西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

函数之间实现批量数据传递

1、用数组作为函数参数

2、数组参数的传递机制

6.3 数组与函数

- 可以将数组对象作为函数的参数，但与变量作为函数的参数有明显的区别。

6.3.1 数组作为函数的参数

- 一维数组元素可以直接作为函数实参使用，其用法与变量相同。假设有函数：

```
int max(int a,int b);
```

- 那么：

```
int A[5]={1,2,3,4,5} , c=2, x;
x=max(c,-10); //使用变量作为函数实参
x=max(A[2],-10); //使用数组元素作为函数实参
```

6.3.1 数组作为函数的参数

- C语言不允许数组类型作为函数类型，但可以作为函数的形参，称为形参数组。形参数组可以是一维数组，也可以是多维数组，基本形式为：

```
返回类型  函数名(类型  数组名[常量表达式],.....)
{
    函数体
}
```

6.3.1 数组作为函数的参数

- 示例

```
double average(double A[100],int n)
{
   ..... //函数体
}
```

6.3.1 数组作为函数的参数

- 函数形参如果是数组类型时，则调用实参就不能是元素，而必须是数组对象（数组名）。因为此时的形参是一个数据集合，所以实参也应该是一个数据集合，C语言不会将基本类型隐式转换为构造类型，反之亦不成立。

6.3.1 数组作为函数的参数

- 例如有函数原型：

```
double average(double A[100],int n);
```

- 则函数调用：

```
double x, y, A[100], B[2][100];
int P[100];
x=average(y,100); //错误，double不能对应double数组
x=average(A[10],100); //错误，double元素不能对应double数组
x=average(P,100); //错误，int数组不能对应double数组
x=average(A,100); //正确，double数组对应double数组
x=average(B[1],100); //正确，double数组对应double数组
```

6.3.2 数组参数的传递机制

- 前面讲过变量作为函数参数的传递机制是值传递，那么数组参数是否也是这样呢？
- 例如：

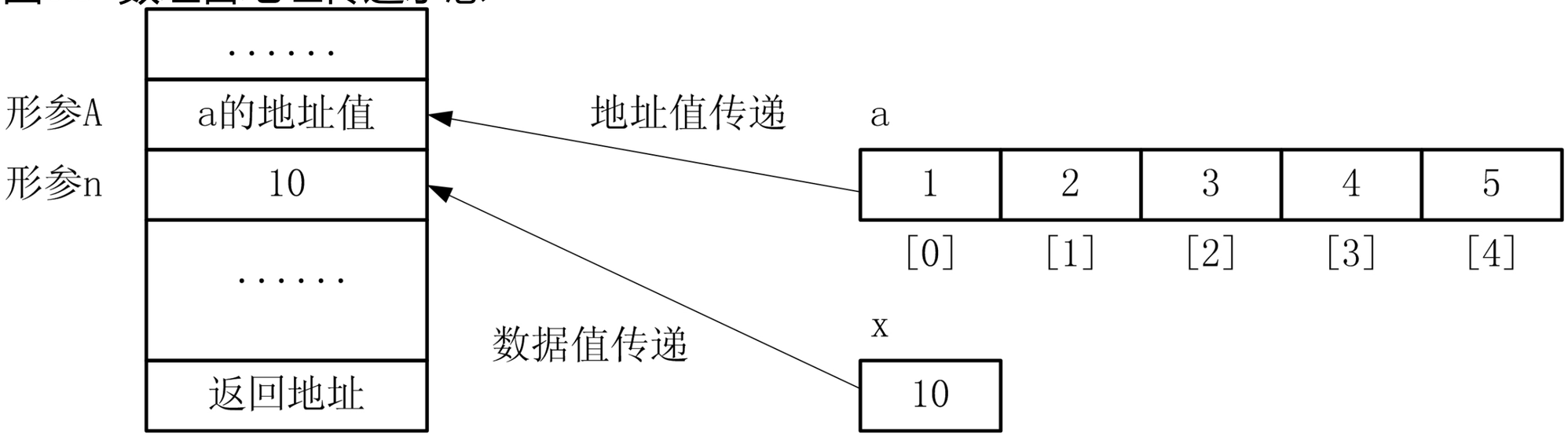
```
void fun(int A[10],int n);
int main()
{
    int a[10]={1,2,3,4,5} , x=5;
    fun(a,x); //实参分别是数组和整型变量
}
```

