Programing Basics C++

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

[1]Universidad Nacional de Colombia

9 of October of 2015

Structure of a Program

We've already mentioned how is the way to write down a code in C++, but in this presentation we are going to use it a lot, that's why we're going to review it; let's take a look back to the structure.

```
//Random Program
#include<library>
//Functions Definition
int main(){
//Code
.
.
.
return 0;
}
//Functions Description
```

Let's Try It

There is a program that is traditional made, let's do it, so, what we want for start, is print *Hello World*:

Let's Try It

There is a program that is traditional made, let's do it, so, what we want for start, is print *Hello World*:

```
//My first program
#include<iostream>
int main(){
std::cout<<"Hello World"
return 0;
}
```

How Do We Use It

Once the program is done, How do we use it?.

How Do We Use It

Once the program is done, How do we use it?.
First of all, we have to save it in a known folder, for example:

`~/Documents/Programming/examples`

How Do We Use It

Once the program is done, How do we use it?.
First of all, we have to save it in a known folder, for example:

```
~/Documents/Programming/examples
```

If that folder does not exist, we create it by typing on the prompt:

```
mkdir ~/Documents/Programming/examples
```

How Do We Use It

Once the program is done, How do we use it?.
First of all, we have to save it in a known folder, for example:

```
~/Documents/Programming/examples
```

If that folder does not exist, we create it by typing on the prompt:

```
mkdir ~/Documents/Programming/examples
```

We go to the folder using the command `cd`.

How Do We Use It

Once the program is done, How do we use it?.
First of all, we have to save it in a known folder, for example:

```
~/Documents/Programming/examples
```

If that folder does not exist, we create it by typing on the prompt:

```
mkdir ~/Documents/Programming/examples
```

We go to the folder using the command `cd`.
It's important to save the program with the extension `.cpp`, for example, let's call the program:

```
example1.cpp
```

Compiling

As we see before, C++ is a compiled language, so we must compile every program before running it, there's a lot of programs that do it, but the one we're going to use is called g++, once we're on the folder where is our code saved with a .cpp extension, we use it as follows:

```
g++ example1.cpp
```

that action creates an executable, by default it's called a.out, and we run it by typing:

```
./a.out
```

Variables

But, write out `Hello World` isn't enough, we want some numerical process, for example; so, how do we do it?

Definition 1

A **variable** *is a space in memory reserved to store a value*

It can be a little different of what we understand of *variable*[1]. But, if a variable is an space in memory, how do we access to the exact space we want?, basically, we *name* every section, or every variable, that's an *Identifier*

[1] It's important to keep in mind that any variable we define, is a section in memory that is reserved and no program can use it, unless we release it.

For example, if we want to sum two integers

For example, if we want to sum two integers

```
//My second program
#include<iostream>
int main(){
int a=1;
int b=2;
std::cout<<a+b;
return 0;
}
```

For example, if we want to sum two integers

```
//My second program
#include<iostream>
int main(){
int a=1;
int b=2;
std::cout<<a+b;
return 0;
}
```

But, what does `int` mean?, that's the type of variable, it's an integer.

For example, if we want to sum two integers

```
//My second program
#include<iostream>
int main(){
int a=1;
int b=2;
std::cout<<a+b;
return 0;
}
```

But, what does `int` mean?, that's the type of variable, it's an integer.

What if we do not write an integer value in an integer variable?, Try it.

Types

There's a lot of data types, but, some of the more used are shown in the next table:

Group	Name	Size
Character	`char`	8 bits
	`char32`	At least 16 bits.
	`char54`	At least 32 bits.
Integer[2]	`short int`	16 bits
	`int`	At least 16 bits.
	`long int`	At least 32 bits.
Float-point	`float`	32 bits
	`double`	64 bits
	`long double`	128 bits

Cuadro 1: Data types

[2] It can be *signed* or **unsigned**

So, there is a limit for the values we save in the variable, that's why we have to choose wisely which one are we going to use. if we represent in numbers, the biggest number we can save depending of the size of the variable are shown in the next table

Size	Unique representable values	Notes
8 bit	256	2^8
16 bit	65536	2^{16}
32 bit	4294967296	2^{32}(4 billion)
64 bit	18446744073709551616	2^{64} (18 billion billion)

Cuadro 2: Biggest number that can be saved depending the size

Constants

Sometimes it isn't enough having some values that change *variables*, we need fixed values too.

