2020/04/22_第16课_指针数组、数组指针、指针的各种运算(指针3)

笔记本:	C
创建时间:	2020/4/22 星期三 15:36
作者:	ileemi
标签:	数组指针, 指针的各种运算, 指针数组

指针数组

是一个特殊的数组，每个元素都是指针类型，元素是指针

指针数组（变成字符串数组）varchar
结合指针作为数组的元素，既可以存储更紧凑，又可以带来访问的优势
特点：访问效率比较低，存储空间比较紧凑
将每个字符串的地址保存到数组内

```
    //指针数组（变成字符串数组）
    char *ary[] = {
        "Hello",
        "World",
        "I Like C\r\n"  //字符串拼接，不需要添加逗号，用于长字符串，便于UI体现
        " and C++\r\n"
        " and python"
    };
```

缺陷：间接访问

定长字符串数组
优点，访问数组内第n个元素比较快

```
//定长字符串数组
    char szBuf[][32] = {
        "Hello",
        "World",
        "I Like C\r\n"  //字符串拼接，不需要添加逗号，用于长字符串，便于UI体现
        " and C++\r\n"
```

```
        " and python"
    };
```

指针数组的对其元素的访问方式:

[数组名]是[数组第0个元素]类型的指针常量
[二维数组]的元素是[一维数组]
[对某类型的指针]做 * (取内容) 运算，得到[某类型]的标识符引用

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    //定长字符串数组
    char szBuf[][32] = {
        "Hello",
        "World",
        "I Like C\r\n"  //字符串拼接，不需要添加逗号，用于长字符串，便于UI体现
        " and C++\r\n"
        " and python"
    };


    //指针数组（变成字符串数组）
    char *ary[] = {
        "Hello",
        "World",
        "I Like C\r\n"  //字符串拼接，不需要添加逗号，用于长字符串，便于UI体现
        " and C++\r\n"
        " and python"
    };

    char *psz = ary[1];

    printf("%c\r\n", *psz);
    printf("%c\r\n", *(ary[1]));      //针对数组尽量用下标，针对指针尽量用*(取内容)的方式
    //printf("%c\r\n", **ary);

    printf("%c\r\n", szBuf[0][0]);
    /*
    [数组名]是[数组第0个元素]类型的指针常量
    [二维数组]的元素是[一维数组]
    [对某类型的指针]做*（取内容）运算，得到[某类型]的标识符引用
```

```
    szBuf的元素是char [32]
    szBuf是char[32]类型的指针常量
    *szBuf, 得到char[32]的标识符引用
    也就是说，*szBuf得到了一维数组的标识符引用
    *szBuf是char类型的指针常量
    **szBuf,就得到char的标识符引用

    */
    printf("%c\r\n", **szBuf);       // 两次间接访问

    system("pause");
    return 0;
}
```

数组名是数组第0个元素类型的指针常量
二维数组的元素是一位数组
对某类型的指针做 * 运算，得到某类型的引用

指针运算：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[])
{
    int nAry[2][4] = {
        {0x10, 0x20, 0x30, 0x40},
        {0x50, 0x60, 0x70, 0x80}
    };

    /*
    [数组名]是[数组第0个元素]类型的指针常量
    [二维数组]的元素是[一维数组]
    [对某类型的指针]做*（取内容）运算，得到[某类型]的标识符引用

    */

    //nAry是int[4]类型的指针常量
    printf("%p\r\n", nAry);          //0012ff60

    //nAry的元素是int[4]
    //nAry[0]得到int[4]的标识符引用
    //nAry[0]是int[4]类型的指针常量
    printf("%p\r\n", nAry[0]);       //0012ff60

    //nAry[0][0]，得到int的标识符引用
    printf("%p\r\n", nAry[0][0]);    //00000010
```

```
    // type *p = ...
    // int n = ...
    // p + n 得到type *类型的指针常量
    // p + n == (type * const)((int)p + sizeof(type)*n)

    // nAry是 int[4] 类型的指针常量
    // (int)nAry + sizeof(int[4])*1
    // 0012ff60 + 16(0x10) == 0012ff70
    printf("%p\r\n", nAry + 1);      //0012ff70

    // *nAry, 得到 int[4] 的标识符引用
    // int[4] 是个一维数组
    // *nAry 是 int 类型的指针常量
    // *nAry + 1 == (int)nAry + sizeof(int)*1
    //  0012ff60 + 4 = 64
    printf("%p\r\n", *nAry + 1);     //0012ff64

    // *nAry 是 int 类型的指针常量
    // **nAry, 得到 int 类型 的标识符引用
    // **nAry + 1 == *(*(nAry)) + 1 == *(0012ff60) + 1 == 00000011
    printf("%p\r\n", **nAry + 1);    //00000010

        system("pause");
    return 0;
}
```

