指针变量作为函数参数

函数的参数可以是指针类型，它的作用是将一个变量的地址传送到另一个函数中。

指针变量作为函数参数与变量本身作函数参数不同，变量作函数参数传递的是具体值，而指针作函数参数传递的是内存的地址。

把数组作为参数传入函数中，实际上就是利用了传递指针（即传递数组的首地址）的方法。通过首地址，我们可以访问数组中的任何一个元素。

对于指向其他类型变量的指针，我们可以用同样的方式处理。

例：变量交换

【问题描述】

输入 a、b 两个整数，交换两个变量的值。

【正确程序】

```
#include<iostream>
using namespace std;
void swap(int *q1, int *q2)
{
    int t = *q1;
    *q1 = *q2;
    *q2 = t;
}
int main()
{
    int a, b;
    cin >> a >> b;
    int *p1 = &a;
    int *p2 = &b;
    swap(p1, p2);
    cout << a << " " << b << endl;
    cout << *p1 << " " << *p2 <<endl;
    return 0;
}
```

【输入样例】

```
1 2
```

【输出样例】

```
2 1
2 1
```

【分析】

将 a 和 b 的地址传给函数 swap()，然后在函数中通过地址得到变量 a 和 b 的值，并且对它们进行修改。当退出函数时，a 和 b 的值就已经交换了。

【错误程序】

```cpp
#include<iostream>
using namespace std;
void swap(int x, int y)
{
    int t = x;
    x = y;
    y = t;
}
int main()
{
    int a, b;
    cin >> a >> b;
    int *p1 = &a;
    int *p2 = &b;
    swap(a, b);
    cout << a << " " << b << endl;
    cout << *p1 << " " << *p2 <<endl;
    return 0;
}
```

【输入样例】

```
1 2
```

【输出样例】

```
1 2
1 2
```

【分析】

调用 swap(a, b)并没有将 a 和 b 的值互换。这是因为在传入变量 a 和 b 的时候，是将 a 赋值给函数中的形参 x，将 b 赋值给形参 y。接下来的操作就与 a 和 b 无关了，函数将变量 x 和 y 的值互换，然后退出函数。因此无法对传进来的变量进行修改。

【总结】

将指针传入函数与将变量传入函数的区别在于：前者是通过指针来使用或修改传入的变量；而后者是将传入的变量的值赋给新的变量，函数对新的变量进行操作。

用指针变量作函数参数，在被调函数的执行过程中，应使指针变量所指向的参数值发生变化。这样，函数在调用结束后，其变化值才能保留回主调函数。

函数调用不能改变实参指针变量的值，但可以改变实参指针变量所指向变量的值。

函数指针

程序的数据和代码都是存放在内存空间中的。

一个指针变量通过指向不同的数据的地址，就可以对存放数据的变量进行操作。

同样，函数的入口地址也是可以用指针访问的。在 C++中，函数名就代表了函数的地址。

有些函数在编写时对要调用的子函数无法确定，只有在执行时才能根据实际情况决定调用的子函数。比如下面的 sort 函数，其中的比较函数 cmp 就是根据需要传给 sort 的。这里其实就是传递了函数指针。

```
1. sort(a, a+n, cmp);
```