C++ Grammar: Array, Scope of Variables

Yizuo Chen and Yiyou Chen

November 22, 2016

At last meeting we talked about some basic C++ grammars, and today we will continue our C++ grammar learning plan. First we will use array to solve the “Fibonacci Sequence” we did at last meeting, and then we will talk about the scope of variables. While we’re learning C++ language, we also try to put some algorithm thinking inside at the same time. For example, at this meeting we put a simple algorithm to count the appearance of difference numbers in a “n * n” square table.

The code for today’s lecture:

```
#include <bits/stdc++.h>
using namespace std;
int occurred[1005];
int main() {
/*int i = 3;
for(int i = 1; i <= n; i++){

}
//sequence: 1 1 2 3 5 8 13 21 34 55 ...
int n;
cin >> n;
int left = 1, mid = 1, right = 2;
while(i <= n){
  right = left + mid;
  left = mid;
  mid = right;
  i++;
  //break;
}
if(n == 1) cout << 1 << endl;
else if(n == 2) cout << 1 << endl;
else if(n == 3) cout << 2 << endl;
else cout << right << endl;*/
/*int a[200];
a[0] = 0;
a[1] = 1;
int n;
cin >> n;
for(int i = 2; i <= n; i++){
  a[i] = a[i - 1] + a[i - 2];
}
cout << a[n] << endl;*/
/*1 23 2 555  666 ...   a_i <= 1000.
n <= 1000
a_1, a_2, a_3,
input:
5
```

```
1 1 3 500 6
output:
4*/
int n, ans = 0;
int a[1005], b[1005];
//memset(a, 0, sizeof(a));
cin >> n;
//cin >> column >> rows;
for(int i = 1; i <= n; i++) cin >> a[i]; // != not equal
for(int i = 1; i <= n; i++){
  if(occurred[a[i]] == 0){ // never show up before
   ans++; // count as a new one
   b[ans] = a[i];
   occurred[a[i]] = 1; // avoid double counting
  }
}
cout << ans << endl;
for(int i = 1; i <= ans; i++) cout << b[i] << " ";

return 0;
}
```

Explanations:

```
int a[200];
```

This is the way to define an array. What is an array? It's a tool to save a certain size of elements. After we define an array, our computer will automatically give us a certain size of space for us to save our elements. The way to define an array is:

DataType array_name[max_array_size]

In this case, we defined an array "a" of a max size of 200 with integer elements saved inside. We can regard an array as a train with a lot of carriages inside.

pos	0	1	2	3	4	5	6	7	8
a[pos]	5	7	9	11	45	5645	234234	234234	565

For each position in the array, a[pos] represents a value that you save inside this position.

Be careful, array always starts at pos = 0; thus, the max pos you can have with an array with max_array_size equals to n is n – 1.

```
a[0] = 0;
a[1] = 1;
```

This is the way to give values to array elements. After this, when we use a[0], it always gives us value of 0; when we use a[1], it always gives us value of 1.

```
for(int i = 2; i <= n; i++){
  a[i] = a[i - 1] + a[i - 2];
}
```

We can solve Fibonacci Sequence problem easily by using array. For each term i, the value of this term equals the sum of its previous two terms. We can express this pattern in a[i] = a[i - 1] + a[i – 2]. If we want to get value of term n, we can simply loop it from 2 to n, which gives us the value of term n.

Q: How do you know it works?

A: we can use a table to simulate the whole process:

i	0	1	2	3	4	5	6	7	8	9
a[i]	0	1	1	2	3	5	8	13	21	34

Obviously a[i] is a Fibonacci sequence. By using array, we can simply accomplish building a Fibonacci sequence using its definition.

```
int n, ans = 0;
int a[1005], b[1005];
memset(a, 0, sizeof(a));
```

Now let's talk about the scope of the variables. There're mainly two types of variables. One is called the global variables and the other is called the local variables. We can tell from the name that global variables must be more dominate than local variables. Actually, the difference between these two variables is:

Global variable: it must have been cleared by system when you define it

Local variables: you don't what's inside after you defined it: may not be empty.

Q: What does "cleared" mean?

A: "to be cleared" means there's nothing inside. For an integer, it equals to 0; for a string, it has nothing inside (size of 0). For a character, it doesn't have anything; for a Boolean variable, it is false, etc.

Q: How about local variables?

A: After you defined int a; "a" may have values like 234328943, 23423, 23456, 234,8346, 32475, or others. Basically it may be a random number that you've never seen. For a string, it may have already had a certain length before you put elements inside, etc.

We can clear local variable arrays in this way:

```
memset(name_of_array, 0, sizeof(name_of_array));
```

If we want to clear array "a",

```
memset(a, 0, sizeof(a));
```

```
for(int i = 1; i <= n; i++){
  if(occurred[a[i]] == 0){ // never show up before
   ans++; // count as a new one
   b[ans] = a[i];
   occurred[a[i]] = 1; // avoid double counting
  }
}
```

In order to record what numbers appears in an array, we use another array occurred[] to record the appearance of elements in a[], and b[] to record the value of these elements.

If a[i] has never occurred before, the occurred[a[i]] will be 0; otherwise, occurred[a[i]] will equal to 1. If a[i] has never appeared before, we add a[i] into array b[]. Finally, all the different values will be added into array b[].

Be Sure to know:

Difference between global variable and local variable **Define and use array to solve problems** **How to record the appearance of numbers**