Programming - Control Structures C++

José Mauricio Sevilla[1]
Andrés Camilo Sevilla[1]

[1]Universidad Nacional de Colombia

9 of October of 2015

Introduction
Structures
- if and else if
- for
- while
- do while
- switch

Jump Statements
- break
- continue

Functions
References

Introduction

At first, we wrote some instructions in C++ by making definitions, initializations, comparisons and so on, but what if we need to evaluate cases or repeat some calculations?

Well, those commands are a essential part of a programming language, in C++ exist:

- `if`
- `else`
- `for`
- `while`
- `do - while`
- `switch`

if and else

An `if` conditional is used to execute a statement, or a block, *if, and only if*, a certain condition is fulfilled. So a block

```
{ statement1; statement2; statement3;...}
```

are executed after evaluating the condition

```
if(condition)
```

sometimes we need to do any process by evaluating other condition, but only if the first one isn't true, so that's an `else if()`

```
if(condition)
.
.
.
else if(condition)
```

So, if what we want is to run some code lines if no one of the conditions we evaluate are true, we just use `else`, so that means like *otherwise do*, for example:

```
#include<iostream>
int main(){
int a,b;
a=5;
b=10;
if (a==5){
std::cout<<"in if"<<std::endl;
}else if(b==10){
std::cout<<"in else if"<<std::cout;
}else{
std::cout<<"in else"<<std::endl;}
return 0;
}
```

what does it print?, let's try.

for

A `for` means a loop, a repetition of some code lines, but that doesn't mean that we are *doing the same thing*; but in the loop, the variables can change their values, so, the syntax goes as follows

```
for(initialization; condition; increase - decrease)
```

the `for` and `if` are the most important structures, due to the rest of them can be written from they, so we're going to learn them, but more often we'll use `for` and `if`.

so, let's try two ways of writing a for loop, so, first

```
#include<iostream>
int main(){
int a,b;
a=5;
b=10;
for (int i=0; i<b;i++){
std::cout<<a<<std::endl;
a++;}
return 0;}
```

what happen if we replace that for and use the next instead

```
for (int i=0, int n=10; n>=i;n--,i++){
std::cout<<i<<n<<std::endl;
}
```

while

The while loop has a syntax like:

while(condition), for example

```
#include<iostream>
int main(){
int a,b;
a=5;
b=10;
while(a<b){
std::cout<<a<<std::endl;
a++;}
return 0;}
```

This loop first evaluate the condition, if it's true, run the commands, if it isn't true, continues the program.

do while

But, sometimes we want that at leas one time the commands in the loop run, so that's what do while does.

```
#include<iostream>
int main(){
int a,b;
a=5;
b=10;
do{
std::cout<<a<<std::endl;
a++;
}while(a<b);
return 0;}
```

but if we said that any other command could be written as a combination of a if and a for, how do we write a do while with them?

Range-Based for loop**

```
// range-based for loop
#include <iostream>
#include <string>
int main ()
{
   string str {"Hello!"};
   for (char c : str)
   {
      std::cout << "[" << c << "]"<<std::endl;
   }
}
```

switch

swirch is the last structure we're going to see, and it's very simple, it works as an else if, for example

```
#include <iostream>
int main ()
{int x=1;
switch (x) {
  case 1:
    std::cout << "a";
    break;
  case 2:
    std::cout << "b";
    break;
  default:
    cout << "error";
  }  return 0;}
```

```
#include <iostream>
int main ()
{
int x=1;
if (x == 1) {
  std::cout<<"a";
}
else if (x == 2) {
  std::cout << "b";
}
else {
  std::cout<<"error";
}      return 0;}
```

Jump Statements

In the last frame, when we use a switch, we also use a `break`, that one is part of what is called *jump statements*, here, we'll study two of them, `break` and `continue`

```
// break loop example
#include <iostream>
int main (){
for(int n=10; n>0; n--){
 std::cout << n << ", ";
 if (n==3){
   std::cout << "---";
   break;
  }
 }
return 0;}
```

```
// continue loop example
#include <iostream>
int main ()
{
  for (int n=10; n>0; n--) {
    if (n==5) continue;
    std::cout << n << ", ";
  }
  std::cout << "\n";
}
```

Functions

The idea of a function in programming is very similar than the defined in mathematics, that needs a set of input and a set of output

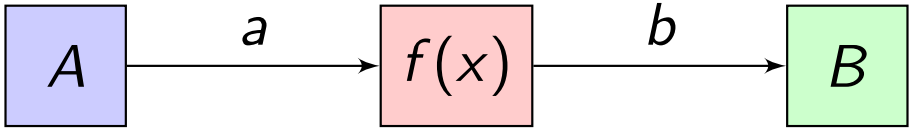

Figure 1: One Variable Function

Where A is the set of input and a belongs to it, and B is the output set and b it's element, so, we understand that f has it's domain and range, and that it doesn't work for any set outside it, the same happens in computing, but the sets of input and output are the kind of variables.

References

CPLUSPLUS.COM *http://www.cplusplus.com/doc/tutorial/control/*

CERN *https://twiki.cern.ch/twiki/pub/Main/CollaborativeProjects/Shell.pdf*, 2015

CERN *https://twiki.cern.ch/twiki/pub/Main/CollaborativeProjects/Preprogramming.pdf*, 2015

CERN *https://twiki.cern.ch/twiki/pub/Main/CollaborativeProjects/Basics.pdf*, 2015