第十章 指针回顾总结

指针需要回顾得有很多, 我将我所知道的重要的指针的知识点做一个总结：

指针就像是C语言和C++的核武器, 用好指针的话, 它将会为整个程序带来极大的方便, 如果用不好的话, 就是毁灭一切程序的核武器, 很有杀伤力。

指针就是一个存地址的变量, 它和其他的变量没有什么区别, 指针存的只是地址, 而不是其他的普通数据, 就像是int型只能存整数一样, 在使用时必须先定义, 例如, int\* p, float\* p, double\* p, p就被定义为指针变量, 那么定义完的p就简称为的指针;

定义后自然会将地址赋值给指针, 如果你想存的是一个int型, float型, double型的普通变量x, 就可以如下：p=&x;

如果是数组的话就是在定义一个数组之后 int a[N], 直接将数组名放入指针中：p=a, 因为数组名代表数组的首地址与数组的第1个元素的地址相同：a=&a[0];

4.地址就是一个数据, 就是个地址常量, 地址就像一个门牌号, 给变量每个存储单元标记上编号, 而这个地址是一个8位16进制数;

地址又好像是一个路标, 指向存储空间, 利用它可以访问存储空间, 从而获得存储空间的内容;

5.系统会根据变量的类型为该变量分配相应字节的内存空间：系统最小开辟4字节存储空间, 虽然字符型最小占1个字节空间, 但还是要多分配3个字节的空间, 分配即为开辟字节;

6.“&”是取地址运算符, 单目运算符, 右结合;&变量名, 获得变量开辟的空间的地址

7.变量的地址输出时要使用格式修饰符“%p”;

8.指针的指向就是指针变量存放的变量的地址, 我们叫这个指针指向了这个变量, 如果这个指针赋值为一个数组名, 那么就叫做这个指针指向了这个数组;

9.指针变量和地址不是等价的, 地址可以用指针来替代, 但指针终归还是一个变量, 地址还是一个地址常量;

10.我们在定义一个指针时应该是：类型说明符\* 变量名;这里的变量名依然必须符合标示符的命名规则;//这只是定义一个什么类型的指针变量。

！！注意：一个指针只能指向同类型的变量, 什么型变量就能指什么型。

11.指针变量同普通变量一样, 使用之前不仅要定义说明, 而且还要赋予具体的值。

未经赋值的指针变量不能使用, 否则会造成系统混乱, 甚至停机;

12.指针赋予谁的地址就是指针指向谁。

赋值时应该是：例：int a=3;

int\* p;

p=&a;//这就说明地址指向a这个变量;

13.指针变量的引用：使用格式：\*指针变量名;在这里\*为指向运算符, 是单目运算符, 右结合;

作用：求运算符后面的指针变量所指向的变量的值;也就是指针变量所指向的存储空间的内容。

14.scanf("%d%d", &a, &b);与scanf("%d%d", p, q);等价。

15.指针数组, 指针指向数组, 应该把数组名赋值给指针, 数组的地址就可以由数组指针替换, 但始终要注意指针只是一个变量, 它俩不是完全等价的, 而数组名是数组首地址, 这也就是说数组名是地址, 既然是地址了, 那么这个地址就是地址常量, 虽然可以替换也只是替换成了同类型地址, 而他们还是有本质上的区别

例如：p是一个指针, a[N]是一个数组,

p是可以p++, 但当p指针指向数组a时, a就可以a++吗？, 答案是当然不可以

16.数组在内存中占据的存储空间是连续的空间, 说到内存空间, 栈区, 我们在定义变量时, 系统分配的存储空间的地址大小是按：越先定义的变量所在的十六进制数地址越大, 越后声明的变量存储空间地址越小。

17.当然, 定义一个指针变量指向数组与定义一个指针指向一个普通变量差不太多, 只是在赋值时有一些区别, 但问题不太大, 举个例子：

指向普通变量:int x;

int\* p;

p=&x;//这一步就是将指针指向变量x;

将x的地址赋值给指针变量p, 就可以说指针变量p指向了变量x

指向数组：int a[N]; //先定义一个指针变量

int\* p;

法一： p=a;//数组名是地址所以将数组名就是地址, 将数组名赋值给指针就是使指针p指向数组a。

法二： p=&a[0];//数组的第一个元素就是数组的首地址所以将首地址赋值给指针就是指针指向数组, 也就可以用指针来访问数组。

法一实质上与法二是等价的。

18.一维数组表示法：(五种)

先定义一个数组, 再定义一个指针;

int arrint[N];//定义数组

int i;

int\* piont;//定义指针

piont =arrint;//将指针指向数组

1.for (i=0; i<N; i++)

{

print("%d", arrint[i]);

}

printf("\n");

2.for(i=0;i<N;i++){

printf("%d", \*(arrint+i));

}

printf("\n");

3.for(i=0;i<N;i++){

printf("%d", piont [i]);

}

printf("\n");

4.for(i=0;i<N;i++){

printf("%d", \*( piont +i));

}

printf("\n");

5.for(i=0;i<N;i++){//这种方法的特殊, 就在于指针是一个变量, 变量就可以有变化, 可以自增自减.

printf("%d", \* piont);

piont ++; //这涉及到指针偏移, 指针偏移量就是指针指向变量类型的单个字节数.