6.3.2 数组参数的传递机制

- 分析一下函数调用栈，实参的值是通过进栈方式传递到函数中去的，进栈必须有栈空间。
- 由于数组数据较多，如果采用一一进栈的方式，将使得函数在调用时光处理大批量数据的传递就要消耗无数的时间，这种方法显然不可取。
- 而且为巨大的数据集合再来一个副本，内存开销太大而且也无必要。
- 所以数组实参不是将每个元素一一传递到函数中。

6.3.2 数组参数的传递机制

图6.3 数组首地址传递示意



C语言处理数组实参，实际上是将数组的首地址传到了函数形参中。

6.3.2 数组参数的传递机制

- 尽管数组数据很多，但它们均从一个首地址连续存放，这个首地址对应的正是数组名。

- 如果实参使用数组名调用，本质上是将这个数组的首地址（一个数值），像变量实参那样值传递到形参中。

- 所以C语言传递数组时，依然是通过值传递方式。

6.3.2 数组参数的传递机制

- 不过尽管都是通过值传递，但变量与数组实参还是有很大的不同。
- 如图所示，变量x传的值是变量的数据值（10），这样形参n就是实参x的副本。数组实参a传的是数组首地址，形参A定义为数组形式，它现在的地址与实参数组a一样，则本质上形参数组对象A就是实参数组对象a（内存中两个对象所处位置相同，则它们实为同一个对象）。

6.3.2 数组参数的传递机制

- 当数组作为函数参数时，有下面的特殊性。
- （1）由于形参数组就是实参数组，所以在被调函数中使用形参就是在间接使用实参，这点与变量作为函数参数的情况是不同的。

6.3.2 数组参数的传递机制

- 例如：

```
void fun(int A[5],int n)
{
    A[1]=100; //A[1]实质就是实参a[1]
    n=10; //赋值给形参n，不影响实参x
}
void caller()
{
    int a[5]={1,2,3,4,5},x=5;
    fun(a,x);
    printf("%d,%d\n",a[1],x); //a[1]=100,x=5
}
```

6.3.2 数组参数的传递机制

- 实际编程中，可以用数组参数将被调函数处理过的数据返回到主调函数中。

6.3.2 数组参数的传递机制

【例6.6】

编写函数求一个二维数组中最大的元素及其下标。

6.3.2 数组参数的传递机制

例题分析

令max为元素最大值，采用枚举法逐一比较二维数组中的每一个元素A[i][j]和max，若A[i][j]大于max说明有一个更大的值出现，则令max=A[i][j]且记录r=i和c=j，遍历完所有元素，则A[r][c]就是最大的元素。由于max必然是数组中的一个元素值，且先比较才有max=A[i][j]，故设置max的初值为A中一个元素值，例如A[0][0]。

由于函数需要返回最大元素值及下标行、列三个数据，而函数返回只能是一个数据，所以使用数组B传递到函数中，将下标行、列值“带回”。

6.3.2 数组参数的传递机制

例6.6

```
#include<stdio.h>
int findmax(int A[3][4],int B[2])
{
  int i,j,max,r=0,c=0;
  max=A[r][c]; //max初值设为A[0][0]
  for (i=0; i<3; i++) //枚举二维数组所有元素
    for (j=0; j<4; j++)
      if (A[i][j]>max) {
        r = i , c = j; //记录此时的下标
        max = A[r][c]; //新的最大元素值;
      }
  B[0]=r, B[1]=c; //下标行、列通过B数组返回到主调函数中
  return max; //最大值通过函数值返回到主调函数中
}
int main()
```

