



计算机导论与程序设计——第9篇

函数与模块化编程

Computer Introduction and Programming

学习目标



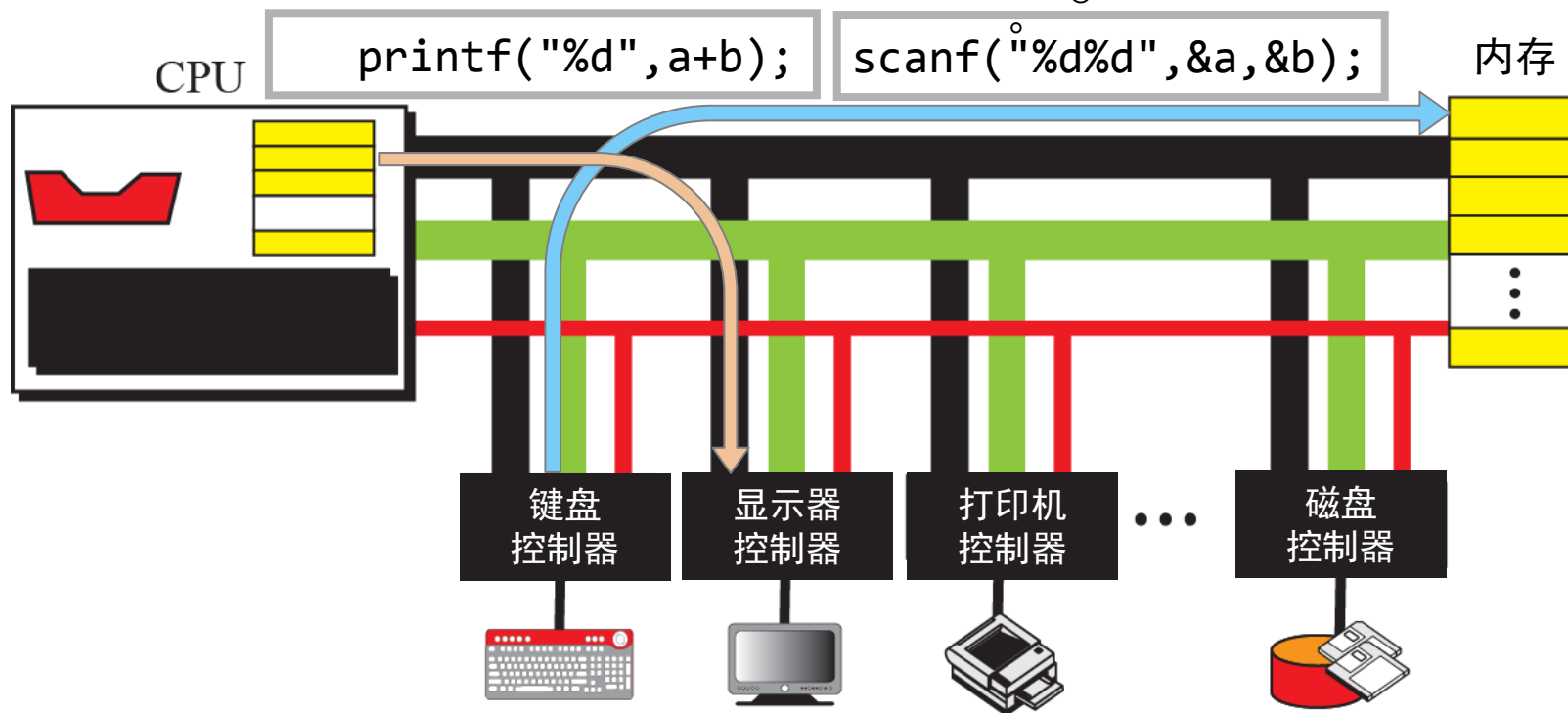
- 了解模块化编程
- 熟悉函数的基本概念
- 掌握函数的定义、声明和调用
- 掌握参数传递机制和变量的作用域
- 熟悉递归方法
- 熟悉C语言常用库函数

为什么要用函数

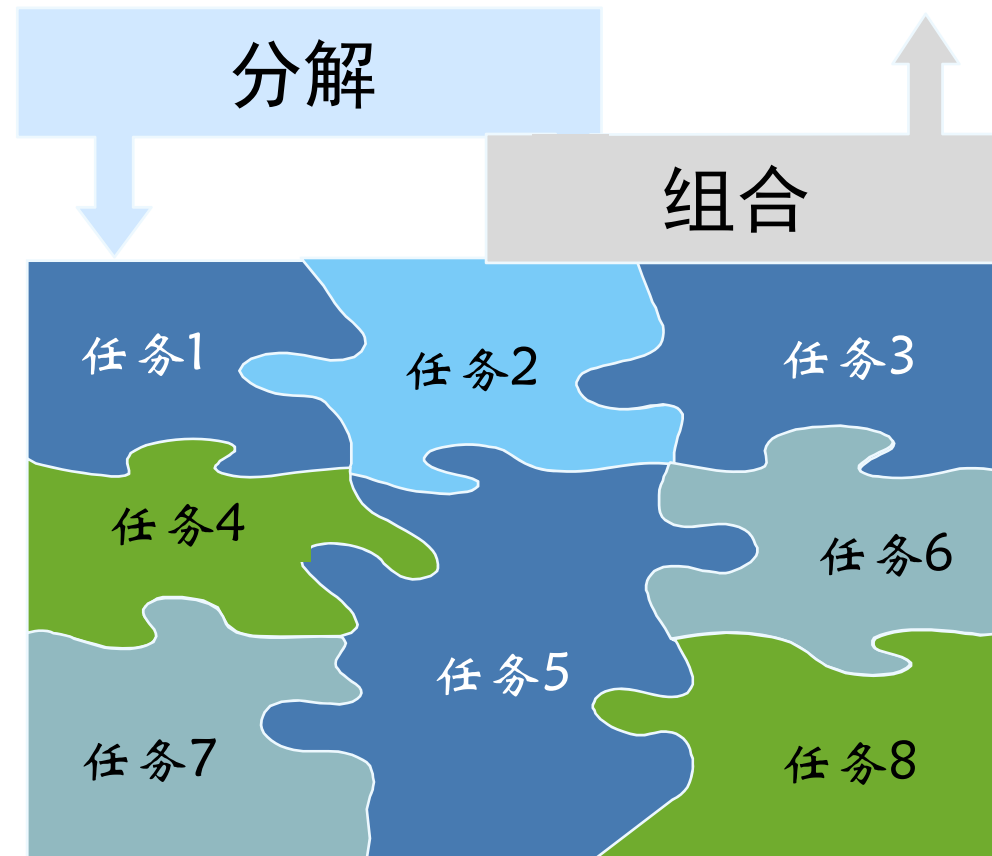
复习

函数就是功能。每一个函数用来实现一个特定的功能。

函数



模块化设计思想



开发和构建大型程序的最好方法就是用较小的程序片段或模块来构建它——分而治之

模块化设计思想

模块

模块：能完成指定功能的子程序。C 语言中称之为“函数”

