实验八 指针与数组（1）

## 【目的与要求】

1．掌握指针也变量地址的关系，掌握指针\*与&运算基本方法；

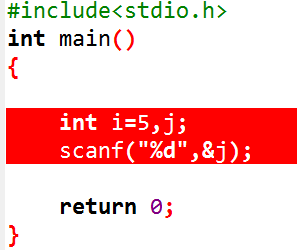
2．掌握指针和一维数组间的关系，掌握用指针变量引用一维数组元素的方法。

## 【上机内容】

### **【一般示例】**

#### 【例1】 地址与指针

计算机的主存储器被分成一个个存储单元，为了区分各存储单元，要为每个存储单元编号，这个编号即地址。

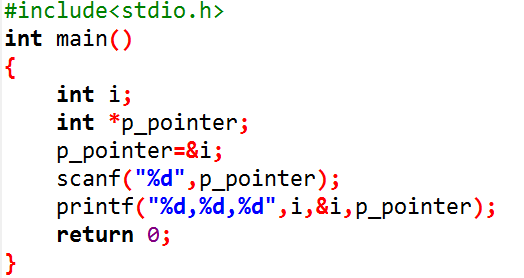


上面程序中，标红部分语句分别表示将整数3送给变量 i所在的存储空间，将键盘输入整数送给变量j所在的存储空间。

程序语言中，可使用另一个变量来指向存储单元的地址。

所谓“指向”，是通过地址来体现的。如运行下列程序，分配给变量i的地址是6487620，使用变量p\_pointer存储该地址，这样就在pointer和变量之间建立起一种联系，即通过pointer能知道i的地址，从而找到变量i的内存单元。

因而在C语言中，将地址形象化地称为“指针”。



上述程序中，&i表示变量i的存储单元地址，使用p\_pointer指向i的存储单元地址。

程序运行结果如下：



可以看出，变量内容为5，与变量地址6487620是不一样的，指针变量p\_pointer存储了变量i的地址，通过p\_pointer能找到以它为地址的内存单元。一个变量的地址称为该变量的“指针”。

内存单元的**地址**和内存单元的**内容**是两个不同的概念。

**指针**：就是地址，即内存单元的编号。

**指针变量**：用来存放另一变量的地址(即指针)的变量。

如：地址6487620是变量 i的指针；p\_pointer是指针变量，其值就是指针6487620。

#### 【例2】 指针基本操作

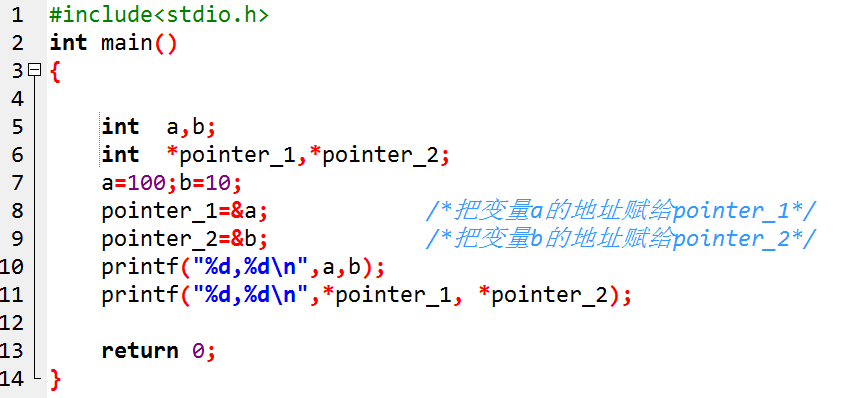
指针变量有两个运算符：

**&：取地址运算符**

功能：取变量地址；单目，右结合。

**\*：取内容运算符（“间接访问”运算符）**

功能：只能跟地址，取变量所指向单元的内容；单目，右结合。



上述程序的运行结果是



程序说明：

1、在第6行虽然定义了两个指针变量，只是提供了两个指针变量，但并未指向任何一个整型变量。称为指针“悬空”。

2、第11行\*pointer\_1和\*pointer\_2与变量名a和b等价。

3、程序中两处出现\*pointer\_1等，含义不同。程序第6行中的\*pointer\_1表示定义指针变量pointer\_1。它前面的\*只是表示该变量是指针变量。程序11行中的\*pointer\_1则代表指针变量的\*运算，即取pointer\_1所指向的变量值。

4、第8行中的pointer\_1=&a 是将a的地址赋给指针变量pointer\_1，而不是\*pointer\_1。因此不能写成：\*pointer\_1=&a;

5、从上例中可看出，\*pointer\_1等价于a，\*pointer\_2等价于b，故凡在程序中出现a的地方均可用 \*pointer\_1 代替。

为了进一步了理解&和\*运算符，建议通过上机实验，思考下列问题：

假设已执行语句pointer\_1=&a;

