

2020春季学期C++整理3.0

1.关于指针的几种定义:

1. `int a`;表示一个内存空间, 这个空间用来存放一个整数 (`int`) ;
2. `int* a`;表示一个**内存空间**, 这个空间**用来存放一个指针**, 这个指针**指向一个存放整数的空间**, 即1中提到的空间;
3. `int * a`;表示一个内存空间, 这个空间用来存放一个指针, 这个指针指向一个存放指针的空间, 并且指向的这个空间中的指针, 指向一个整数。也简单的说, 指向了一个b)中提到的空间; *
4. `int (* a)[4]`;表示一个内存空间, 这个空间用来存放一个指针, 这个指针指向一个长度为4、类型为 `int`的数组; 和`int * a`的区别在于, ++、+=1之后的结果不一样, 其他用法基本相同。
5. `int (*a)(int)`;表示一个内存空间, 这个空间用来存放一个指针, 这个指针指向一个函数, 这个函数有一个类型为`int`的参数, 并且函数的返回类型也是`int`。

注意:

`int p[]`和`int (*p)[]`

前者是**指针数组**, 后者是**指向数组的指针**。

前: 指针数组;是一个**元素全为指针的数组**.

后: 数组指针;可以直接理解是指针,只是这个指针类型不是`int`也不是`char`而是 `int [4]`类型的数组

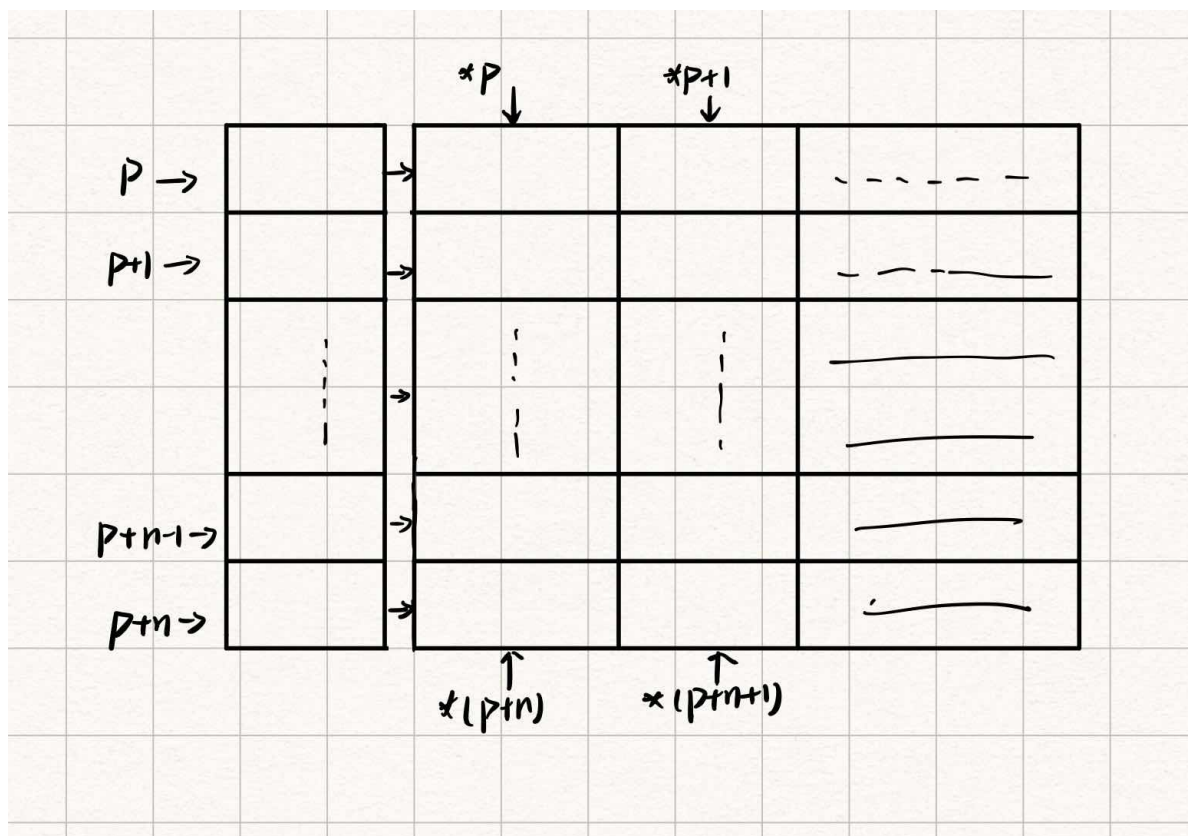
`int * p[4]`-----**p是一个指针数组**, 每一个指向一个`int`型的

