



Introducción a C++

Diego Useche - dh.useche@uniandes.edu.co

Metodos Computacionales II

Physics Department, Universidad de los Andes, Bogotá



Initial program

- # to include libraries

```
#include <iostream>
using namespace std;

int main()
{

    return 0;
}
```

Namespace and hello world

- cout without namespace would be std::cout

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!";
    return 0;
}
```

Comment code

- multiline comment `/* */`
- single line comment `//`

```
/*We are going to  
   print "Hello world!" */  
  
   cout << "Hello world!"; // prints Hello world!
```

Declaration of variables

```
#include <iostream>
using namespace std;

int main()
{
    int myVariable = 10; ←
    cout << myVariable;
    return 0;
}
```

Data types

Table 2-6 Integer Data Types

Data Type	Typical Size	Typical Range
short int	2 bytes	-32,768 to +32,767
unsigned short int	2 bytes	0 to +65,535
int	4 bytes	-2,147,483,648 to +2,147,483,647
unsigned int	4 bytes	0 to 4,294,967,295
long int	4 bytes	-2,147,483,648 to +2,147,483,647
unsigned long int	4 bytes	0 to 4,294,967,295
long long int	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long int	8 bytes	0 to 18,446,744,073,709,551,615

Table 2-8 Floating Point Data Types on PCs

Data Type	Key Word	Description
Single precision	float	4 bytes. Numbers between $\pm 3.4\text{E-}38$ and $\pm 3.4\text{E}38$
Double precision	double	8 bytes. Numbers between $\pm 1.7\text{E-}308$ and $\pm 1.7\text{E}308$
Long double precision	long double*	8 bytes. Numbers between $\pm 1.7\text{E-}308$ and $\pm 1.7\text{E}308$

Image taken from <https://www.cs.mtsu.edu/~xyang/2170/datatypes.html>

Standard input and standard output (input() and print())

```
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cout << "Enter a number \n";
    cin >> a;
    cout << "Enter another number \n";
    cin >> b;

    return 0;
}
```

Prefix and postfix

```
x = 5;  
y = ++x;  
// x is 6, y is 6
```


If statement

- put curly brackets { }

```
int a = 55;  
int b = 33;  
if (a > b) {  
    cout << "a is greater than b";  
}
```

While loop

```
int num = 1;
while (num < 6) {
    cout << "Number: " << num << endl;
    num = num + 1;
}
```

← While

```
int a = 0;
do {
    cout << a << endl;
    a++;
} while(a < 5);
```

← Do while

For loop

```
int myArr[5];  
  
for(int x=0; x<5; x++) {  
    myArr[x] = 42;  
}
```

Switch statement to analyze cases

```
int age = 25;
switch (age) {
    case 16:
        cout << "Too young";
        break;
    case 42:
        cout << "Adult";
        break;
    case 70:
        cout << "Senior";
        break;
    default:
        cout << "This is the default case";
}
```

And / Or

```
int age = 20;
if (age > 16 && age < 60) {
    cout << "Accepted!" << endl;
}
```

And: &&

```
int age = 16;
int score = 90;
if (age > 20 || score > 50) {
    cout << "Accepted!" << endl;
}
```

Or: ||

Arrays

```
int arr[] = {11, 35, 62, 555, 989};  
int sum = 0;  
  
for (int x = 0; x < 5; x++) {  
    sum += arr[x];  
}  
  
cout << sum << endl;
```

Multidimensional arrays

```
int x[2][3] = {  
    {2, 3, 4}, // 1st row  
    {8, 9, 10} // 2nd row  
};
```

Functions

must indicate the return type, if not return use "void"



```
int addNumbers(int x, int y) {  
    int result = x + y;  
    return result;  
}  
  
int main() {  
    cout << addNumbers(50, 25);  
}
```


Functions

must indicate the type of parameters



```
int addNumbers(int x, int y) {  
    int result = x + y;  
    return result;  
}  
  
