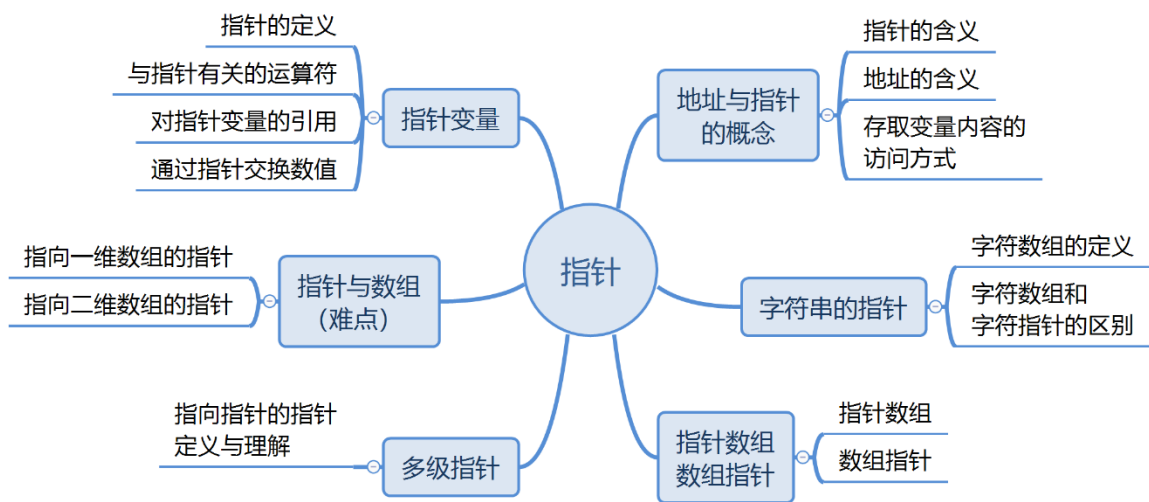




专题六 指针

【内容预览】



【知识清单】

6.1、地址与指针的概念

6.1.1、指针的含义

- 指针就是用来存放地址（变量占据的内存单元的首地址）的变量。
1. 这个地址可以是变量的地址，也可以是数组、函数的起始地址，还可以是指针的地址。某个指针存放了哪个变量的地址，就说该指针指向了这个变量。
 2. 需要注意的是指针变量的值是它所指向的变量的地址，而不是该变量的值。

6.1.2、地址的含义

计算机中 CPU 可以直接访问的、用来存储数据的记忆部件称为存储器。存储器由成千上万个顺序存储单元组成，每个单元有一个唯一的地址标识。给定计算机的存储器地址范围为从 0 到所安装的存储器数量的最大值减 1。在计算机上所运行的每一个程序都要使用存储器。

在程序中定义一个变量，在编译时就给这个变量分配内存单元，系统根据程序中定义的变量类型，分配一定长度的空间。例如，一个 int 型变量占据的存储单元为 2 个或 4 个字节，一个 float 型为 4 个字节，一个 char 型为一个字节等。变量所占据的内存单元的首地址就是变量的地址。

6.2、指针变量

6.2.1、指针的定义

定义格式：

类型名 *指针名

1. *号是一个指针变量的标志；**指针可与非指针变量一起说明**。如：char *p,ch;
2. **在使用指针之前，一定要先使指针具有明显的指向**，这一点可以通过对指针初始化或者赋值来完成。

如下：int i,j;

```
int *p1,*p2; //定义指针
```

```
p1=&i;
```

```
p2=&j; //赋值
```

与常规的变量未赋初值相同，没有明确指向的指针不会引起编译器出错，**但是对于指针来说，可能导致无法预料的或者隐藏的灾难性后果**。

3. **只有整型变量的地址才能放到指向整型变量的指针变量中去**，因为不同数据类型的指针指向下一位需要移动不同的字节数，如字符型需要移动一个字节，而整型需要移动二个或四个字节.....