1、&\*pointer\_1含义是什么？

&\*pointer\_1与&a相同，即变量a的地址。

2、\*&a的含义是什么？

先进行&a运算，得a的地址，再进行\*运算。

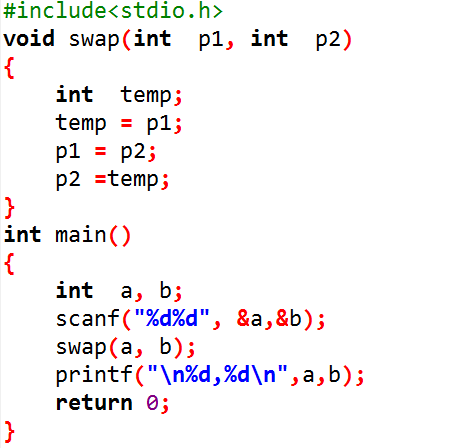
\*&a、\*pointer\_1及变量a等价。

3、(\*pointer\_1) + + 相当于a + +。

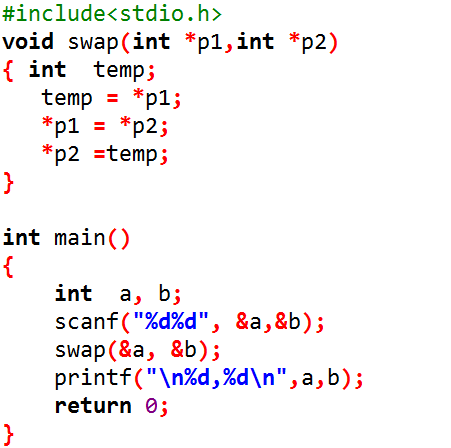
它与\*pointer\_1 + + 不同。

4、\*pointer\_1 + + 等价于\*(pointer\_1 + +)，即先进行\*运算，得到a的值，然后使pointer\_1的值改变，这样pointer\_1不再指向a了。

#### 【例3】指针变量作为函数参数



运行上面程序，能否实现主函数中变量a与b的交换。



### **【实验任务】**

完成下列至少4道以上题目

#### 任务1：指针求数组平均值

**题目描述**

主函数中随意输入十个整数，在getaver(int \*p)中，利用指针实现对这十个整数的访问，求其平均值，结果保留到小数点后两位。

**输入**

1行：

输入十个绝对值不大于100的整数。

**输出**

1行：

平均数，保留到小数点后两位。

**【示例1】**

**输入**

1 1 1 1 1 1 1 1 1 1

**输入**

1.00

#### 任务2：数字交换位置

**题目描述**

函数void max\_min\_value (int \*number,int n)实现将一个数组中最小数与第一个数对换，最大数与最后一个数对换。

**输入**

2行：

第1行，n(n<20)

第2行，输出数组中n个元素

**输入**

1行：

交换后的数组，每个元素之间用空格分开

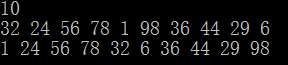
**【示例1】**

**输入**

10

32 24 56 78 1 98 36 44 29 6

**输出**



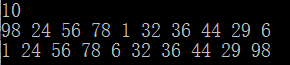
**【示例2】**

**输入**

10

98 24 56 78 1 32 36 44 29 6

**输出**



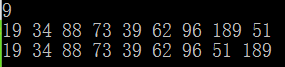
**【示例3】**

**输入**

9

19 34 88 73 39 62 96 189 51

**输出**



#### 任务3：数字向后移动

**题目描述**

有n个整数，编写函数int move\_n(int\* a,int n,int m)，使前面各数顺序向后移m个位置，即最后m个数变成最前面m个数。

**输入**

3行：

第1行，n

第2行，m

第3行，n个整数（中间以空格隔开）

**输入**

1行：

变序后的n个整数

**【示例1】**

**输入**

6

2

1 2 3 4 5 6

**输出**

5 6 1 2 3 4

**【示例2】**

**输入**

8

5

1 2 3 4 5 6 7 8

**输出**

4 5 6 7 8 1 2 3

#### 任务4：数字反序排列

**题目描述**

将数组a中n个整数按相反的顺序存放。编写函数void invert(int \*p,int n)实现以上功能。

**输入**

2行：

第 1行，输入数组a的长度n；

第2行，输入数组a。

**输出**

1行：

反序后的数组a。

**【示例1】**

**输入**

4

**输出**

4

**【示例2】**

**输入**

4

**输出**

4

#### 任务5：约瑟夫问题

**题目描述**

n个人围成一圈，顺序排号。从第1个人开始报数（从1到m）,凡报到m的人退出圈子，问最后留下的是原来第几号。用函数和指针实现以上功能，返回数组中最后留下的元素序号。

**输入**

1行：

n和m，中间使用空格分开，分别表示总人数和该出局的报数。

**输入**

输出最后留下者的序号x,输出格式为：NO.x

**【示例1】**

**输入**

8 3

**输出**

NO.7

**【示例2】**

**输入**

13

**输出**

NO.13

#### 任务6：数部局部排序

**题目描述**

有一数组a[N]，需要对其局元素进行从高到低排序，请编写函数localsort(int \*start ,int len)，实现数组指定位置开始后的len个元素从高到低排序。N<100，start+len<&a[N-1]。

**输入**

2行：

第1行,n,b,l，n表示数组元素个数，n<=N,b表示开始排序的下标，1<b<=n,l表示从开始下标起排序元素长度,1<=b+l<=n；

第2行,n个数组元素。

**输出**

1行，局部排序后的数组

**【示例1】**

**输入**

10 2 5

1 2 3 4 5 6 7 8 9 10

**输出**

1 6 5 4 3 2 7 8 9 10

**输入**

5 1 5

6 8 7 4 2

**输出**

8 7 6 4 2