**初识指针（尽量用内存跟踪）**

1. 指针变量的定义
2. 定义指针
   * 1. 我们定义普通的变量 int a； 指针变量 int \*pointer;
     2. 赋值 pointer = &a; 我们需要的是地址哦且pointer钱买你不要再加 \* 了
     3. 一个指针变量只能指向同类型的变量（int -> int; float -> float; const int -> const int）
3. 指针变量的引用
4. 指针变量中只存放地址
   * 1. 不要把pointer = 4； 这样子类型的，记得是地址【4 是整数】
5. 指针运算符（很重要的哦）
   * 1. 取值 \*
     2. 取值 &
     3. 前方高能（可以用VS来内存跟踪一下 Shift+F9）
        1. (\*p)++ ==> 是对值进行++
        2. \*p++ ==> 是取完值对地址++
        3. 引用和指针之间得关系（在C 和 C++ 中是不一样的）
           1. &\*p ==> 对值取地址，所以等价于p
           2. \*&a ==> 先取地址再取值，等价于p
6. 指针作为函数参数
7. 例如，我们现在要调换a, b 的值
   * 1. 代码

void swapAB(int \*a, int \*b){

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int main(){

int a = 3;

int b = 4;

int \*p1 = &a;

int \*p2 = &b;

printf("a = %d, b = %d\n", a, b);

swapAB(p1, p2);

printf("a = %d, b = %d\n", a, b);

}

1. 我们来试想一下，可不可以用下面的方式来实现
   * 1. 代码

void swapAB(int \*a, int \*b){

int \*temp;

temp = a;

a = b;

b = temp;

}

* + 1. 答案是不行的，为什么呢？
       1. temp地址 指向 a的地址
       2. a 的地址指向 b 的地址
       3. b 的地址指向 temp 即 a 的地址
       4. 那我们可以看出，知识地址的指向而已，实质a， b 的值都没有操作到，而且形参和局部变量在函数调用结束就释放了，所以没有啥实质性的东西

1. 总结
   1. Swap() 不能改变实参指针锁指向的值，即 p1 始终指向 a， p2 始终指向 b
   2. Swap() 可以改变实参指针所指向的变量的值