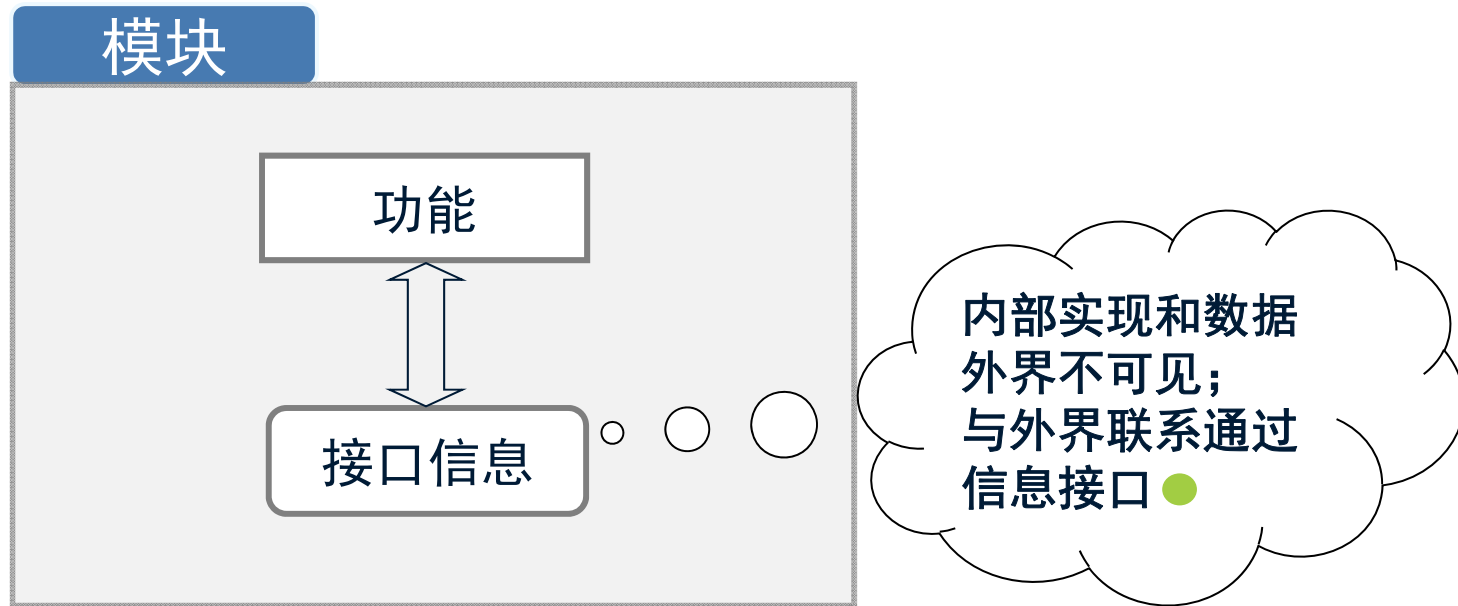
模块复用

可以将重复调用的功能独立成一个模块

多模块结构

将程序分为不同的模块。每一模块实现不同的功能。

模块化设计思想



接口信息

是其他模块或程序使用该模块的约定方式，
包括输入输出信息等

函数的定义

函数定义格式

设计

```
/******  
函数功能： 描述函数实现的功能  
函数参数： 解释所有输入数据的含义  
函数返回值： 解释函数处理的结果和其类型  
*****/
```

类型标识符 函数名（形式参数表）

```
{  
  
}  
语句部分; // 函数功能的实现部分，也称函数体
```

类型标识符—说明函数返回值的类型(即函数结束后return给主调函数的值的类型);

函数名—由用户定义，应符合标识符的规则。

形式参数表—由若干变量及其类型说明符构成，也可以无参数。

函数设计三要素

函数设计要素

函数功能	输入信息	输出信息
函数功能用函数名 简要描述	输入信息的性质与个数 决定形参表的形式	输出信息的类型决 定函数的类型
函数名	形参表	函数类型

函数的例子

定义一个用于比较两个整数大小并返回较大值的函数。

```
1  /*****
2  函数功能： 比较两个整数大小，并返回较大值
3  函数参数： 两个整型数x和y
4  函数返回值： 返回两个整型数的较大值
5  *****/
6  int max(int x, int y) {
7      if (x > y)
8          return x;
9      return y;
10 }
```

函数的调用方式

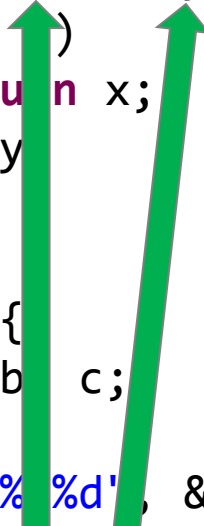
函数名(实参1, 实参2, ..., 实参n);

说明:

1. 实参可以是常量、变量或表达式，但必须有确定的值。
2. 实参的个数、类型应与函数定义时的形参完全一致，在函数调用时这些参数的值会传递给函数的形参。
3. 被调用的函数定义应出现在主调函数之前。否则，需要对函数进行声明，声明格式为：类型标识符 函数名(形参列表);
4. 如果是标准函数库中的函数，则需在调用前包含相应的头文件。

【例9.1】输入两个整数，用一个函数求出大数，然后输出。

```
1 #include<stdio.h>
2
3 int max(int x,int y) {
4     if (x > y)
5         return x;
6     return y;
7 }
8
9 int main() {
10     int a, b, c;
11
12     scanf("%d %d", &a, &b);
13     c = max(a, b);
14     printf("%d\n", c);
15 }
```

Two green arrows originate from the arguments 'a' and 'b' in the function call 'max(a, b)' on line 13. One arrow points vertically upwards to the parameter 'x' in the function definition 'max(int x, int y)' on line 3. The other arrow points diagonally upwards and to the right to the parameter 'y' on line 3. This illustrates the passing of actual arguments to formal parameters.