数组指针

指向数组的指针

```
    int nAry[2][4] = {
        {0x10, 0x20, 0x30, 0x40},
        {0x50, 0x60, 0x70, 0x80}
    };
    //nAry是int[4]类型的指针常量
    int (*pAry)[4] = nAry; //pAry是一个指向数组的指针
```

对谁取地址就得到谁的地址
对某类型取地址就得到某类型的地址
对整型数组取地址得到整型数组的地址（不等于数组元素的地址）

```
    int nAry[2] = {0};
    int *pAry = &pAry;    //语法错误，编译器不通过，cannot convert from
'int (*)[2]' to 'int *'

    //对整型数组取地址得到整型数组的地址
    //应该这样写
    int ary[2] = {0};
    int (*pAry)[2] = &nAry;
```

数组指针的指针的各种运算：

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
        int nAry[2][4] = {
        {0x10, 0x20, 0x30, 0x40},
        {0x50, 0x60, 0x70, 0x80}
    };

    /*
    [数组名]是[数组第0个元素]类型的指针常量
    [二维数组]的元素是[一维数组]
    [对某类型的指针]做*（取内容）运算，得到[某类型]的标识符引用
    */
    //nAry是int[4]类型的指针常量
    int (*ptr)[4] = nAry;

    //指针ptr指向二维数组nAry的首地址
    printf("%p\r\n", ptr);        //0012ff60

    /*
    *ptr，得到 int[4] 的标识符引用
    int[4]是一个一维数组
    *ptr是 int 类型的指针常量
    */
    printf("%p\r\n", *ptr);       //0012ff60

    //**ptr是int类型，对*(ptr)去内容，*(*(ptr)) == *(0012ff60) ==
00000010
    printf("%p\r\n", **ptr);      //00000010


    // (int)ptr + sizeof(int[4]) * 1
```

```
    printf("%p\r\n", ptr + 1);     //0012ff70

    /*
    *ptr 是 int 类型的指针常量
    *ptr + 1 是 int 类型的指针常量
    (int)ptr + sizeof(int) * 1
    */
    printf("%p\r\n", *ptr + 1);    //0012ff60 + 4 = 0012ff64

    /*
    ptr + 1 得到int[4]指针常量
    *(ptr + 1)是int类型的指针常量
    */
    printf("%p\r\n", *(ptr + 1)); //*(0012ff70) = 00000050

    /*
    *ptr是int类型的指针常量
    *ptr + 1 == (int)ptr + sizeof(int) * 1 == 0012ff60 + 4
    *ptr + 1 是 int类型的指针常量
    *(*ptr + 1)，得到int的标识符引用
    */
    printf("%p\r\n", *(*ptr + 1));           //*(0012ff64) == 00000020

    /*
    ptr + 1 是 int[4] 类型的指针常量
    *(ptr + 1) 是 int 类型的指针常量
    0012ff70 是 int 类型的指针常量
    *(ptr + 1) + 1 是 int 类型的指针常量
    0012ff70 + sizeof(int)  == 0012ff74
    */
    printf("%p\r\n", *(ptr + 1) + 1);        //0012ff74