如：float a; //定义 a 为 float 型

```
int *pointer_1; //定义 pointer_1 为类型为 int 的指针变量
```

```
pointer_1=&a; //将 float 型变量的地址放到指向 int 型指针变量中，错误
```

4. 指针变量中只能存放地址，**不能将一个常量（或任何其它非地址类型的数据）赋给一个指针变量**。

如：int *pointer_1=100; //错误，pointer_1 为指针变量，100 为常量。

6.2.2、与指针有关的运算符

&：取地址运算符

*：指针运算符（或称“间接访问”运算符）。取其指向的内容

1. &运算符**只能作用于变量**，包括基本类型和数组的元素，**不能作用于数组名和常量**。

如：int a[20],n;

表达式&n,&a[0]是合法的,&a 是非法的。

2. 单目运算符*是&的逆运算，它的操作对象是地址，*运算的结果是指针指向的对象本身。

如：char c;

```
char *pc;
```

```
pc=&c; //那么*pc 的结果就是 c 的值
```

3. “&”和“*”两个运算符的优先级相同，但是按**自右向左的方向结合**，如&*p 因此先进行*p 的计算，再执行&运算。
4. **(*p)++、*p++和*(++p)的辨别**：(*p)++是指*p 所指向的值加一，*p++相当于*(p++)，先得到*p，然后再使 p 的值加一；*(++p)指的是 p 先加一，然后再取*p。

6.2.3、对指针变量的引用

1. 给指针变量赋值。如：

```
int a,*p;
```

```
p=&a; //把 a 的地址赋给指针变量 p
```

2. 引用指针变量指向的变量

(1) 如果已执行“p=&a;”，即指针变量 p 指向了整型变量 a，

则 printf("%d",*p);

作用是以整数形式输出指针变量 p 所指向的变量的值，即 a 的值。

(2) 如果有以下的**赋值语句**

```
int a,*p;
p=&a;
*p=1;
```

表示将整数 1 赋给 p 当前所指向的变量，如果 p 指向变量 a，则相当于把 1 赋给 a，即“a=1;”。

(3) 引用指针变量的值。如：

```
printf("%o",p);
```

作用是以八进制数形式输出指针变量 p 的值，如果 p 指向了 a，就是输出了 a 的地址，即&a。

3. 下面举一个例子分析一下：

```
#include<stdio.h>
void main()
{
    int a;
    int *pa=&a;
    a=10;
    printf("a:%d\n",a);
    printf("*pa:%d\n",*pa);
    printf("&a:%x(HEX)\n",&a);
    printf("pa:%x(HEX)\n",pa);
    printf("&pa:%x(HEX)\n",&pa);
}
```

运行的结果为：

```
a:10
*pa:10
&a:fff4(HEX)
pa:fff4(HEX)
&pa:fff2(HEX)
```

上述输出的结果中，第一和第二行输出的结果是值，不会改变，而后三行输出的是地址值，每次运行可能输出的地址值不同。

6.2.4、通过指针交换数值

```
int a,b,*m,*n,p,*q;
m=&a;
n=&b;
```

值交换

```
p=*m;
*m=*n;
*n=p;
```

6.3、指针与数组

6.3.1、指向一维数组的指针

```
int a[10];
int *p;
p=&a[0];或 p=a; //将数组 a 的首地址赋给指针变量 p
```

表 6-1：访问数组元素的的三种方式对比（三种方法都是等价的）

下标法	数组名	指针变量
a[0]/p[0]	*a	*p
a[1]/p[1]	*(a+1)	*(p+1)
a[2]/p[2]	*(a+2)	*(p+2)
.....

6.3.2、指向二维数组的指针

```
int a[10][10];
```

表 6-2：与数组 a 相关的地址表示形式

表示形式	含义
&a	指向二维数组的指针
a	二维数组名，数组首地址
a[0],*(a+0),*a	第 1 行第 1 列元素地址
a+1,&a[1]	第 2 行首地址
a[1],*(a+1)	第 2 行第 1 列元素地址
a[1]+1,*(a+1)+1,&a[1][1]	第 2 行第 2 列元素地址
(a[1]+1),(*(a+1)+1),a[1][1]	第 2 行第 2 列元素的值

说明：

1. a[0]、a[1]、a[2]是一维数组名，而在 C 语言中数组名代表数组的首地址，因此而 a[0]代表第 1 行一维数组中第 1 列元素的地址。即&a[0][0]。a[1]的值就是&a[1][0]；a[2]的值就是&a[2][0]。
2. 既然 a[0]是第 1 行的首地址（[一个一维数组的首地址](#)），那么 a[0]+1 就是第 1 行第 2 列的元素的地址（[这个一维数组的第 2 个元素的地址](#)）了。
3. a[i]和*(a+i)等价，则 a[i]+1 和*（a+i）+1 的值都是&a[0][1]。
4. [指向行的指针和指向列的指针的区别](#)
如：`char a[2][10]={"hello","xuejie"}`
 - （1）二维数组名（如 a）是指向行的，因此“a+1”代表“xuejie”这一行。
 - （2）一维数组名（如 a[0]、a[1]）是指向列元素的。“a[0]+1”指向的是“hello”中的字符‘e’。
 - （3）在指向行的指针前面加上一个“*”，就转换成了一个指向列的指针。如 a 和 a+1 都是指向行的指针，前面加上“*”就变成了*a 和*(a+1)，它们是指向列的指针。相反如果在列指针前面加上一个“&”，则它们就成了指向行的指针。