主函数中包含了一个函数调用`max(a,b)`。`max`后面括号内的`a`和`b`是实参。`a`和`b`是在`main`函数中定义的变量，`x`和`y`是函数`max`的形式参数。通过函数调用，在两个函数之间发生数据传递，实参`a`和`b`的值传递给形参`x`和`y`。

【例9.1】输入两个整数，用一个函数求出大数，然后输出。

```
1 #include<stdio.h>
2
3 int max(int x,int y) {
4     if (x > y)
5         return x;
6     return y;
7 }
8
9 int main() {
10     int a, b, c;
11
12     scanf("%d%d", &a, &b);
13     c = max(a, b);
14     printf("%d\n", c);
15 }
```

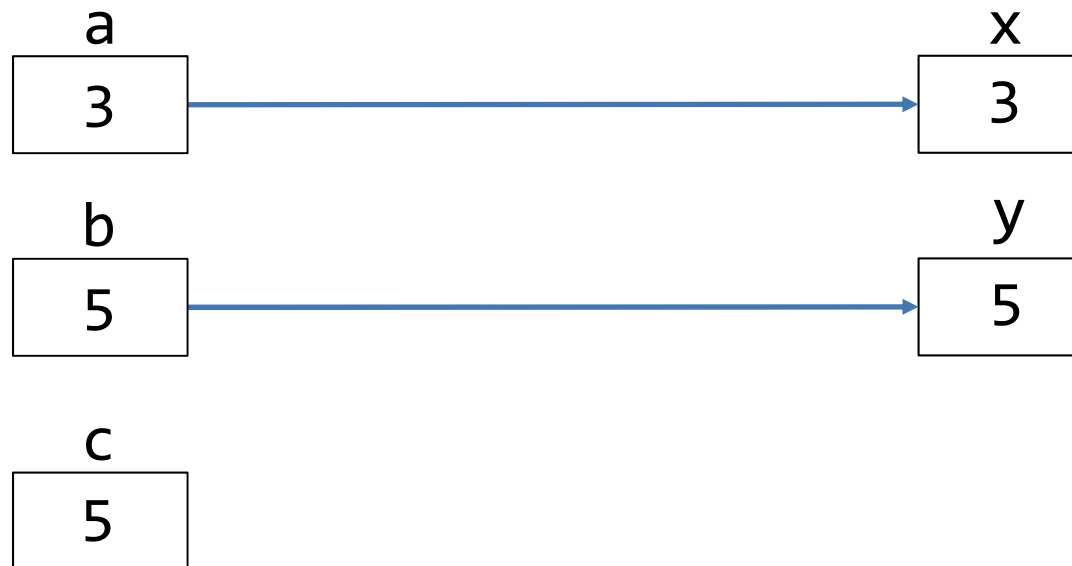


在max函数中把x和y中的大数作为函数值返回main函数，赋给变量c。

调用过程

```
9 int main() {  
10     int a, b, c;  
11  
12     scanf("%d%d", &a, &b);  
13     c = max(a, b);  
14     printf("%d\n", c);  
15 }
```

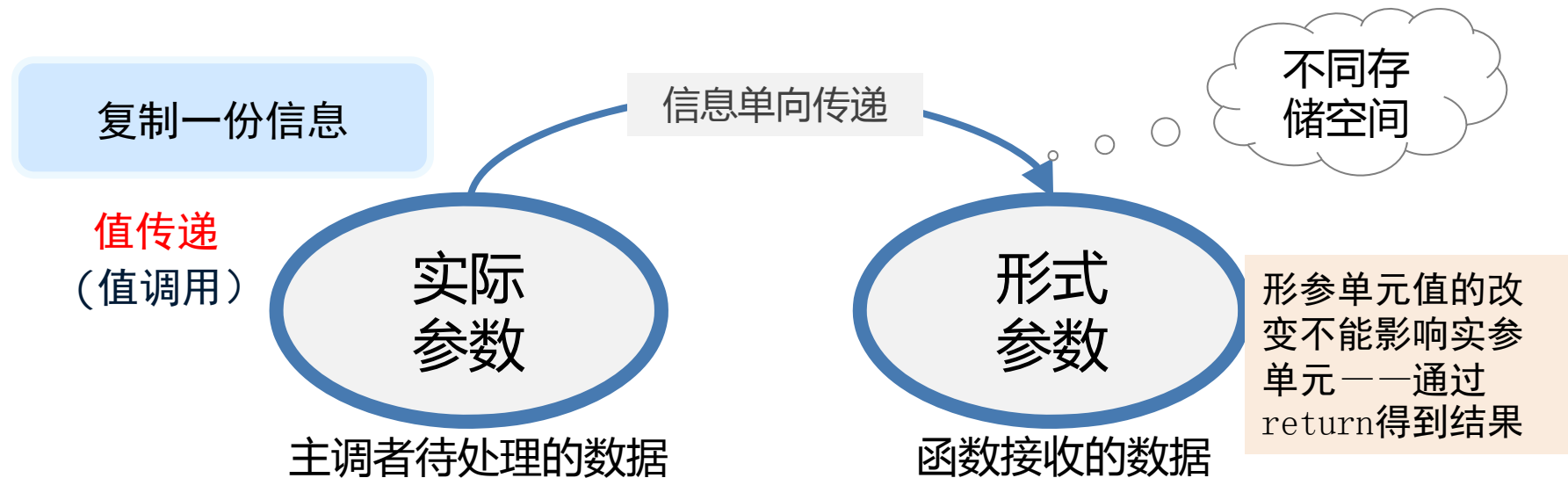
```
3 int max(int x, int y) {  
4     if (x > y)  
5         return x;  
6     return y;  
7 }
```



参数传递

```
12 scanf("%d%d", &a, &b);  
13 c = max(a, b);  
14 printf("%d\n", c);  
15 }
```

```
3 int max(int x, int y) {  
4     if (x > y)  
5         return x;  
6     return y;  
7 }
```



函数的三种格式

函数声明格式

简介

函数类型 函数名（形式参数表）;

函数调用格式

实现

函数名（实际参数表）

函数定义格式

设计

```
/******  
函数功能： 描述函数实现的功能  
函数参数： 解释所有输入数据的含义  
函数返回值： 解释函数处理的结果和其类型  
*****/
```

