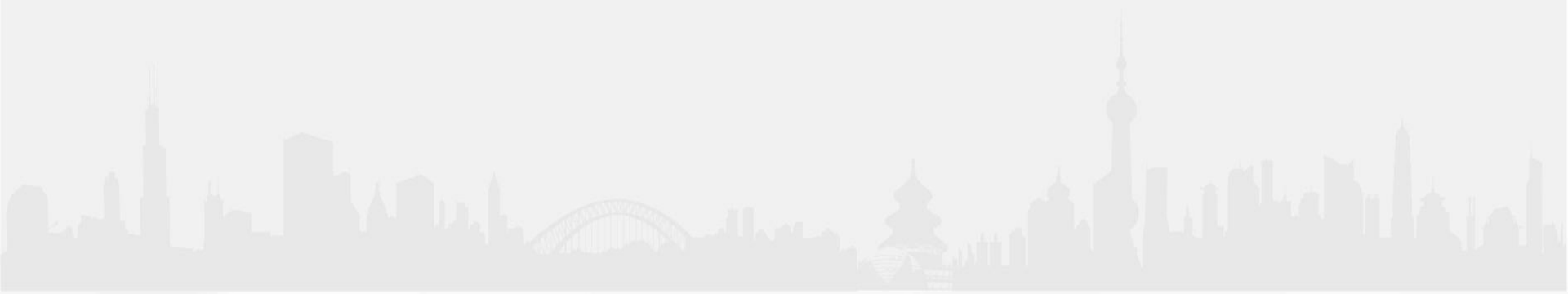
函数调用

```
  printf(" max=%d\n" , c);
```

```
}
```

Part.2

7.2 函数分类与定义



7.2 函数分类与定义

➤ 函数的分类

- 从**用户的观点**分为：
 - C编译系统提供的标准库函数
 - 用户自定义函数；
- 从函数间**数据传送的关系**分为：
 - 有参函数、无参函数
 - 有返回值函数、无返回值函数；
- 从**函数的调用**分为：
 - 内部函数和外部函数。

7.2 函数分类与定义

➤ 函数的定义

• 形式:

函数类型 **函数名(参数表)** —— **函数头**

{
 数据说明
 执行语句
}

} 函数体

```
int max(int x, int y)
{ if (x>y) return x ;
  else return y ;
}
```

7.2 函数分类与定义

◆ 函数类型是指函数值的数据类型

- **void**— 表示函数不返回任何值
称“无类型”或“空类型”

```
void time( long a)           /*无返回值*/  
{ int i;  
  for( i=1; i<=a; i++); } /*延迟一个小的时间片*/  
void main( )  
{ time( 10000 ); }
```

7.2 函数分类与定义

- ◆ 函数名：可以是**任何有效的标识符**。
 - 在一个程序中除了主函数外其余函数的名字可以任意取，但**应有意义**。
 - “()” 函数标志

```
int max( int x, int y )  
{ if (x>y) return x ;  
  else return y ;  
}
```

7.2 函数分类与定义

◆ 参数表

- 定义函数时参数表中的参数称为形式参数, 用逗号分隔。

如: **int** max(**int** x , **int** y)

- 形式参数可以是变量名、数组名, 不允许是常量、表达式或数组元素。必须为每个参数指定数据类型。

如: **float** fact(**int** n, **int** x[i]) ×

- 形参省略时称**无参**函数, 但此时函数名后的圆括号不能省。

```
dump( )  
{  
}
```

7.2 函数分类与定义

◆ 常见的程序设计错误有:

- 把同一种类型的参数声明写成:

`float max(float x, y) ×`

正确的是: `float max(float x, float y)`

- 在定义函数时, 在右圆括号后使用分号。

如: `float max(float x, float y) ; ×`

7.2 函数分类与定义

◆ 函数体：用来完成具体任务的一个程序段。

结构： $\left\{ \begin{array}{l} \text{数据说明} \\ \text{执行语句} \end{array} \right.$

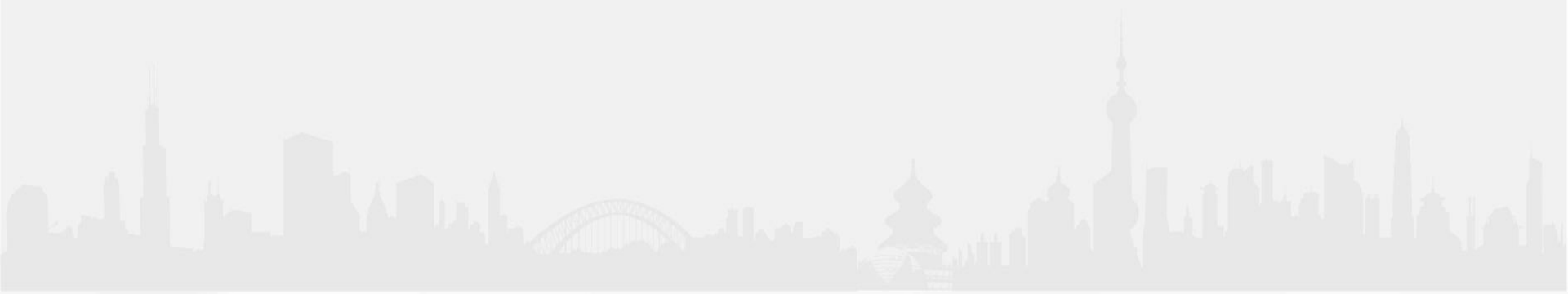
```
int max( int x, int y )  
{ int z ; /*本函数体内所用变量的说明*/  
  z=x>y ? x : y ;  
  return(z);  
}
```

函数体内若无任何语句时，为空函数。
不能在函数体内定义形参。

```
dump()  
{ }
```

Part.3

7.3 函数调用



7.3 函数调用

- 7.3.1 函数的返回值
- 7.3.2 函数调用的一般形式
- 7.3.3 函数的参数
- 7.3.4 对被调用函数的说明
- 7.3.5 数组名作为函数参数

7.3.1 函数的返回值

- 当被调用函数完成一定的功能后, 可将**处理的结果**返回到调用函数, 这种数据传送称为**函数的返回值**。如果函数有返回值, 在函数体内应包含**return**语句。
- 格式: **return (表达式);**
 return 表达式 ;
- 作用:
 - ★ 将表达式的值返回给调用函数
 - ★ 结束被调用函数的执行, 并把程序的控制返回到调用它的函数。

7.3.1 函数的返回值

如: **int max(int x, int y)**
 { int z ;
 z=x>y ? x : y ;
 return z ; ← 返回结果
 }

- 当没有返回值时, 可以写成:

— **return ;** ← 不返回结果
— **}** ← 不返回结果

7.3.1 函数的返回值

注意：

- 1) 函数的返回值的类型应与函数的类型**一致**。如不一致，**以函数类型为准**，**对返回值进行类型转换**，然后传送给调用函数。

如：

```
int f()  
{ return 3.5 ;  
}  
void main()  
{  
    int a=f() ;           /*a被初始化为3 */  
}
```


7.3.1 函数的返回值

注意：

2) 一个函数可以有多个return语句，但只可能执行其中一个。

例：

```
int max(int x, int y)
{
    if (x>y) return x ;
    else return y ;
}
```

7.3.1 函数的返回值

例：填空，完成下列函数。

(a) 计算函数值 $f(x)=\sin x+y^2$ ，函数名fs。

```
float fs( float x, float y)  
{ float z  
  z=sin(x)+y*y ;  
  return z ;  
}
```

7.3.1 函数的返回值

例：编写计算n!函数，函数名 **fact**。



```
float fact( int n)
{ float t=1.0 ; int i ;
  if (n<=0)
  { printf( “parameter error\n” ) ;
    return ; }
  for( i=2 ; i<=n ; i++ )
    t=t*i ;
  return t ;
}
```

