**将数组作为参数进行传递**

有两种传递方法，一种是function(int a[]); 另一种是function(int \*a)

这两种两种方法在函数中对数组参数的修改都会影响到实参本身的值！

对于第一种，根据之前所学，形参是实参的一份拷贝，是局部变量。但是数组是个例外，因为数组的数据太多了，将其一一赋值既麻烦又浪费空间，所以数组作为参数传递给函数的只是数组首元素的地址，数据还是在内存里的，函数在需要用到后面元素时再按照这个地址和数组下标去内存查找。也就是说后面的元素根本没到函数里来。所以，这里也不能在test()函数内部用sizeof求数组的大小，必须在外面算好了再传进来。

对于第二种，则是传址调用，无需再说。

这里还有几点需要注意：

1.在以上两个函数的形参前面加上const则表示整个数组只读，而不是只有首地址对应存储的数据只读。

2.第二种形式不能用C++11中的for...auto来循环打印。

3.数组的大小要用sizeof()来求，不能用.size()，因为.size()只有struct 或者union才能用， vector算是struct！

4.如果在函数内部又声明一个int\* tmp类型的变量，然后把p赋值给tmp, 通过tmp修改数数组也是一样，都会修改实参本身！

测试代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | #include <stdio.h>  #include <algorithm>  using namespace std;    void test1(int[], int size);  void test2(int \*p, int size);  //void test2(const int \*p, int size);    int main(void)  {      int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};      int size = sizeof(a)/sizeof(int);      /\*这里打印出的a值与test1(),test2()中的p值相等       \*即地址值相同\*/      printf("%p \n", a);      //test1(a, size);      test2(a, size);      int i;      printf("main: ");      for(i = 0; i < size; ++i)      {          printf("%d ", a[i]);      }  }    void test1(int p[], int size)  {      printf("%p \n", p);      p[4] = 111;      printf("test1: ");      int i;      for(i = 0; i < size; ++i)      {          printf("%d ", p[i]);      }      printf("\n");  }    void test2(int \*p, int size)  {      printf("%p \n", p);      \*(p+4) = 222;      printf("test2: ");      int i;      for(i = 0; i < size; ++i)      {          printf("%d ", \*(p+i));      }      printf("\n");  } |