6.3.2 数组参数的传递机制

例6.6

```
16 {
17    int A[3][4]={{7,5,-2,4},{5,1,9,7},{3,2,-1,6}}, B[2], max;
18    max=findmax(A,B);
19    printf("max:A[%d][%d]=%d\n",B[0],B[1],max);
20    return 0;
21 }
```

6.3.2 数组参数的传递机制

例6.6

程序运行屏幕

```
max:A[1][2]=9
```

6.3.2 数组参数的传递机制

- （2）既然形参数组对象就是实参数组对象，所以函数定义中的形参数组就不像变量那样建立一个数组副本，即函数调用时不会为形参数组分配存储空间。
- 形参数组不过是用数组定义这样的形式来表明它是个数组，能够接收实参传来的地址，形参数组的长度说明也无实际作用。

6.3.2 数组参数的传递机制

- 因此，形参数组的长度与实参数组长度可以不相同，形参数组的长度可以是任意值，形参数组甚至可以不用给出长度。

6.3.2 数组参数的传递机制

- 假设有函数调用：

```
int a[15];
f(a);
```

- 则以下函数定义：

```
void f(int A[100]);
//形参数组长度完全由实参数组确定，因此函数中并不能按100个元素处理
void f(int A[10]);
//形参数组长度完全由实参数组确定，因此函数中并不能按10个元素处理
void f(int A[]); //表明形参是数组形式即可
```

- 均是正确的。

6.3.2 数组参数的传递机制

- （3）虽然实参数组将地址传到了被调函数中，但被调函数并不知道实参数组的具体长度，那么假定的大小对于实参数组来说容易数组越界。
- 实际编程中可以采用下面两个方法来解决：
- ①函数调用时再给出一个参数来表示实参数组的长度；
- ②在实参数组中（一般是末尾）放上一个约定条件的数据，被调函数只要遇到这样的数据就结束对数组的遍历。

6.3.2 数组参数的传递机制

【例6.7】

编写average函数求一组数据的平均值。

6.3.2 数组参数的传递机制

例题分析

为了让average函数能够适用于任意长度的数组，需要将数组的长度当作一个参数传入到函数中。

6.3.2 数组参数的传递机制

例6.7

```
#include<stdio.h>
double average(double A[],int n)
{
  int i; double s=0; //累加初值为0
  for (i=0; i<n; i++) s=s+A[i]; //先累加
  return n!=0 ? s/n : 0.0; //计算平均值
}
int main()
{
  double A[3]={1,2,3};
  double B[5]={1,2,3,4,5};
  printf("A=%lf\n",average(A,3)); //传递数组长度即可正确计算
  printf("B=%lf\n",average(B,5)); //传递数组长度即可正确计算
  return 0;
}
```

6.3.2 数组参数的传递机制

例6.7

程序运行屏幕

```
A=2.000000
B=3.000000
```

```
1 #include<stdio.h>
,3)); //传递数组长度即可正确计算
,5)); //传递数组长度即可正确计算
```

6.3.2 数组参数的传递机制

- （4）多维数组作为函数的参数，形参数组第1维可以与实参相同，也可以不相同；可以是任意长度，也可以不写长度；但其他维的长度需要相同。因为编译器是根据形参来检查实参调用的，它可以忽略第1维的长度大小，但其他维的长度由于决定了形参数组的结构而不能忽略。

- 编译器不能对不同结构的数组类型作隐式转换。

6.3.2 数组参数的传递机制

- 例如有函数调用：

```
int a[5][10]
f(a);
```

- 则函数定义：

```
void f(int A[5][10]); //正确
void f(int A[2][10]); //正确
void f(int A[][10]); //正确
void f(int A[][]); //错误，第2维长度必须给出
void f(int A[5][5]); //错误，第2维长度必须相同
void f(int A[50]); //错误，必须是二维数组
```


程序设计