主函数中可以这样调用：

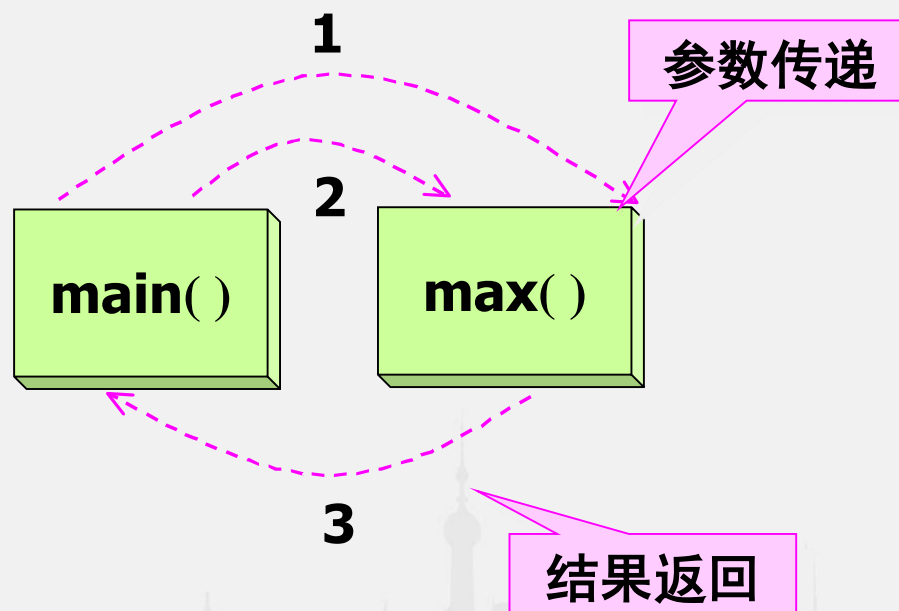
fact(5), fact(8)

7.3.2 函数调用的一般形式

📖 函数的使用是通过**函数调用**实现的。

📖 所谓函数调用就是**调用函数向被调函数 传送数据并将控制权**交给被调用函数，当**被调用函数执行完后，将结果回传给调用函数并交回控制权**。

```
int max( int x, int y )  
{  
    .....  
    return z;  
}  
void main()  
{  
    .....  
    c=max( a, b );  
}
```



7.3.2 函数调用的一般形式

被调用函数名([实参表])

- 实际参数简称“**实参**”，是一个**具有确定值**的表达式。函数在调用时，**将实参的值赋给对应的形参**。
- 实参表中的**实参个数、顺序及类型**应与被调用函数中的**形参一致**。

```
void main()  
{ int a, b, c;  
  scanf( "%d,%d" ,&a,&b);  
  c=max(a, b); ←.....实参  
  printf( "%d" ,c);  
}
```

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

7.3.2 函数调用的一般形式

- 实参可以是常量, 变量, 表达式、数组元素和数组名。
- 当实参表中有多个实参时, 对实参的求值顺序并不确定, 与所用系统有关。**VC是按从右往左的顺序求值。**

```
int max( int a, int b )  
{ int c=a>=b? a: b;  
  return c; }  
void main()  
{ int x=6, y;  
  y=max( ++x, x );  
  printf( "%d", y );  
}
```

7

```
b=x;  
a=++x;
```

在参数传递时

7.3.2 函数调用的一般形式

例 求 $1!+3!+5!+\dots+19!$ （调用函数）

- 函数调用过程:

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

```
float fact(int n);
```

```
void main()
```

```
{ float sum=0.0;
```

```
  int k;
```

```
  for( k=1; k<=19; k+=2 )
```

```
    sum=sum+fact(k);
```

```
  printf( "sum=%.1f\n",sum);
```

```
}
```

```
float fact(int n)
```

```
{ int i;
```

