C++ Grammar: Arrays, Void and Function

Yizuo Chen and Yiyou Chen

November 29, 2016

Based on the 1-D array, we talked about the more dimensional arrays and their applications. Then we discussed C++ void and functions, which help to clear the structure of the program. Please check the problem and our analysis if you haven't looked through them.

Links: http://sutherland-programming-club.co.nf/week2.html (apple man and easy task from Codeforces)

Today's Code:

```
#include <bits/stdc++.h>
using namespace std;
int n;
int num;
char a[102][102];   // size = 1024^2 integers
void readdata(){
int i, j;
cin >> n;
for(i = 1; i <= n; i ++){
  for(j = 1; j <= n; j ++){
   cin >> a[i][j];
  }
}
}
void checkread(){
int i, j;
  for(i = 1; i <= n; i ++){
  for(j = 1; j <= n; j ++){
   cout << a[i][j];
  }
  cout << endl;
}

}
int work(){
int i, j;
for(i = 1; i <= n; i ++){
  for(j = 1; j <= n; j ++){
   num = 0;
   if(a[i + 1][j] == 'o'){
    num ++;
   }
   if(a[i][j + 1] == 'o')
    num ++;
   if(i && a[i - 1][j] == 'o')
    num ++;
   if(j && a[i][j - 1] == 'o')
    num ++;
  // cout << num << endl;
```

```
   if(num % 2 == 1){
    return 0;
   }
  }
}
return 1;
}

int main() {
// your code goes here
   // i = row, j = column
int answer;
readdata();
// checkread();
answer = work();  //answer = 0, NO; answer = 1, yes
if(answer == 0)
  cout << "NO" << endl;
else
  cout << "YES" << endl;

return 0;
}
```

Explanations:

```
char a[102][102];
```

If 1-dimensional array is to give a[i] (0 <= i <= n) for each i a value, then k-dimensional array is to give $a[p_1][p_2][p_3]...[p_k]$ ($p_i \in [l_i, r_i]$) for each p_i a value. We can regard each "[]"s as a directional vectors, then $a[p_1][p_2][p_3]...[p_k]$ is a final vector which has a unique combination of bases. We drew one dimensional array into a linear table. In this case, if we want to draw two-dimensional array, we can draw it in a two-dimensional table:

a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]
a[2][1]	a[2][2]					
a[3][1]	a[3][2]					
a[4][1]	a[4][2]					
a[5][1]	a[5][2]					
a[6][1]	a[6][2]					
a[7][1]	a[7][2]					a[7][7]

You can try to fill the rest blocks. Instead of a[102], which you can only save a[0], a[1], a[2] … a[101], you can save a[0][0], a[0][1] … a[0][101], then a[1][0], a[1][1] .. a[1][101], until a[101][101]. So a[102][102] has a size of 102 * 102.

A very good application of two-dimensional array is to save a table like a checkerboard. If we make the first "[]" save row number and second "[]" save column number, a[#row][#column], we can easily express every cell on the board; after that we can use brute force to solve this problem.

```
int work(){

return 1;
}
```

This is an example of C++ function. Hope this reminds you of the main function. Yes, main function is also a function. It is a structure that you put some code inside rather than put all of them in the main function. The basic structure of a function is:

```
Data_Type function_name(value1, value2, … value n){ //ex. int work(int a, int b)

      Code here.

      return return_value;

}
```

Q: what's the return_value for?

A: You will put something like `ans = work(1, 2);` in the main function, so the return_value will go to the position of work(1, 2).

Q: what are the values in "()"?

A: They're the local variables which can only be used within the function. These values come from the values put in the main function.

```
void readdata(){

}
```

The only difference between void and function is that void doesn't have return value. So in the main function, you don't need to write ans = readdata(); instead, readdata();

Be Sure to know:

function and void **More dimensional array**