int main() {  
    cout << addNumbers(50, 25);  
}
```

Default parameters

```
int volume(int l=1, int w=1, int h=1) {  
    return l*w*h;  
}  
  
int main() {  
    cout << volume() << endl;  
    cout << volume(5) << endl;  
    cout << volume(2, 3) << endl;  
    cout << volume(3, 7, 6) << endl;  
}
```

Default parameters can be set on python as well.

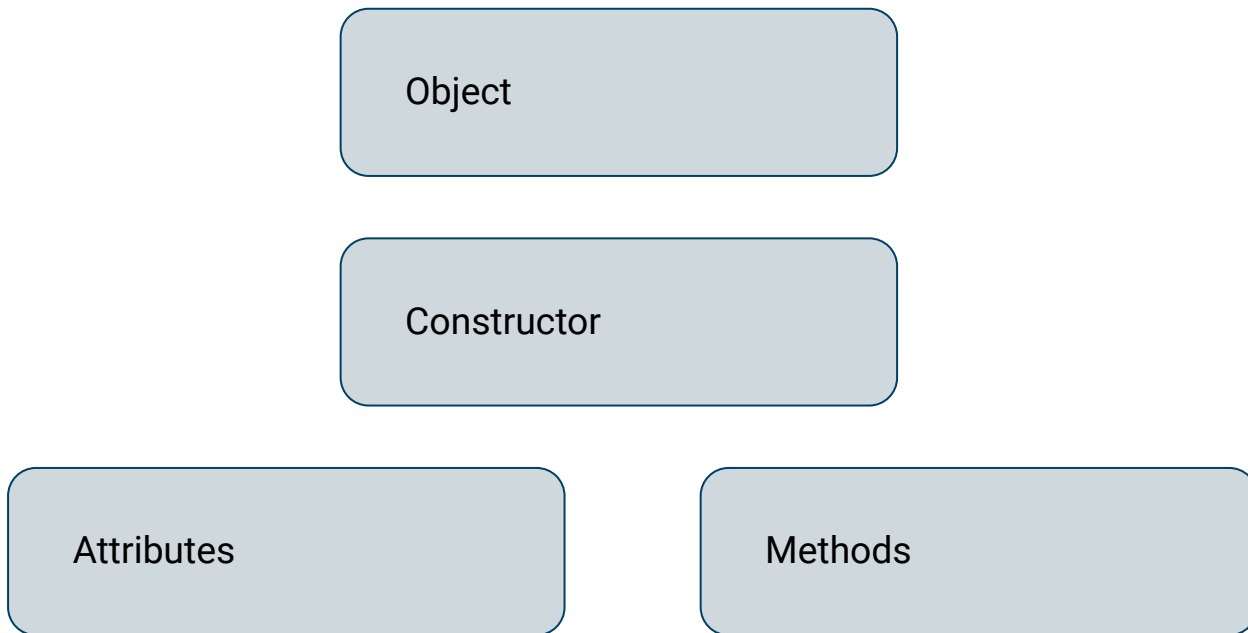
OOP paradigm

- C++ al igual que python permite programar orientado a funciones y orientado a objetos.




OOP Paradigm

- General structure



Classes

Attributes



```
class myClass {  
    private:  
        string name;  
    public:  
        myClass() {  
            cout << "Hey";  
        }  
        void setName(string x) {  
            name = x;  
        }  
        string getName() {  
            return name;  
        }  
};  
  
int main() {  
    myClass myObj;  
  
    return 0;  
}
```

Classes

— Constructor

```
class myClass {  
    private:  
        string name;  
    public:  
        myClass() {  
            cout << "Hey";  
        }  
        void setName(string x) {  
            name = x;  
        }  
        string getName() {  
            return name;  
        }  
};  
  
int main() {  
    myClass myObj;  
  
    return 0;  
}
```

Classes

Functions

```
class myClass {  
    private:  
        string name;  
    public:  
        myClass() {  
            cout << "Hey";  
        }  
        void setName(string x) {  
            name = x;  
        }  
        string getName() {  
            return name;  
        }  
};  
  
int main() {  
    myClass myObj;  
  
    return 0;  
}
```

Classes

```
class myClass {  
    private:  
        string name;  
    public:  
        myClass() {  
            cout << "Hey";  
        }  
        void setName(string x) {  
            name = x;  
        }  
        string getName() {  
            return name;  
        }  
};  
  
int main() {  
    myClass myObj;  
  
    return 0;  
}
```

Create the object in the main



Example

You have to create a class, named Student, representing the student's details, as mentioned above, and store the data of a student. Create setter and getter functions for each element; that is, the class should at least have following functions:

- `get_age, set_age`
- `get_first_name, set_first_name`
- `get_last_name, set_last_name`
- `get_standard, set_standard`

Also, you have to create another method `to_string()` which returns the string consisting of the above elements, separated by a comma(,). You can refer to stringstream for this.

Pointers: pointer address

- `&score`, indicates the address in memory of variable `score`.

```
#include <iostream>
using namespace std;

int main()
{
    int score = 5;
    cout << &score << endl;

    return 0;
}
```

Pointers: creating a pointer variable

- `int* scorePtr`, indicates that `scorePtr` is a pointer of an `int`

```
#include <iostream>
using namespace std;

int main()
{
    int score = 5;
    int* scorePtr;
    scorePtr = &score;

    cout << scorePtr << endl;

    return 0;
}
```

Pass by reference vs pass by value

By value

```
#include <iostream>
using namespace std;

void myFunc(int x) {
    x = 100;
}

int main() {
    int var = 20;
    myFunc(var);
    cout << var;
}
```

By reference

```
#include <iostream>
using namespace std;

void myFunc(int* x) {
    *x = 100;
}

int main() {
    int var = 20;
    myFunc(&var);
    cout << var;
}
```

Example

Returns

- The function is declared with a void return type, so there is no value to return. Modify the values in memory so that a contains their sum and b contains their absolute difference.
- $a' = a + b$
- $b' = |a - b|$

Input Format

Input will contain two integers, a and b , separated by a newline.

Sample Input

```
4
5
```

Sample Output

```
9
1
```

References

<https://www.cs.mtsu.edu/~xyang/2170/datatypes.html>

<https://www.sololearn.com/>

<https://www.hackerrank.com/>