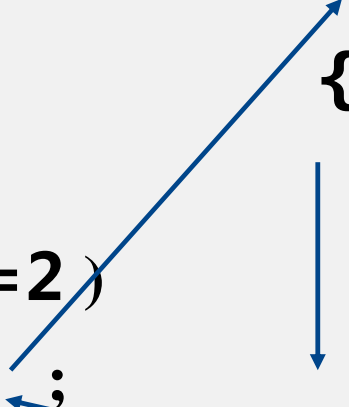
```
  float t =1.0;
```

```
  for(i=2; i<=n; i++)
```

```
    t = t*i;
```

```
  return t;
```

```
}
```



7.3.2 函数调用的一般形式

◆ 调用方式:

1) 函数表达式

如: `c=2*max(a, b);`

2) 作为语句用 (函数语句)

如: `printf(" * * * *"); fun();`

3) 作为函数参数

如: `m=max(a, max(b, c));`

`printf("%f\n" , max(a, b));`

7.3.3 函数的参数

 在调用函数时，大多数情况下，调用函数与被调用函数之间有**数据传递关系**，即实参与形参的结合。

正确地进行结合是函数调用的关键。结合时应**注意：实参与形参的个数相等，顺序一致，类型应相同。**

◆ 形参与实参间的数据传递

- 实参与形参结合的**原则**是：
 - 当实参为常量、变量、表达式或数组元素时，对应的形参只能是**变量名**。
 - 当实参为数组名时，所对应的形参必须是**同类型**的**数组名或指针变量**。

7.3.3 函数的参数

◆ 形参是变量名时的结合

- ✧ 实参和形参之间的数据传递采用单方向的“值传递”。
- ✧ 只能把实参值传给形参，而不能由形参传回来给实参。在内存中实参单元和形参单元是不同的存储单元。
- ✧ 因此，形参的值如发生改变，不会影响到调用函数中实参的值。

7.3.3 函数的参数

例 特点：单向传递，传递实参的值。

实参与形参占用不同的存储单元

```
#include
```

```
void change( int x, int y );
```

```
void main( )
```

```
{ int a, b ;
```

```
  a=2 ; b=3 ;
```

```
  change(a, b) ;
```

```
  printf( "a=%d b=%d\n" ,a,b);
```

```
}
```

```
void change( int x, int y )
```

```
{ int t ;
```

```
  t=x ; x=y ; y=t ;
```

```
}
```

main函数

a

1000H

2

b

1004H

3

(实参)

输出结果:

a=2 b=3

2000H

2004H

7.3.4 对被调用函数的说明

- 1) 被调用的函数必须存在
- 2) 标准库函数在调用前，应使用`#include`包含头文件。

如在使用输入输出函数时，应在文件开头用：

`#include <stdio.h>`

使用字符串处理函数，应该用：

`#include <string.h>`

使用数学库中的函数，应该用：

`#include <math.h>`

所有的数学函数返回double型的

7.3.4 对被调用函数的说明

3) 用户**自定义函数**在调用前, 必须对该函数进行声明。函数声明就是**函数原型**。

- 作用: 将函数名、函数类型及形参个数、顺序、类型告诉编译器。编译器用函数原型测试函数调用是否正确。

- 格式:

类型符 **函数名**(**类型** 形参1, ...);

或:

类型符 **函数名**(**参数类型**1, ...);

7.3.4 对被调用函数的说明

◆ 函数原型

- 函数原型要与函数的定义匹配
- 如果函数的定义在函数调用的后面就需要函数原型

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

```
float max(float ,float); ;
```

```
void main()  
{ float a,b,c;  
  scanf( "%f%f" , &a,&b);  
  c=max( a,b) ;  
  printf( "max is %f" , c) ;  
}
```

```
float max(float x, float y)  
{  
    return(x>y?x:y);  
}
```

7.3.4 对被调用函数的说明

◆ 函数原型

- 如果被调函数的定义出现在调用函数之前,可省略函数原型。

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

```
int max( int x, int y )
```

```
{ if (x>y) return x ; else return y ; }
```

```
void main( )
```

```
{ int  a, b, c ;
```

```
scanf( “%d%d” , &a, &b);
```