！arrint++//！是绝对不行的！！！！！, 因为数组名是地址常量, 永远不能出现在运算符的左侧。

}

printf("\n");

特别注意：以上的程序中的第五种方法, 最好在编写程序的最后使用, 否则就会更改原指针指向。一发不可收拾.

19.指针变量的运算(指针的移动)：移动的方式有以下几种：

p+n, p-n, p++, p--, ++p, --p;

进行加法运算时表示p向地址增大的方向移动, 进行减法运算时p向地址减小的方向移动。移动的具体长度取决于指针指向的数据类型;

p是啥子类型,就偏移这种类型的字节数：

p++, 如果p是整型的, p++就是偏移这个整型的字节数4字节;

p+n, 如果p是整型的, 就是p的地址偏移这个型的n倍个字节;

当一个指针指向一个数组的话, 就是：

p+2代表p指向了数组a的第三个元素a[2];

20.指针可以偏移！再说一遍很！重要！的话, 以后还要说：地址是地址常量, 指针是地址变量, 千万要记住！！.

21.时刻要牢记啊！：a[n]是数组中的第n+1个元素, 因为数组第一个元素是从a[0]开始的;

&a[0]是数组的首地址也就是数组还没有偏移时的地址！

22.同类型的指针变量之间的运算：

(1)指向同一数组的两个指针变量之间可以运算：只能做减法运算。

两个指针(指针变量)相减之差即是两个指针之间的相对距离（距离就是相差数据元素个数）通俗点来说就是两个地址值相减之差再除以该元素的长度（字节数）。

值得注意的是：其实也可以不去注意, 指针变量不能做加法运算, 因为做加法运算毫无实际意义。

(2)两个指针变量的关系运算（提一下关系运算：<, >, <=, >=, ==, !=）

指的是同类型地址之间的关系运算。举个例子参考一下：

#include<stdio.h>

#define N 10

int main()

{

int a[N]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int\* p;

sum=0;

for(p=a; p-a<N; p++){ //括号里的p-a<N, 就做了指针变量的关系运算, p-a的值是当前循环访问到的数组元素下标

sum+=\*p;

}

printf("%d\n", sum);

return 0;

}

p==0表示p是空指针, 他不指向任何变量。

23.指针指向字符串：

首先来说字符串, 字符串是存储在字符数组中的, 字符数组的存储是在栈区中的, 而字符串存储是在全局区的。字符串是全局区的字符常量。当指针变量指向字符串时, 就会取到全局区的地址。

如果单独取字符串的地址的话, 那么取到的地址是全局区的地址, , 如果要是将字符串存在字符数组里的话, 取其地址就是在栈区取到的。

24.字符串型指针变量与字符数组的区别：

对字符串型指针方式：char\* p="hello c";

对字符数组的字符串的方式为char a[]={"hello c"}; 麻烦！不要这样指向。

当一个指针变量未取得确定的地址前使用是危险的！

字符串指针在接收字符串时, 必须先开辟存储空间。

对于字符数组str[20]可以用语句scanf("%s", str);进行输入;

而对于字符串指针p则不能直接用它输入, 必须先给p指向的变量开辟存储空间后,才可以进行输入。

指针变量在定义后,被开辟了一个p指针变量它自己的存储空间, 这个空间无论是什么型都是开辟4个字节, 因为指针变量只是一个可以存8位16进制数地址的变量, 不管是什么型它们的第之都是8位16进制数;

指针是一个变量, 既然是变量就会有一个空间, 这个空间是4个字节, 这说明指针也是有地址的, 我们对指针存地址是把所要指向的变量(各种类型)的地址给他, 这指针就可以代替变量来对变量中的内容进行访问, 或对其进行修改

这样的话, 指针既可以做为一个变量, 又可以做为一个变量的指针(地址)来使用, 这样的指针就是一个非常强大的工具, 使程序运行更加方便, 更加高效

char a[N];

char\* p;

p=a;

scanf("%s", p);//将存放的内容 存入地址指向的变量 占有的空间中

scanf("%s", &p);//将存储的内容存入指针变量本身所占有的存储空间

printf("%s", a);