`int (* q)[4]`-----**q是一个指针**, 指向`int[4]`的数组。

定义涉及两个运算符: “*” (间接引用)、“[]” (下标) , “[]”的优先级别大于“*”的优先级别。

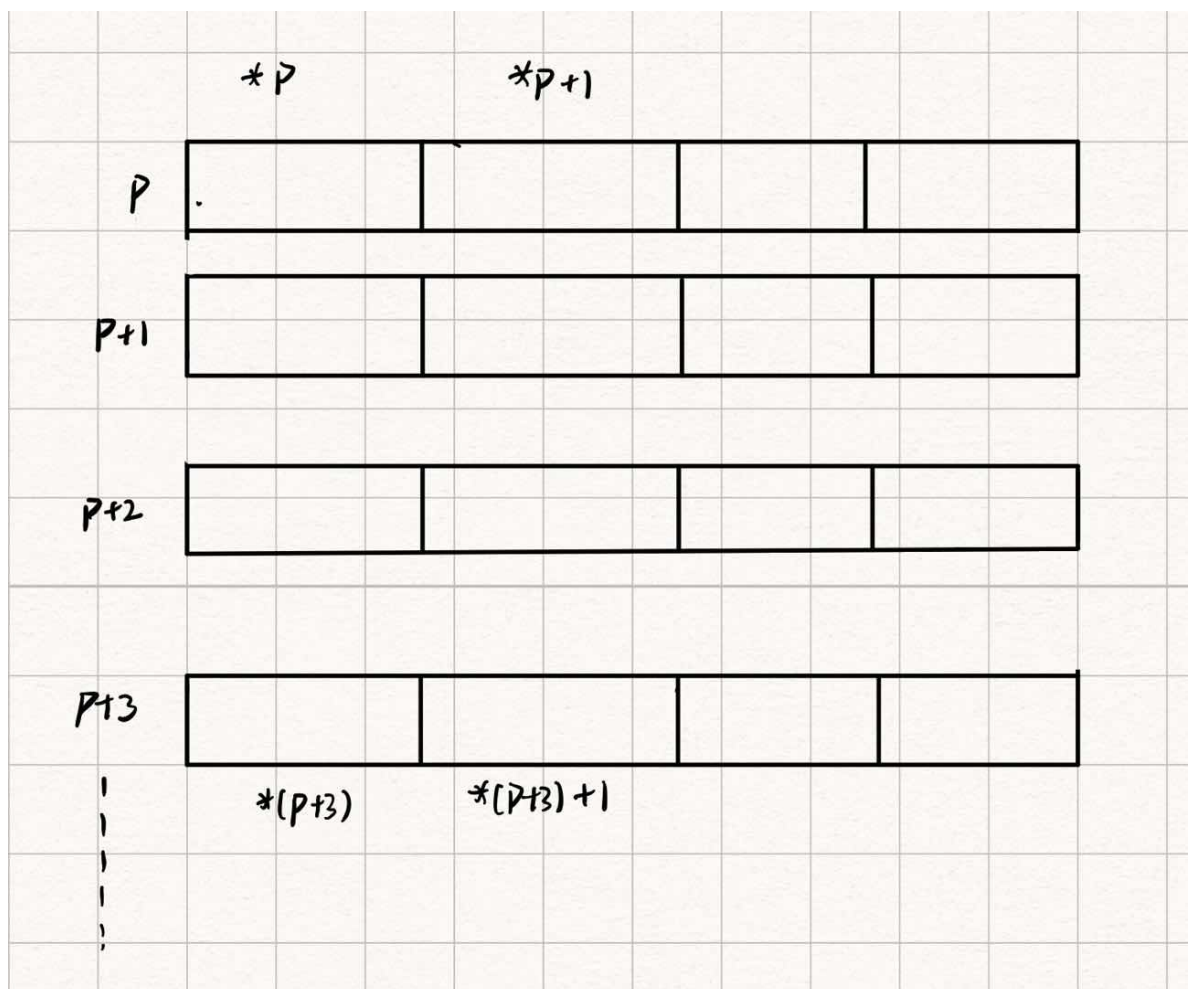
下面用几幅图来表示下:

对于`int *p[4]`:它首先是个大小为4的数组, 即`p[4]`; 剩下的“`int *`”作为补充说明, 即**说明该数组的每一个元素为指向一个整型类型的指针**



(PS:最后应该是 $*(p + n) + 1$)

再看一下int (*p)[4]:它首先是个指针，即*q，剩下的“int [4]”作为补充说明，即说明**指针q指向一个长度为4的数组**。



综上所述，我们可以将这两种指针定义出的p看作是一个二维数组来理解。

可以通过一些样例来说明:

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      int a[2][4]={2,5,6,8},{22,55,66,88}};
6
7      int c[4]={5,8,9,4};
8
9      int d[3]={23,12,443};
10
11     int *p[4],(*q)[4];
12
13     q=a;
14
15     *p=c;
16
17     *(p+1)=d;
18     return 0;
19 }
```

通过调试追踪可看出:

监视 1		
搜索(Ctrl+E)		
名称	值	类型
q	0x00cffa38 {2, 5, 6, 8}	int[4] *
p	0x00c99f4 {0xffffffff {???}, 0xffffffff {???}, 0xffffffff {???}, 0x...	int *[4]
*p	0xffffffff {???}	int *
*q	0x00cffa38 {2, 5, 6, 8}	int[4]
*(q + 1)	0x00c99f48 {22, 55, 66, 88}	int[4]
*(q + 1) + 1	0x00c99f4c {55}	int *

确实可以理解为一个二维数组的形式。

2.标注注释:

注释一:

注释1

```
int main()
{
    int a[4]={1,2,3,4};

    int *ptr=(int*)(&a+1);
    printf("%d",*(ptr-1));
}
```

Q a: 是指的 int[4]

定义的为 int()[4]*

↓ 一个指向一个 int 型数组的指针

这一步完成后 ptr 指向类型为 int()[4]*

指向在 a[4] 的下一空间

可以认为 ptr[0] → a[4]

尽管非法 但空间存在

强制类型转换

↑ a[3]

注释二:

注释2

```
1  int main()
2  {
3      int a[5][5];
4      int(*p)[4]; → 定义一个指针 p 指向 int[4] 的数组
5      p = (int(*)[4])a; 强制转换为 int[4] 型
6      printf("%p,%d\n", &p[4][2] - &a[4][2], &p[4][2] - &a[4][2]);
7
8      system("pause");
9      return 0;
10 }
```

地址值差

相当于在内存中二者距离多少

注释三:

注释3

```
1  int main()
2  {
3      char *a[] = { "work", "at", "alibaba" };
4      char **pa = a;
5      pa++;
6      printf("%s\n", *pa);
7      system("pause");
8      return 0;
9  }
```

二维数组 → 二级 pointer

这个语句在 VS19 不能运行！
在 Dev C++ 中 Warning

"at"