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

```
printf( “max=%d\n” , c);
```

```
}
```

改错:

```
#include<stdio.h>
void max( float , float) ;
void main( )
{ float  a, b, c ;
    scanf( "%f%f" , &a,&b);
    c=max( a b ) ;
}

void max( float x, float y)
{ float z , y;
    z=x>y ? x : y ;
    printf( "max is %.0f" , z ) ;
    return z ;
}
```



7.3.5 数组名作为函数参数

- 应在调用函数和被调函数中分别定义数组，并且实参数组与形参数组的**类型必须相同**，但**大小不要求一致**。

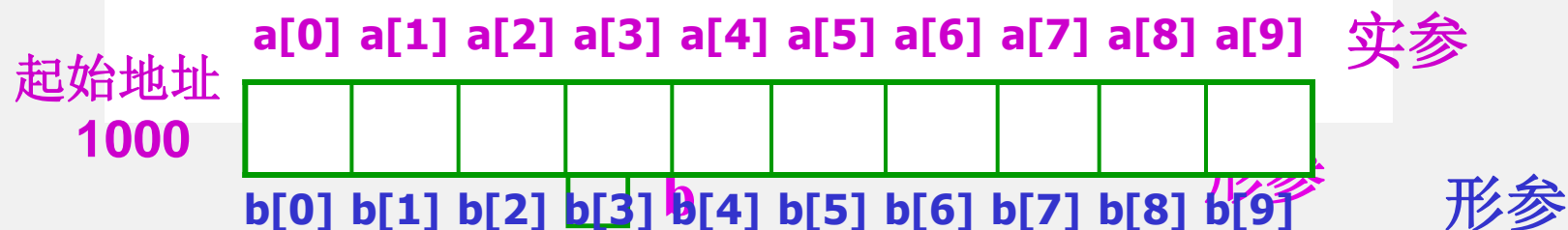
```
void sz1( int x[10] );  
void sz2( int y[4][5] );  
void main()  
{ int a[10], b[4][5];  
  ...  
  sz1 ( a );  
  sz2 ( b );  
  ...  
}
```

```
void sz1( int x[10] )  
{ ...  
}  
void sz2( int y[4][5] )  
{ ...  
}
```

7.3.5 数组名作为函数参数

- 由于**数组名**实际上是数组第一个元素的**地址**,调用函数时,用**数组名作实参时**,传送给形参是一个**地址值**,即实参数组的首地址,对应的形参应该是**数组名或一个指针变量**。
- 实参与形参之间的数据传递是**地址传递**。

7.3.5 数组名作为函数参数



```
void main()  
{ int a[10];  
  f(a);  
  ⋮  
}
```

```
void f( int b[10])  
{  
  ⋮  
}
```

数组名作为实参时，f 函数中对应的形参可用三种形式进行说明：

- ① f(int b[10]) ② f(int b[]) ③ f(int *b)

7.3.5 数组名作为函数参数

例 利用数组进行换数

```
#include<stdio.h>
void swap(int x[ ] );
```

```
void main( )
```

```
{ int i, a[2]={5,10};
```

```
    swap(a);
```

```
    for( i=0; i<2 ; i++ )
```

```
        printf( "a[%d]=%d\n" , i, a[i]);
```

```
} 输出结果: a[0]=10
```

```
          a[1]=5
```

```
void swap( int x[ ] )
```

```
{ int t;
```

```
    t=x[0]; x[0]=x[1];
```

```
    x[1]=t ;
```

```
}
```

a= 2000H a[0]

10

2004H a[1]

5

7.3.5 数组名作为函数参数

- 参数传递的方式：
 - 传递实参数组的地址
- 注意：
 - 形参数组的长度可以省略
 - 为了在被调用函数中处理数组元素的需要, 可另设一个参数传递数组元素的个数。

7.3.5 数组名作为函数参数

参数传递小结:

实 参	形 参	传递数据
基本变量 常 数 表 达 式 数组元素	基本变量	传 值
数组名	数组名 指针变量	传 址