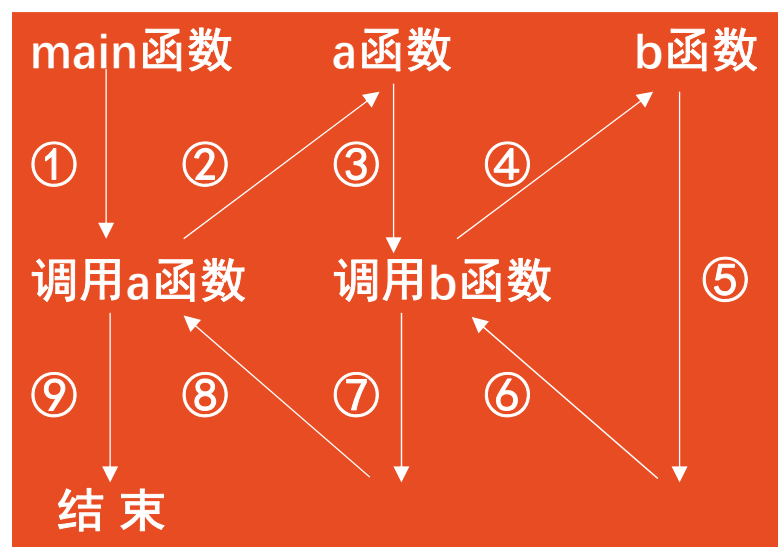
```
函数类型 函数名（形式参数表）  
{  
    语句部分;  
}
```

【例9.2】输入两个实数，用一个函数求出它们之和。

```
1 #include <stdio.h>
2 float add(float x, float y) { //定义add函数
3     float z;
4     z = x + y;
5     return (z); //把变量z的值作为函数值返回
6 }
7
8 int main() {
9
10     float a, b, c;
11
12     printf("Please enter a and b:"); //提示输入
13     scanf("%f%f", &a, &b); //输入两个实数
14     c = add(a, b); //调用add函数
15     printf("sum is %f\n", c); //输出两数之和
16     return 0;
17 }
18
```


函数的嵌套调用

C语言的函数可以嵌套调用函数，即在调用一个函数的过程中，又调用另一个函数。



【例9.3】输入4个整数，找出其中最大的数。用函数的嵌套调用来处理。

```
1 #include <stdio.h>
2 int max2(int a,int b) { // 定义max2函数
3     if (a>=b)
4         return a; // 若a>=b, 将a为函数返回值
5     return b; // 若a<b, 将b为函数返回值
6
7 }
8
9 int max4(int a,int b,int c,int d) { // 定义max4函数
10
11     int m;
12     m=max2(a,b); // 调用max2函数, 得到a和b两个数中的大者, 放在m中
13     m=max2(m,c); // 调用max2函数, 得到a,b,c三个数中的大者, 放在m中
14     m=max2(m,d); // 调用max2函数, 得到a,b,c,d四个数中的大者, 放在m中
15     return (m); // 把m作为函数值带回main函数
16 }
17
```

【例9.3】输入4个整数，找出其中最大的数。用函数的嵌套调用来处理。

```
18
19 int main() {
20
21     int a,b,c,d,max;
22     printf("Please enter 4 interger numbers:"); // 提示输入4个数
23     scanf("%d%d%d%d",&a,&b,&c,&d); // 输入4个数
24     max=max4(a,b,c,d); // 调用max4函数，得到4个数中的最大者
25     printf("max=%d\n",max); // 输出4个数中的最大者
26     return 0;
27 }
28
```

函数的递归调用

(回顾) 迭代与递归

迭代和递归是两种在计算机科学中广泛使用的基本设计方法，虽然两者都用于重复性操作的处理，但是两种方法有各自的特点。

迭代利用计算机运算速度快、适合做重复性操作的特点，让计算机对一组操作进行重复执行，在每次执行这组操作时，都将从原值推出它的一个新值。

(回顾)迭代与递归

例如，迭代求 $1+2+3\dots+n$ 的过程，设 $n=5$ ， $\text{sum}=0$

$$0 + 1 = 1$$

$$1 + 2 = 3$$

$$3 + 3 = 6$$

$$6 + 4 = 10$$

$$10 + 5 = 15$$

(回顾)迭代与递归

递归则是一种算法自我调用的过程。当一个算法在执行过程中，又直接或间接调用了自身算法，这个过程就是递归。

使用递归需具备以下条件：

1. 原问题可以通过转化为较小的子问题来解决，而子问题的求解方法与原问题相同，被处理的数据有规律地减少。
2. 当子问题小至一定程度时，调用自身算法的过程会终止。

(回顾)迭代与递归

例如，用递归求 $1+2+3\dots+n$ 的过程，设 $n=5$ ， $sum=0$

$sum(5)$

$5+sum(4)$

$5+4+sum(3)$

$5+4+3+sum(2)$

$5+4+3+2+sum(1)$

$5+4+3+2+1+sum(0)$

$5+4+3+2+1+0$

$5+4+3+2+1$

$5+4+3+3$

$5+4+6$

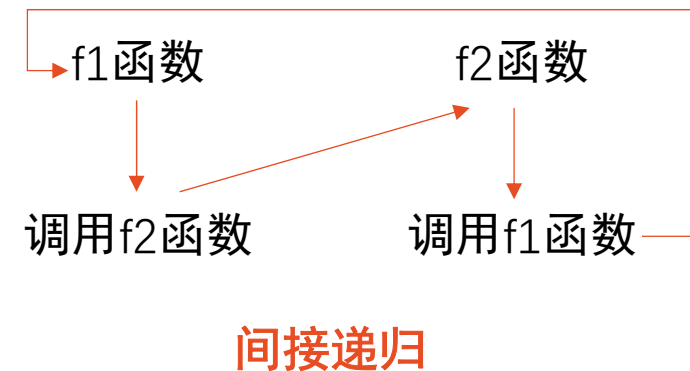
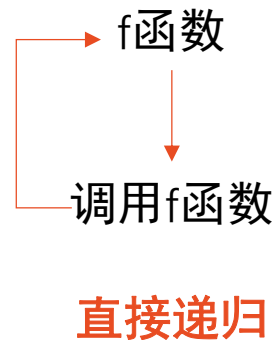
$5+10$

