实验八、指针

班级：机设（3）班 学号：115040100304 姓名：章文瑾

1. 目的要求
2. 掌握地址与指针、指针变量的概念。
3. 掌握指针的定义、赋值、使用和传递。
4. 学会正确区分和使用变量的指针和指针变量。
5. 掌握数组指针和指针数组的使用方法。
6. 了解指向函数的指针。
7. 实验步骤与内容

指针变量的定义、赋值、操作（储存单元的引用，移动指针的操作，指针的比较），取地址运算符（&）和间接运算符（\*）的功能，数组元素的多种表示形式。

1. 实验小结
2. 指针的定义与之前实验中变量的定义很相似，只不过在变量名前多了一个“\*”号。如果定义一个指向整型变量的指针变量，可写成：int \*a;。如果想定义一个指向字符型变量的指针变量，写成：char \*a;。指针变量也是一个变量，它也相当于一个储存空间，只不过这个空间内只能存放地址，由这个地址去指向一个值，而不是直接存储一个值。例如想提取一个变量c的地址则要使用“&”符号。可以写成“a=&c”。这样c的地址就存储到a里面去了。则现在修改c的值，就有两种方法了。一是直接修改，例如：c=1。二是通过指针修改：\*a=1。实际上如果a是c的指针，那么“c”和“\*a”是等价的。这里的“\*a”是运算a指向到的地址内容的意思，与定义之中的“\*a”要注意区分。
3. 我们在定义数组的时候，实际上也是定义了一个首地址，也就是第一个元素的地址，然后系统根据其存储数据的类型以及个数去分配所需要的存储空间。那么这里的数组名也就是相当于一个地址。前面我们知道指针也存储着一个地址，这一点二者很相似。唯一不同的是，数组名是常量指针，它是不可更改的。而指针变量却是可以更改的。例如：int a[];int \*p;p=a;。因为a和p都是地址，所以可以直接相等，无需加取地址符号（\*）。这样我们再一次去表示这个数组的时候，方法就是多种多样了。当然我们可以写成最熟悉的样式：a[i]。除此之外还有：\*(a+i),\*(p+i),p[i],\*p++。这里我们特别注意下最后一个“\*p++”这是指针变量表示数组特有的方法，我们不能写“\*a++”，因为a是常量，它是在你定义数组名的时候就已经确定好了的，它是不可更改的。这里还有一点需要额外注意，那就是几种表示方法各有利弊。“a[i]”比较直观明了，但“\*(a+i)”效率比较高。因为我们在用“a[i]”的时候，系统首先要计算a+i\*sizeof（数组类型），再通过此地址去访问对应的内存空间。而“\*（a+i）”则能根据其地址直接访问，提高了效率。所以我们在表示数组元素的时候，常用“\*(a+i)”方法。
4. 二维数组与指针的关系与前面的一维有很多类似的地方。只不过，这里的“\*(a+i)”不再代表第i个元素，而是代表第i行的第0列，是列指针。这里不能认为“\*(a+i)”是“a+i”指向的对象，因为在二维数组中“a+i”是指向行的，而不是指向具体的元素。所以，我们想表示一个二维数组中某个具体的元素，我们可以使用“a[i][k],\*(a[i]+k),\*(\*(a+i)+k)”。另外，如果：int \*p;，再要想用p来表示数组a，我们可以：p=a[0];，这里与一维数组写法有些不同，但从其含义上来讲确实一样的。p要等于二维数组的首地址，但二维数组的首地址是“a[0]”。做了上面的定义之后，我们任可以使用“\*p++”来表示数组。而且不用管第几行第几列，因为不论是一维还是二维数组，在内部空间储存的时候，都是以线性来存储的，之所以不同，是我们定义的提取方式不同而已。所以这里你的“\*p++”可以提取到该二维数组里的所有元素，只是在最终输出时需要注意一行的元素满足列数之后要换行。写法上可以参照如下：

For(j=0;j<行数;j++)

{

For(i=0;i<列数;i++)

Cout<<setw(理想字符间距)<<\*p++;

Cout<<endl;

}

便可以得到该二维数组，并且以我们习惯的方式输出了。

1. 字符型数组与指针的关系与前面提到的一位整型是相同的。如果：char a[]=’china’;char \*p;p=&a;，那么我们想输出这个字符串，可以写：“cout<<\*p;”，若要指定第i个元素，则写：cout<<\*(p+i);即可。
2. 数组的输入也可以用指针来实现，例如：int a[6],i;int \*p;for(i=0;i<6;i++){cin>>\*(p+i);，也是可以实现输入的。
3. 上机题6需要实现将两个字符串对应位置字符中的较大者存放到数组c之中，这题需要用到一个字符串连接函数“strcat(a)”。需要考虑的情况是两个字符串的长度不一样，这样就需要把多的那部分连接到比较的部分中去。写法如下：“strcat(c,p)”这是p比q长的情况。为什么这里只写一个p就可以把剩下的所有字符串连接上去了呢？这是因为我们比较到“\0”前一位的时候，p的地址也已经指向这一位了，所以这时候如果单纯的写“cout<<p;”那么就会把剩下的部分输出来了。也就是说，p现在代表着剩下没有比较的部分，因为此时的p与之前的p不一样了，因为我们比较的时候已经把p的地址逐步增加到了后面的位数了，所以这时候如果以字符型数组的眼光去看p，它已经是剩下的字符串数组了。但这样的方法对整型数组却是不适用的。
4. 运用指针还可以实现通过改变形参变量来修改实参变量的目的。如果我们写“int \*t;t=p1;p1=p2;p2=t;”这里我们只是交换了地址，但是地址上内容我们却没有改变，相当于它的本身并没有变，但是你找它的方式改变了。所以此时实参的量是不变的，输出a、b的值仍是不变。但是若写：“int t;t=\*p1;\*p1=\*p2;\*p2=t;”这就是直接把地址上的内容改变了，此时如果输出a、b的值，是和原来不一样的。这里便体现出了指针的一个小优势，因为在程序之中，无论是赋值也好，函数也罢。归根到底还是基于对每一个地址上存储内容的处理，只不过有些处理的步骤简单，例如就是原样输出该地址上的内容。而有的处理复杂，例如，提取这个地址的内容，又要去提取另一个地址的内容，然后把它们相加，再存放进某一个地址。可不论多么繁杂，仍然是跳脱不出地址存储单元的这个框架的。所以，我们使用指针，也就是从这个框架最基础的部分入手，形式上是简单的，但运用却是灵活高效的。
5. 在使用指针的时候，我们要注意，就是“int \*t；t=\*p；”这是错误的，因为你定义了t是一个存放整型地址的变量，但你又让它与一个数值相等，是不可以的，它只能与地址相等。你可以写“t=p”,或者“\*t=\*p”。