C++ Grammar: Input & Output, If & Else, For & While

Yizuo Chen and Yiyou Chen

November 15, 2016

Before really starting to solve programming problems or trying contests' problems, we need to learn at least one programming language. Here we'll briefly talk about C++ grammars.

In this club, we're trying to learn grammars for competitive programming. We will first put complete codes, then explain grammars one by one.

Code from 11/15/2016:

```
#include <bits/stdc++.h>

using namespace std;

int main() {

//int a = 1, b = 5; // define variables double, float, char

//cin >> a >> b; // read

//cout << a + b; //print "Hello world" directly print

//n th number in the sequence 1 1 2 3 5 8 13 21...

int n, left, middle, right;

left = 1;

middle = 1;

right = 2;

cin >> n;

for(int i = 3; i <= n; i++){ // for loop
```

```
  right = left + middle;

  left = middle;

  middle = right;

}

if(n == 1) cout << 1;

else if(n == 2) cout << 1;

else if(n == 3) cout << 2;

else {

  cout << right << endl;

}

//n th number in the sequence 1 1 2 3 5 8 13 21...

n = 1;

while(n >= 1){ // while loop

  n++;

  if(n > 10){

   break;

  }

  cout << n << endl;

}

}
```

Explanations:

A. The basic structure of a C++ program:

```
#include <stdc++.h>

Using namespace std;

int main(){



      return 0;

}
```

```
 #include <stdc++.h>
```

This line is the header file to run preprocessors which is necessary for a C++ program.

```
 using namespace std;
```

This line is also necessary for a C++ program. It is used to declare the elements of the C++ standard library.

```
int main(){

      return 0;

}
```

This is the main function of the whole program. When the program begins to run, it will always go to the main function first. You can put expressions (like your orders) inside to make the computer do what you want it to do.

```
return 0;
```

It this shows the ending of the main function. After “return 0” in the main function appears, the whole program will stop and end.

Q: Why it appears on my screen #include <iostream> rather than #include <bits/stdc++.h>, what’s the difference between these two?

A: #include <bits/stdc++.h> is a new header file which contains almost all the header files in C++. So you can use this one line to replace tons of lines of header files.

```
#include <iostream>
  #include <cstdio>
  #include <cstdlib>
  #include <algorithm>
  #include <cstring>
  #include <queue>
  #include <vector>
  #include <stack>
  #include <ctime>
  #include <memory>
  #include <cmath>
  #include <bitset>
```

```
    #include <set>
```

Etc.

You might type all the head files above to use many data structures and grammars, but you could also just type #include <bits/stdc++.h> instead.

```
int a = 1, b = 5; // define variables double, float, char

Define the variables:

char ch; // character type: like ‘a’, ‘b’, ‘A’, ‘B’, ‘1’, ‘+’

int a;      // interger type: 1, 2, 3, 4, 5, 6

double d;  // float type: 1.0, 3.1415, 5.1, 2.7
```

```
 cin >> a >> b;

 cout << a + b;

 cout << n << endl;
```

This is the standard input and output form. cin read variable a & b, notice that between a and b use “>>”; cout output the sum of a and b, notice “<<”. If we remove the “c”, it becomes word “in” and word “out”, which helps you to memorize them. “cout << endl;” means output a newline after the end.

So instead of “6”, the output with “endl” is:

“6

newline”

Q: Can we directly use ‘+’ in C++?

A: Yes. ‘+’ ‘-‘ ‘*’ ‘/’ operations are the same forms as you learned in math class.

```
left = 1;

middle = 1;

right = 2;
```

This is to give the variable specific values. In this example, it makes left equal to 1, middle equal to 1, right equal to 2.

Q: Isn’t it the equal sign?

Ans: right, this is a special case that confuses people. In C++ grammar, ‘=’ means give value to a variable, while ‘==’ means equal to.

```
if(n == 1) cout << 1;
else if(n == 2) cout << 1;
else if(n == 3) cout << 2;
else {
  cout << right << endl;
}
```

This is the “Condition structure”. Inside “()” after “if” and “else if” is the condition, “else” means other cases except “if”.

(Someone said, “This is pretty easy to understand!”, which is true because this was the only one that they didn’t have any questions.)

```
  for(int i = 3; i <= n; i++){

  }
```

For loop. How do we write the code if we want to repeat doing something over and over? The answer is that we can use a loop. This is an example of the for loop: inside () is the controlling condition, meaning all the code inside {} will run repeatedly whenever the conditions inside () is true. “for loop” specially, the structure is:

for(initial condition; confining condition; changing condition){}

in this case, the initial value of variable “i” is 3,

confining condition that variable “i” could not be larger than n; otherwise, the loop will stop;

every time “i” increases by 1.

If we look at value of “i”:

It starts from 3, we check if “i” is smaller or equal to n, because it’s true, “i” becomes 4;

Then we check if “i” is smaller or equal to n again, then “i” becomes 5;

So the value of variable “i” is:

3, 4, 5, 6, 7, … n – 2, n – 1, n

Total n – 3 + 1 = n – 2 times. So all the code inside {} repeated n – 2 times.

Q: Can we just put int before “i” to define new variable i?

A: Yes, but this way to define “i” can only be used in for loop.

Q: What does “i ++” mean?

A: it means “i = i + 1”, also means “i” increases by 1.

```
while(n >= 1){
  n++;
  if(n > 10){
   break;
  }
  cout << n << endl;
}
```

This is the other type of loop called “while loop”. In the “()” is the confining condition. So the loop will repeat while the condition in the “()” is satisfied. In this case, the program will output:

```
2
3
4
5
6
7
8
9
10
```

Q: What does “break” mean?

A: It is the way to break the loop, meaning when “break” appears, the loop ends. In this case, when n is greater than 10, the loop will not repeat anymore.

Q: What if we didn't put "break" inside?

A: The loop will repeat forever (never get out of the loop). What will happen if the loop repeats endlessly? The program will run infinite time. This is not allowed during your coding process. So always check if there's any "infinite loop" in your program.

```
left = 1;
middle = 1;
right = 2;
cin >> n;
for(int i = 3; i <= n; i++){ // for loop
  right = left + middle;
  left = middle;
  middle = right;
}
if(n == 1) cout << 1;
else if(n == 2) cout << 1;
else if(n == 3) cout << 2;
else {
  cout << right << endl;
```

When we are trying to solve problems or learn algorithms, a very useful way is to calculate some small data manually. Because we cannot tell what this program is doing, we can build a table to see the change of values of the variables.

Case number	Left	Middle	Right
1	1	1	2
2	1	2	2
3	2	3	3
4	3	5	5

5	5	8	8
6	8	13	13
7	13	21	21
8	21	34	34
9	34	55	55

This is a very simple algorithm – find a "Fibonacci Sequence" without using an array. There is a special case:

If n == 1, the answer is 1 rather than 2.

So we use a "if" to solve this special case.

Be Sure to know:

If & else;   for loop;   while loop;   define variables;   basic variable types;

Input and output;   C++ basic structure;