【C++】Day eleven(2)

Class	C++
Date	@November 29, 2021
Material	
Series Number	
Summary	

【Ch3】Multidemensional Arrays

Using a Range for with Multidimensional Arrays

We can loop through a 2-D array using the following code:

```
size_t cnt = 0;
for(auto &row : ia)  //for every element in the outer array
  for(auto &col : row) { //for every element in the inner array
    col = cnt++; //give this element in the next value, and then increment cnt
  }
```

In the example above, we used references as our loop control varaibles because we wanted to change the elements in the array. However, there is a deeper reason for using references. Consider the following loop:

```
for(const auto &row : ia) //For every row in the outer array
  for(auto col : row) //For every column in the inner array
    cout << col << endl;
```

This loop does not write to the lements, yet we still define the control variable of the outer loop as a reference. We do so in order to avoid the normal array to pointer conversion. If we neglect the reference and write these loops as:

```
for(auto row : ia)
  for(auto col : row)
```

Our program will not compile. As before, the first for iterates through `ia`, whose elements are arrays of size 4. Because row is not a reference, when the compiler initializes row it will convert each array element to a pointer to that array's first element. As a result, in this loop the type of row is `int*`. The inner for loop is illegal, Despite our intentions, that loop attempts to iterate over an int*.

Note: To use a multidimensional array in a range for, the loop control varaible for all but the innermost array must be references.

Exercise

Exercises Section 3.6

Exercise 3.43: Write three different versions of a program to print the elements of `ia`. One version should use a range `for` to manage the iteration, the other two should use an ordinary `for` loop in one case using subscripts and in the other using pointers. In all three programs write all the types directly. That is, do not use a type alias, `auto`, or `decltype` to simplify the code.

Range for:

```
for(int (&row)[4] : ia) {
  for(int col : row)
    cout << col << " ";
  cout << endl;
}
```

Subscript:

```
for(int i = 0; i != 3; ++i) {
  for(int j = 0; j != 4; j++)
    cout << ia[i][j] << " ";
  cout << endl;
}
```

Pointers:

```
for(int (*outPtr)[4] = arr; outPtr != arr + 3; outPtr++) {
  for(int *innerPtr = *outPtr; innerPtr != *outPtr + 4; innerPtr++)
```

```
    cout << *innerPtr << " ";
   cout << endl;
}
```