    /*
    *ptr 是 int 类型的指针常量
    *ptr + 1 是 int 类型的指针常量
    *(*ptr + 1) == *(0012ff64)，得到 int 的标识符引用
    00000020 + 1  == 000000021
    */
    printf("%p\r\n", *(*ptr + 1) + 1);       //000000021

    /*
    ptr[1] == *(ptr + 1)，是int类型的指针常量
    ptr[1] + 1 ，是int类型的指针常量 +1
    0012ff70 + sizeof(int) = 0012ff74，是int类型的指针常量
    (ptr[1] + 1)[1] == *(*(ptr + 1) + 1)
    *(是int类型的指针常量 + 1)，得到int
    *(0012ff74 + 1) == *(0012ff78)，得到int
```

```
    ptr: 0012ff60
    ptr[1] == *(ptr + 1), 是int类型的指针常量
    ptr + 1 == 0012ff60 + sizeof(int[4])  == 0012ff70 ,得到int[4]的指针
    *(ptr + 1), 得到 int[4]
    int[4] 是一维数组,  所以*(ptr + 1)就是一维数组的标识符引用名
    *(ptr + 1)是 int[4]的第0个(int) 类型的指针常量
    *(ptr + 1)是 int 类型的指针常量 == ptr[1]是 int 类型的指针常量
    ptr[1] + 1, 得到int类型的常量
    (int)ptr[1] + sizeof(int) == 0012ff70 + 4 == 0012ff74
    (ptr[1] + 1)[1],相当于对int类型的指针做下标运算
    0012ff74 + sizeof(int) == 0012ff74 + 4 == 0012ff78
    将0012ff78解释为int输出
    */
    printf("%p\r\n", (ptr[1] + 1)[1]);        //00000070

    system("pause");
    return 0;
}
```

所有通过指针变量的操作，都是属于间接访问

指针运算需要注意的三个事项:

1、是什么类型的指针参与运算
2、怎么运算
3、运算后得到什么

Work

; 代码示例:

```
    //写出下面通过之指针进行各种的运算的运算步骤:
    int ary[2][4] = {
    {0x10, 0x20, 0x30, 0x40},
    {0x50, 0x60, 0x70, 0x80}
    };

    int (*p)[4] = ary;  // ary的地址

    printf("%p\r\n", p + 2);
    printf("%p\r\n", p[2]);
    printf("%p\r\n", p[2] + 2);
    printf("%p\r\n", p[2][2]);
    printf("%p\r\n", *p + 2);
    printf("%p\r\n", (p + 2) + 2);
```