15

函数的递归调用

在调用一个函数的过程中又出现直接或间接地调用该函数本身，称为函数的递归调用。

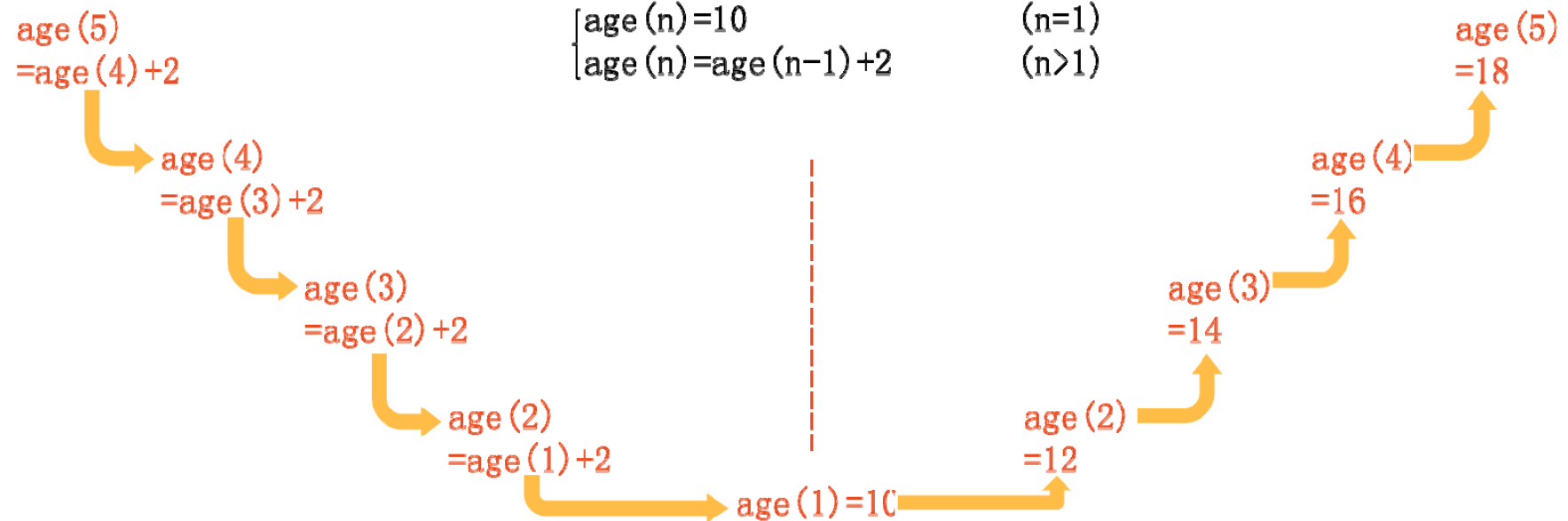
```
int f(int x)
{
    int y,z;
    z = f(y); //在执行f函数的过程中又要调用f函数
    return (2*z);
}
```



程序中不应出现无终止的递归调用，而只应出现有限次数的、有终止的递归调用，这可以用if语句来控制，只有在某一条件成立时才继续执行递归调用；否则就不再继续。

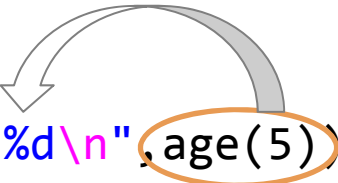
【例9.4】有5个学生坐在一起，问第5个学生多少岁，他说比第4个学生大2岁。问第4个学生岁数，他说比第3个学生大2岁。问第3个学生，又说比第2个学生大2岁。问第2个学生，说比第1个学生大2岁。最后问第1个学生，他说是10岁。请问第5个学生多大。

解题思路：



【例9.4】有5个学生坐在一起，问第5个学生多少岁，他说比第4个学生大2岁。问第4个学生岁数，他说比第3个学生大2岁。问第3个学生，又说比第2个学生大2岁。问第2个学生，说比第1个学生大2岁。最后问第1个学生， he说是10岁。请问第5个学生多大。

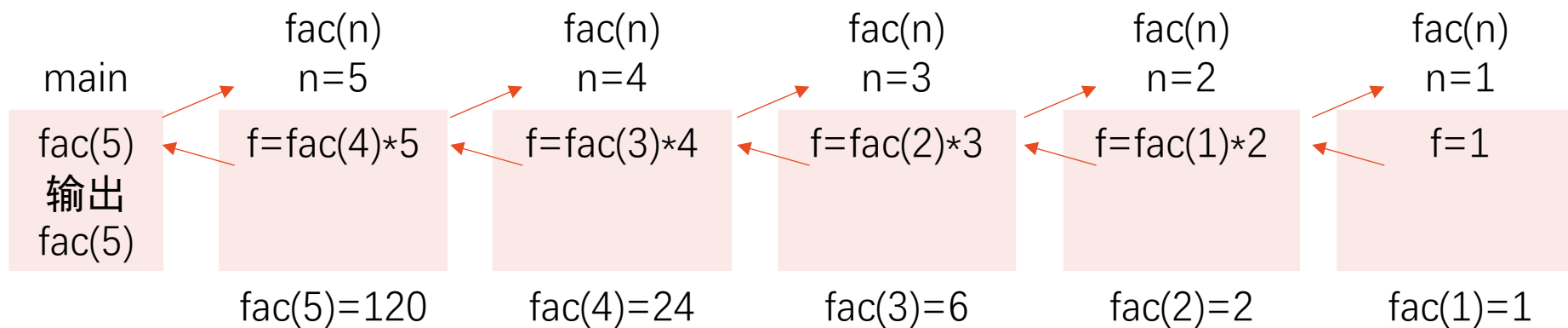
```
1 #include <stdio.h>
2
3 int age(int n) { //定义递归函数
4     int c;
5     if (n==1) // 如果n等于1
6         c=10; // 年龄为10
7     else // 如果n不等于1
8         c=age(n-1)+2; // 年龄是前一个人的年龄加2
9     return c; // 返回年龄
10 }
11 int main() {
12
13     printf("NO.5, age:%d\n", age(5)); //输出第5人的年龄
14     return 0;
15 }
```