6.4、字符串的指针

6.4.1、字符数组的定义

```
char *str="hello world!";或 char *str; str="hello world!";
```

在指向字符串的时候，并不是把字符串的所有字符存放到 `str` 中，也不是把字符串赋给 `*str`，只是把字符串的首地址赋给指针变量。

6.4.2、字符数组和字符指针的定义

```
char string[]="hello world! ";
```

```
char *str="hello world! ";
```

1. `string` 和 `str` 的值都是字符串 "hello world!" 的首地址，但是 `string` 是一个字符数组，名字本身是一个地址常量，而 `str` 是一个指向字符串首地址的字符指针，因而 `str` 可以被赋值，而 `string` 不能。
2. 通过字符数组名或者指针变量可以输出一个字符串，而对一个数值型数组，是不能试图用数组名输出它的全部元素的。如：

```
char string[]="hello world!"; *p=string;      int a[]={1,2},*p=a;
printf("%s\n",string);//正确                  printf("%s\n",a);//错误
```

6.5、指针数组与数组指针

6.5.1、指针数组

如果一个数组的元素是由指针变量组成，那么这个数组称为指针数组。

定义形式：类型名 *数组名[常量表达式] 如：`int *a[10];`

优点：① 指针数组对处理字符串提供了更大的方便和灵活，由于每个元素都为指针变量，因此通过地址运算来操作正文行是比较方便的。

② 用数组来存储这些长度不等的字符串，那么必须设定数组的长度可以容纳下字符串中最长的字符串，这样也许会浪费掉很多的内存空间。

③ 如果要对字符串进行排序，不必改动字符串的位置，只需改动指针数组中各元素的指向。

6.5.2、数组指针

是一个指向长度为整型常量的一维数组的指针。

定义形式：类型名 (*指针名)[整型常量] 如：`int (*a)[10];`

```
int a[3][4]={ {1,3,5,7},{9,11,13,15},{17,19,21,23}};
```

```
int (*p)[4];
```

`p=a+1;` // `p` 指向数组的第二行，则 `*p` 或 `p[0]` 是 `p[1][0]` 的地址。

6.6、多级指针

6.6.1、指向指针的指针定义与理解

指向指针数据的指针变量，简称为指向指针的指针。 例：`int **p;`

理解：指针变量也是变量，和其他类型的变量一致，也需要内存单元，既然占据内存单元，就有相应的地址，那么，可以再定义另外一种“指针”指向这个地址。这种指针就是“指向指针的指针”。

从理论上说，间接方法可以延伸到更多的级，但是实际上在程序中很少有超过二级间址的，级数越多，越容易产生混乱，出错的机会也多。建议初学者在编程时要慎用多级指针，即指向指针的指针。

【解题技巧】

例 6.1 下面的程序段, 运行后的输出结果是: ()

```
char s[]="baodian",*ps=s;  
printf("%c\n",*ps+2);
```

A) b

B) c

C) d

D) 元素 s[2]的地址

正解: C

分析: ‘*’ 的优先级高于 ‘+’, 先运算 *ps 得到 ‘b’, 然后加二, 得到 ‘d’。

例 6.2 int arr[10],*p=arr; 以下能够表示 arr[3] 是: ()

A) (*p)[3]

B) *p[3]

C) *(arr+3)

D) p+3

正解: C

分析: arr[3] 的正确表示有: arr[3], *(arr+3), *(p+3), p[3].