```
    printf("%p\r\n", *(*p + 2) + 2);
    printf("%p\r\n", *p[2] + 2);
```

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int ary[2][4] = {
        {0x10, 0x20, 0x30, 0x40},
        {0x50, 0x60, 0x70, 0x80}
    };

    // int[4] *p = ary;
    int (*p)[4] = ary;      //0012FF60

    /*
    [数组名]是[数组第0个元素]类型的指针常量
    [二维数组]的元素是[一维数组]
    [对某类型的指针]做*（取内容）运算，得到[某类型]的标识符引用
    */

    /*
    指针p保存数组ary的首地址
    p是 int[4] 类型的指针常量
    p + 2 得到 int[4] 类型的指针常量
    (int)p + sizeof(int[4]) * 2 == 0012FF60 + 00000020 == 0012FF80
    */
    printf("%p\r\n", p + 2);            //0012FF80

    /*
    p[2] = *(p + 2)，是int类型的指针常量
    p + 2 == (int)p + sizeof(int[4]) * 2 == 0012FF80，得到int[4]的指针
    *(p + 2)，得到 int[4]
    int[4] 是一个一维数组，所以 *(p + 2)，就是一维数组的引用名
    *(p + 2) 是 int 类型的指针常量
    p[2] 是 int 类型的指针常量
    */
    printf("%p\r\n", p[2]);             //0012FF80

    /*
    p[2] 是 int 类型的指针常量

    p[2] + 2，得到int类型的指针常量

    p[2] + 2 == (int)p + sizeof(int[4]) * 2 +
```

```
                sizeof(int) * 2 == 0012FF60 +  00000020 + 00000008
                == 0012FF88
    */
    printf("%p\r\n", p[2] + 2);          //0012FF88

    /*
    p[2][2] == *(*(p + 2) + 2)

    p是 int[4] 类型的指针常量
    p + 2 得到 int[4] 类型的指针常量
    p + 2 == (int)p + sizeof(int[4]) * 2 == 0012FF60 + 00000010*2 ==
0012FF80
    p[2] = *(p + 2), 做*（取内容）运算, 是int类型的指针常量
    *(p + 2) == 0012FF80

    *(p + 2) + 2, 得到 int 类型的指针常量
    int(p) + sizeof(int[4]) * 2 + sizeof(int) * 2
        == 0012FF60 + 00000010*2 + 00000004*2
        == 0012FF80

    对int类型的指针常量(0012FF80)做*（取内容）运算，得到 int 的标识符引
用
    取0012FF80上的值（这里数组越界），通过内存可以看出该地址上对应的值
为：00000001
    */
    printf("%p\r\n", p[2][2]);           //00000001

    /*
    p是 int[4] 类型的指针常量
    int[4]是一个一维数组
    *p 是 int 类型的指针常量
    *p，得到 int 类型的指针常量
    *ptr + 2 同样得到 int 类型的指针常量
    所以：*ptr + 2 == int(p) + sizeof(int) * 2
                   == 0012FF60 + 00000004*2
                   == 0012FF68
    */
    printf("%p\r\n", *p + 2);            //0012FF68

    /*
    p是 int[4] 类型的指针常量
    p + 2, 同样是一个int[4] 类型的指针常量
    p + 2 == int(p) + sizeof(int[4]) * 2
          == 0012FF60 + 00000010*2
          == 0012FF80
    (p + 2) + 2, 同样是一个int[4] 类型的指针常量
    (p + 2) + 2 == (int(p) + sizeof(int[4]) * 2) + (sizeof(int[4]) * 2)
```

```
                == (0012FF60 + 00000010*2) + 00000010*2
                == 0012FF80 + 00000020
                == 0012FFA0
    */
    printf("%p\r\n", (p + 2) + 2);       //0012FFA0

    /*
    p是 int[4] 类型的指针常量
    int[4]是一个一维数组
    *p 是 int 类型的指针常量
    *p，得到 int 类型的指针常量
    *ptr + 2 同样得到 int 类型的指针常量
    所以：*ptr + 2 == int(p) + sizeof(int) * 2
                   == 0012FF60 + 00000004*2
                   == 0012FF68

    *(*p + 2) == *(0012FF68)
    *(*p + 2)，得到 int 的标识符引用
    *(*p + 2) + 2 == 取*p + 2地址上的值，后加2
                   == 00000030 + 2
    */
    printf("%p\r\n", *(*p + 2) + 2);       // 00000032

    /*
    p[2] == *(p + 2)
    *p[2] == *(*(p + 2))
    *p[2] + 2 == *(*(p + 2)) + 2

    p是 int[4] 类型的指针常量
    p + 2 == (int)p + sizeof(int[4]) * 2 == 0012FF80，得到int[4]的指针
    *(p + 2)，得到 int[4]
    int[4] 是一个一维数组，所以 *(p + 2)，就是一维数组的引用名
    *(p + 2) 是 int 类型的指针常量
    p[2] 是 int 类型的指针常量
    *(p + 2) == 0012FF80

    *p[2] == *(*(p + 2)),

    对p[2]做*（取内容）运算，得到int的标识符引用
    也就是取0012FF80地址上的值
    *p[2] + 2 == *(*(p + 2)) + 2
              == *(0012FF80)(取该地址上保存的值) + 2
              == 0012FFC0 + 2
    */
    printf("%p\r\n", *p[2] + 2);             //0012FFC2

    system("pause");
```

```
    return 0;
}
```