【例9.5】用递归方法求n!。

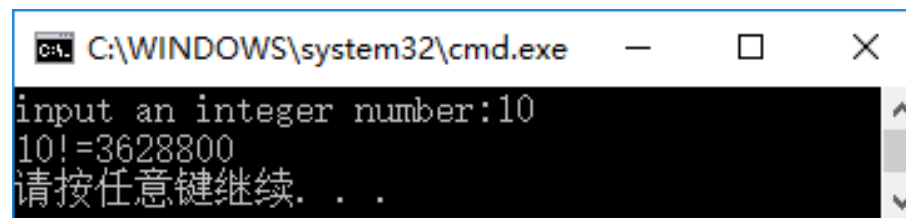
解题思路:

$$n! = \begin{cases} n! = 1 & (n = 0, 1) \\ n \cdot (n-1)! & (n > 1) \end{cases}$$



【例9.5】用递归方法求n!。

```
1 #include <stdio.h>
2 int fac(int n) {
3     int f;
4     if (n<0)
5         printf("n<0,data error!");
6     else if (n==0||n==1)
7         f=1;
8     else f=fac(n-1)*n;
9     return (f);
10 }
11 int main() {
12
13     int n;
14     int y;
15     printf("input an integer number:");
16     scanf("%d",&n);
17     y=fac(n);
18     printf("%d!=%d\n",n,y);
19     return 0;
20 }
21
```



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window has a black background with white text. It shows the program's output: "input an integer number:10", "10!=3628800", and "请按任意键继续. . .". The window has standard Windows window controls (minimize, maximize, close) in the title bar.

数组作为函数参数

【例9.6】输入10个数，要求输出其中值最大的元素和该数是第几个数。

```
1 #include <stdio.h>
2 int max(int x,int y) {
3     if (x > y)
4         return x;
5     return y;
6 }
7 int main() {
8     int a[10],n,t,i;
9     for (i = 0; i < 10; i++)
10         scanf("%d",&a[i]);
11     m = a[0];
12     n = 0;
13     for (i = 1; i < 10; i++) {
14         t = max(m, a[i]);
15         if (t > m) { // 若max函数返回的值大于m
16             m = t; // max函数返回的值取代m原值
17             n = i; // 把此数组元素的序号记下来，放在n中
18         }
19     }
20     printf("%d %d\n",m,n+1);
21     return 0;
22 }
23 }
```

数组元素可以用作函数实参,通过单向值传递的方式传给形参。

【例9.7】有一个一维数组score，内放10个学生成绩，求平均成绩。

```
1 #include <stdio.h>
2 float average(float array[10]) {// 定义average函数
3     int i;
4     float aver,sum=0;
5     for (i = 0; i < 10; i++)
6         sum=sum+array[i]; // 累加学生成绩
7     aver=sum/10;
8     return (aver);
9 }
10 int main() {
11     float score[10],aver;
12     int i;
13     for (i = 0; i < 10; i++)
14         scanf("%f", &score[i]);
15     printf("\n");
16     aver=average(score); // 调用average函数
17     printf("average score is %5.2f\n",aver);
18     return 0;
19 }
```

是值传递吗？

数组的存储

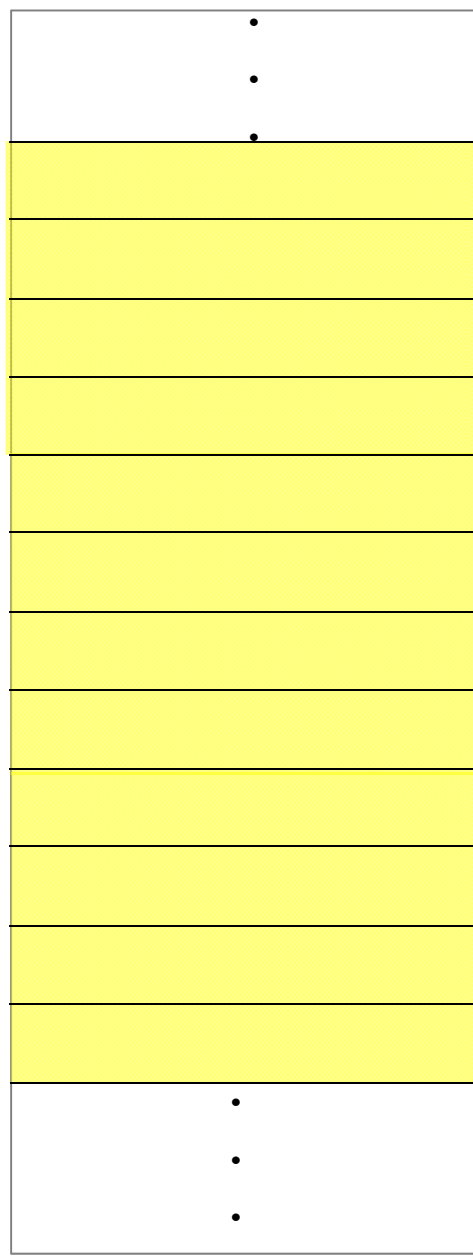
```
int score[80];
```

地址



2 0 0 0
2 0 0 1
2 0 0 2
2 0 0 3
2 0 0 4
2 0 0 5
2 0 0 6
2 0 0 7
2 0 0 8
2 0 0 9
2 0 0 A
2 0 0 B

在C语言中数组名代表该数组第一个元素的地址(数组的起始地址)



score[0]

score[1]

score[2]

【例9.7】有一个一维数组score，内放10个学生成绩，求平均成绩。

```
1 #include <stdio.h>
2 float average(float array[10]) {// 定义average函数
3     int i;
4     float aver,sum=0;
5     for (i = 0; i < 10; i++)
6         sum=sum+array[i];           // 累加学生成绩
7     aver=sum/10;
8     return (aver);
9 }
10 int main() {
11     float score[10],aver;
12     int i;
13     for (i = 0; i < 10; i++)
14         scanf("%f", &score[i]);
15     printf("\n");
16     aver=average(score);           // 调用average函数
17     printf("average score is %5.2f\n",aver);
18     return 0;
19 }
```

传递的是数组
的起始地址

【例9.7】有一个一维数组score，内放10个学生成绩，求平均成绩。

```
1 #include <stdio.h>
2 float average(float array[10]) {
3     int i;
4     float aver,sum=0;
5     for (i = 0; i < 10; i++)
6         sum=sum+array[i];
7     aver=sum/10;
8     return (aver);
9 }
10 int main() {
11     float score[10],aver;
12     int i;
13     for (i = 0; i < 10; i++)
14         scanf("%f", &score[i]);
15     printf("\n");
16     aver=average(score);
17     printf("average score is %5.2f\n",aver);
18     return 0;
19 }
```

