#include <iostream>
using namespace std;
int main()
{
    double * p3 = new double [3]; // space for 3 doubles
    p3[0] = 0.2;                  // treat p3 like an array name
    p3[1] = 0.5;
    p3[2] = 0.8;
    cout << "p3[1] is " << p3[1] << ".\n";
    p3 = p3 + 1;                  // increment the pointer
    cout << "Now p3[0] is " << p3[0] << " and ";
    cout << "p3[1] is " << p3[1] << ".\n";
    p3 = p3 - 1;                  // point back to beginning
    delete [] p3;                 // free the memory
    return 0;
}

Here is the output:

p3[1] is 0.5.
Now p3[0] is 0.5 and p3[1] is 0.8.

As you can see, arraynew.cpp uses the pointer p3 as if it were the name of an array, with p3[0] as the first element, and so on. The fundamental difference between an array name and a pointer shows in the following line:

p3 = p3 + 1; // okay for pointers, wrong for array names

You can't change the value of an array name. But a pointer is a variable, hence you can change its value. Note the effect of adding 1 to p3. The expression p3[0] now refers to the former second element of the array. Thus adding 1 to p3 causes it to point to the second element instead of the first. Subtracting one takes the pointer back to its original value so that the program can provide delete [] with the correct address.

The actual addresses of consecutive ints typically differ by two or four bytes, so the fact that adding 1 to p3 gives the address of the next element suggests that there is something special about pointer arithmetic. There is.