例 6.3 在 32 位的系统中, int 变量占 4 个字节, 定义 int x; int *pi=&x; 如果 printf("%x", pi); 的输出是 12FF80, 那么 printf("%x", pi+1); 的输出为: ()。

A) 12FF81

B) 12FF82

C) 12FF84

D) 12FF88

正解: C

分析: pi+1 指的是指向下一个变量, 而每个 int 型变量占 4 个字节, 则 pi+1 的值为 pi 的地址值加 4。

例 6.4 有以下程序:

```
int aaa(char *s)  
{  
    char *t=s;  
    while(*t++);  
    t--;  
    return(t-s);  
}
```

以下关于 aaa 函数功能叙述正确的是

A) 将串 s 赋值到串 t

B) 比较两个串的大小

C) 求字符串 s 的长度

D) 求字符串 s 所占字节数

正解: C

分析: 函数 aaa 的返回值是 int 型, 排除 A; 再由 while 循环体内容可知 t 最后指向字符串结束标识符处, 而 s 指向字符串首地址, 因此返回值是字符串的长度。

【精选习题】

答案 P120

1. 设已定义: `float x;` 则以下对指针变量 `p` 进行定义且赋初值的语句中正确的是 ()
A) `int *p=(float)x;`
B) `float *p=&x;`
C) `float p=&x;`
D) `float *p=1024;`
2. 设已正确定义字符串 `s[]={"hello!xuexiyun"}`, `char *p=s;` 下列能够正确输出整个字符串的选项是 ()
A) `printf("%s",*s);`
B) `printf("%s",*p);`
C) `printf("%s",&s);`
D) `printf("%s",p);`
3. 设已正确定义字符串 `s[]={"HELLO!xuexiyun"}`, `char *p=s;` 下列选项中能够正确输出字符 'E' 的是 ()
A) `printf("%c",&s[1]);`
B) `printf("%c",&p[1]);`
C) `printf("%c",++p);`
D) `printf("%c",p[1]);`
4. 设已正确定义字符串 `s[]={"HELLO!xuexiyun"}`, `char *p=s;` 下列选项中与 `&s[1]` 等价的是 ()
A) `s++`
B) `&p++`
C) `p[1]`
D) `s+1`
5. 设已正确定义字符串 `s[]={"HELLO!xuexiyun"}`, `char *p=s;` 执行 `strcpy(s,p+=6);` 后, 输出的结果为 ()
A) `HELLO!xuexiyun`
B) `!xuexiyun`
C) `xuexiyun`
D) `uexiyun`
6. 设已正确定义字符串 `s[]={"HELLO!xuexiyun"}`, `char *p=s;` 若此时 `p` 的值为 `0x2FFF FFFA`, 当 `p` 指向字符 'O' 时, `p` 的值为 ()
A) `0x2FFF FFFC`
B) `0x2FFF FFDD`
C) `0x2FFF FFEE`
D) `0x2FFF FFFF`
7. 设有定义语句: `int a,*p=&a,**pp=&p;` 则与 `a=100;` 等价的 C 语句是 ()
A) `**p=100`
B) `**pp=100`
C) `&*p=100`
D) `*pp=10`
8. 设 `char **s;` 以下正确的表达式是 ()
A) `s="good";`
B) `*s="good";`
C) `**s="good";`
D) `*s='g';`
9. 设 `x`、`y`、`temp` 均为 `int` 型变量, `p1` 和 `p2` 为整型指针变量, 则以下语句: `p1=&x; p2=&y; temp=*p1; *p1=*p2; *p2=temp;` 的功能是_____。

- A) 把 x 和 y 按从大到小排列
C) 无确定结果

- B) 把 x 和 y 按从小到大排列
D) 交换 x 和 y 中的值

10. 下列程序输出是_____。

```
#include <stdio.h>
void main()
{
    char *s="ab5ca2cd34ef",*p;
    int i, j, a[]={0,0,0,0,0};
    for(p=s;*p!='\0';p++)
    {
        j=*p-'a';
        if(j>=0&& j<5) a[j]++;
    }
    for(i=0;i<5;i++)
        printf("%d ",a[i]);
    printf("\n");
}
```

11.

```
#include<stdio.h>
void main()
{
    int a[]={1,2,3},b[]={4,2,5};
    int *p,*q;
    int i,x;
    p=a;q=b;
    for(i=0;i<3;i++)
        if(*(p+i)==*(q+i))
            x=*(p+i)*2;
    printf("%d",x);
}
```

程序运行的结果为_____

12.

```
#include<stdio.h>
void main()
{
    int a[]={1,2,3,-1},b[]={4,5,6,7,-1};
    int *p[]={a,b},**pp=p,i;
    for(i=0;i<2; i++)
    {
        while(**pp>=0)
        {
            printf("%3d",**pp);
            (*pp)++;
        }
        printf("\n");
        pp++;
    }
}
```

程序运行的结果为_____