2	0	0	0
2	0	0	1
2	0	0	2
2	0	0	3
2	0	0	4
2	0	0	5
2	0	0	6
2	0	0	7
2	0	0	8
2	0	0	9
2	0	0	A
2	0	0	B

score[0]
array[0]

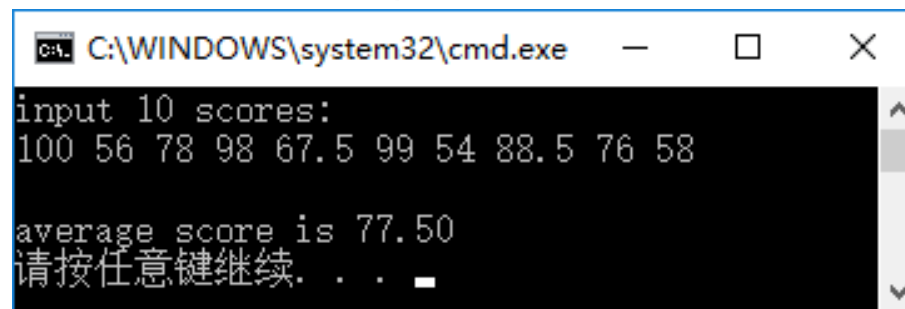
score[1]
array[1]

score[2]
array[2]

【例9.7】有一个一维数组score，内放10个学生成绩，求平均成绩。

```
1 #include <stdio.h>
2 float average(float array[]) {// 定义average函数
3     int i;
4     float aver,sum=0;
5     for (i = 0; i < 10; i++)
6         sum=sum+array[i];
7     aver=sum/10;
8     return (aver);
9 }
10 int main() {
11     float score[10],aver;
12     int i;
13     for (i = 0; i < 10; i++)
14         scanf("%f", &score[i]);
15     printf("\n");
16     aver=average(score);    // 调用average函数
17     printf("average score is %5.2f\n",aver);
18     return 0;
19 }
```

形参数组可以不指定大小，在定义数组时在数组名后面跟一个空的方括号。

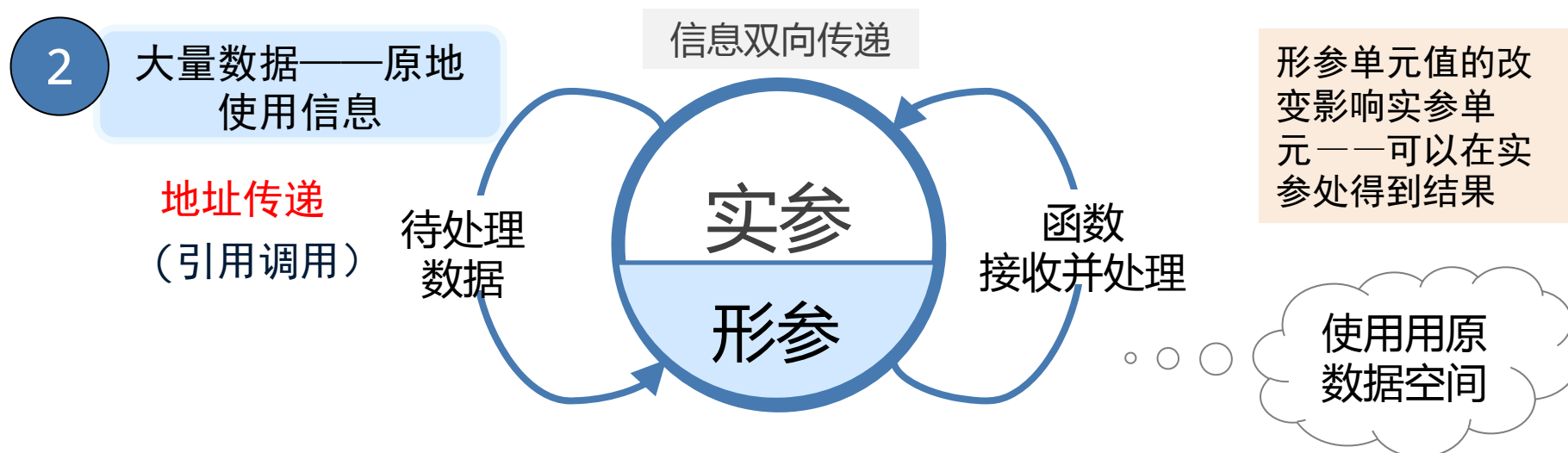
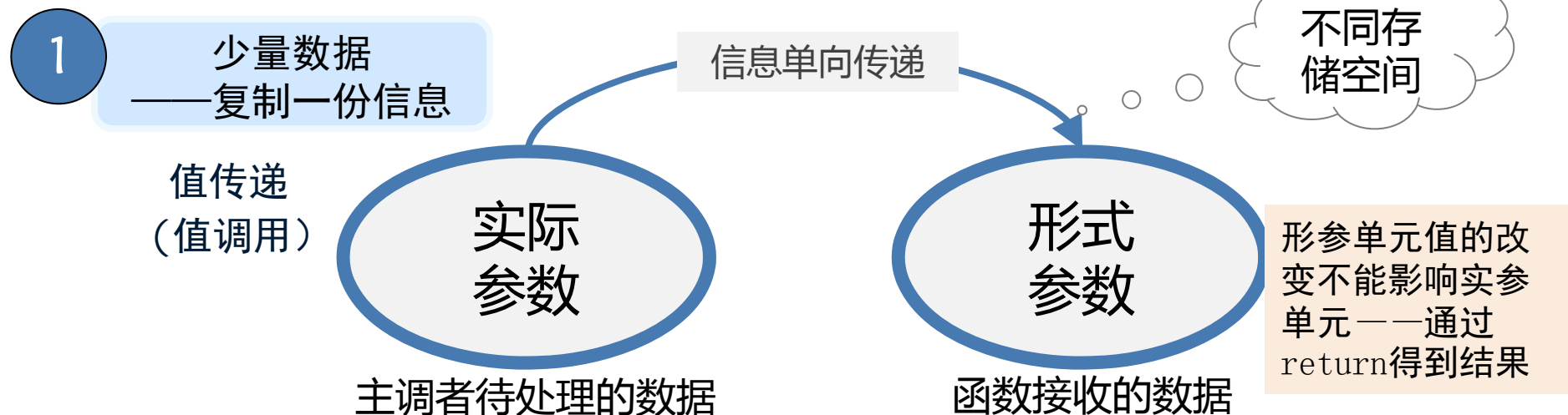


```
C:\WINDOWS\system32\cmd.exe
input 10 scores:
100 56 78 98 67.5 99 54 88.5 76 58
average score is 77.50
请按任意键继续. . .
```