Constants

Sometimes it isn't enough having some values that change *variables*, we need fixed values too.
First we have the most obvious are called literals, and it's assigning a value to a variable

Constants

Sometimes it isn't enough having some values that change *variables*, we need fixed values too.
First we have the most obvious are called literals, and it's assigning a value to a variable

```
int a=5;
```

so `5` is saved on `a`.

Constants

Sometimes it isn't enough having some values that change *variables*, we need fixed values too.
First we have the most obvious are called literals, and it's assigning a value to a variable

```
int a=5;
```

so `5` is saved on `a`.
We can use floating point in some ways

Constants

Sometimes it isn't enough having some values that change *variables*, we need fixed values too.
First we have the most obvious are called literals, and it's assigning a value to a variable

```
int a=5;
```

so `5` is saved on `a`.
We can use floating point in some ways

`3.14159`	3,14159
`6.02e23`	$6{,}02 \times 10^{23}$
`1.6e-19`	$1{,}6 \times 10^{-19}$
`3.0`	3,0

But, What if we want that the *constant* do not change?, so we can declare it in a different way.

```
//Declaring constants
#include <iostream>
const double pi = 3.14159;
const char nw = '\n';
int main (){
double r=2.0
double p;
p=2*pi*r;
std::cout<<p<<nw;
return 0;}
```

Another way to name a constant value is using #define

```
//Declaring constants
#include <iostream>
#define pi  3.14159
#define nw '\n'
int main (){
double r=2.0
double p;
p=2*pi*r;
std::cout<<p<<nw;
return 0;}
```

Operators

We need to operate the constants and variables we just defined, but how do we do it?, first, one that we have used before, is the assignment operator (=)

```
x=5;
```

But it works not just for values, but for other variables

```
y=x;
```

Then, the next operators are the arithmetic operators (+, -, *, /, %)

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo

Cuadro 3: Arithmetic Operators

we also have another type of assignment operators, when the variable already has a value and we want to operate them with constants or variables, (+=,-=,*=,/=, %=), for example:

Expression	Equivalent
`y += x`	`y = y + x`
`a *= (b + 3)`	`a = a * (b + 3)`

Cuadro 4: More Arithmetic Operators

Increment and Decrement

```
#include<iostream>
int main(){
int a,b;
a=5;
b=a;
std::cout<<b<<'\t'<<a<<std::endl;
b++;
a--;
std::cout<<b<<'\t'<<a<<std::endl;
return 0;
}
```

Comparison Operators

We need sometimes compare the values got in any variable, so in C++ we have the next comparison operators

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Cuadro 5: Comparison Operators

The result of that operator is a boolean, or it's true, or false.

Logical Operators

We have three logical operators

Operator	Description
!	Not - Negation
&&	And
||	Or

Cuadro 6: Logical Operators

These are going to be very important when we study loops and conditional functions

Conditional Operators

A conditional operator, is a combination of the previous operators

```
#include<iostream>
int main(){
int a,b;
a=5;
b= (a==5) ? 4 : 3;
std::cout<<b<<std::endl;
a=4;
b= (a==5) ? 4 : 3;
std::cout<<b<<std::endl;
return 0;
}
```

Initialization of Variables

If we name a variable, at first, it will reserve a space in memory, but we must save in this space the number we want, we it mustn't be calculated, we can initialize the value in some ways, for example:

```
// initialization of variables
#include <iostream>
int main (){
  int a=5;    // initial value: 5
  int b(3);   // initial value: 3
  int c{2};   // initial value: 2
  int result;// initial value undetermined
  a = a + b;
  result = a - c;
  std::cout << result <<std::endl;
  return 0;}
```

References

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

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

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