【例9.8】有两个班级，分别有5和10名学生，调用average函数，分别求这两个班的学生的平均成绩。

```
1 #include <stdio.h>
2 float average(float array[ ],int n) {
3     int i;
4     float aver,sum=0;
5     for (i=0; i<n; i++)
6         sum=sum+array[i];
7     aver=sum/n;
8     return (aver);
9 }
10 int main() {
11     float score1[5]= {98.5,97,91.5,60,55};
12     float score2[10]= {67.5,89.5,99,69.5,77,89.5,76.5,54,60,99.5};
13     printf("The average of class A is %6.2f\n",average(score1,5));
14     printf("The average of class B is %6.2f\n",average(score2,10));
15     return 0;
16 }
```

函数间信息传递的方法



【例9.7】有一个一维数组score，内放10个学生成绩，求平均成绩。

```
1 #include <stdio.h>
2 float average(float array[10]) {
3     int i;
4     float aver, sum=0;
5     for (i = 0; i < 10; i++)
6         sum=sum+array[i];
7     aver=sum/10;
8     return (aver);
9 }
10 int main() {
11     float score[10], aver;
12     int i;
13     for (i = 0; i < 10; i++)
14         scanf("%f", &score[i]);
15     printf("\n");
16     aver=average(score);
17     printf("average score is %5.2f\n", aver);
18     return 0;
19 }
```

2 0 0 0	•
2 0 0 1	•
2 0 0 2	•
2 0 0 3	•
2 0 0 4	•
2 0 0 5	•
2 0 0 6	•
2 0 0 7	•
2 0 0 8	•
2 0 0 9	•
2 0 0 A	•
2 0 0 B	•
	•
	•

score[0]
array[0]

score[1]
array[1]

score[2]
array[2]

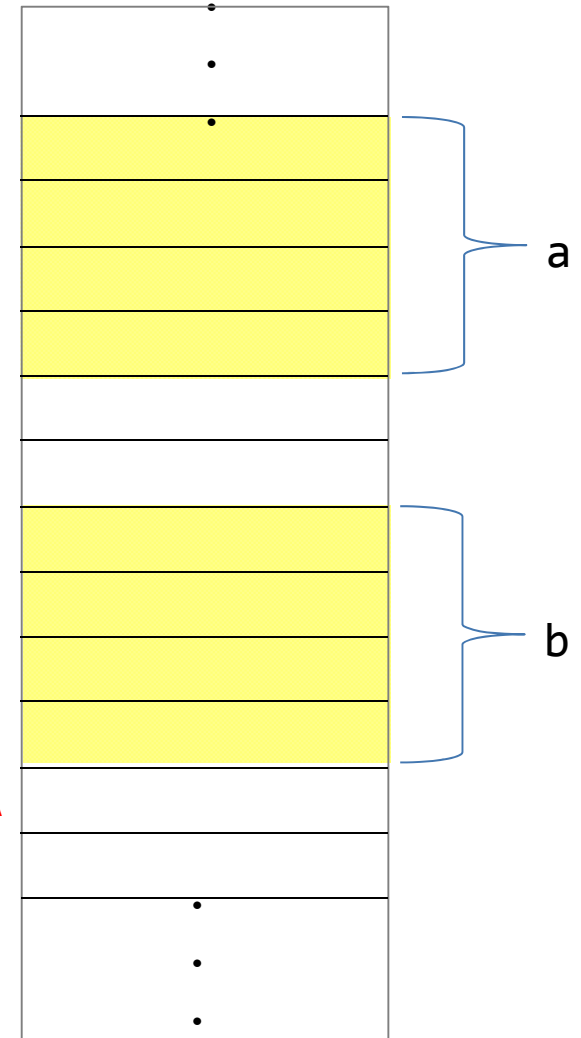
函数间信息传递的方法分析

```
1 #include <stdio.h>
2 int main() {
3     int a, b;
4
5     scanf("%d%d", &a, &b);
6
7     printf("%d+%d=%d\n", a, b, a+b);
8
9     return 0;
10 }
```

地址传递

单向值传递

5 0 0 0
5 0 0 1
5 0 0 2
5 0 0 3
5 0 0 4
5 0 0 5
5 0 0 6
5 0 0 7
5 0 0 8
5 0 0 9
5 0 0 A
5 0 0 B



【例9.9】用选择排序算法对数组中5个整数按由小到大排序。

解题思路:

所谓选择法就是先将5个数中最小的数与a[0]对换;再将a[1]~a[4]中最小的数与a[1]对换……每比较一轮,找出一个未经排序的数中最小的一个。共比较4轮。

a[0]	a[1]	a[2]	a[3]	a[4]	
3	6	1	9	4	未排序时的情况
1	6	3	9	4	将5个数中最小的数1与a[0]对换
1	3	6	9	4	将余下的后面4个数中最小的数3与a[1]对换
1	3	4	9	6	将余下的3个数中最小的数4与a[2]对换
1	3	4	6	9	将余下的2个数中最小的数6与a[3]对换, 完成排序

【例9.9】用选择排序算法对数组中10个整数按由小到大排序。

```
1 #include <stdio.h>
2 void sort(int array[],int n) {
3     int i,j,k,t;
4     for (i = 0; i < n-1; i++) {
5         k = i;
6         for (j = i+1; j < n; j++) {
7             if (array[j] < array[k])
8                 k = j;
9         }
10        t =array[k];
11        array[k]=array[i];
12        array[i]=t;
13    }
14 }
15
```

【例9.9】用选择排序算法数组中10个整数按由小到大排序。

```
16 int main() {  
17     int a[5],i;  
18     printf("enter array:\n");  
19     for (i = 0; i < 5; i++)  
20         scanf("%d",&a[i]);  
21     sort(a,5);  
22     printf("The sorted array:\n");  
23     for (i = 0; i < 5; i++)  
24         printf("%d ",a[i]);  
25     printf("\n");  
26     return 0;  
27 }  
28
```