Introduction to C/C++

Giovanni Bussi
PhD in Physics and Chemistry of Biological Systems
SISSA, Trieste, Italy

C++

Compiled language

Superset of C (i.e. a C program can be compiled with C++, but not the opposite). Backward compatible.

Huge syntax specification (years to learn everything!)

Several standards. Most commonly used now is C++11.

If you want to go deeper, I recommend starting with a book that uses C++11 (many older things are just not needed)

Several implementations:
• gcc (g++)
• clang (clang++)
• ...

Simple C++ program

```
int func(int a); // declaration
// main function
// takes no arg (in this case)
// and returns int (0=success)
int main() {
  return func(a);
}
// definition, also in another file:
int func(int a) {
  return a*2;
}
```

Functions should be "declared"
The language comes with a standard library of functions
(that should be explicitly included, see next slide)

Simple C++ program

```cpp
// preprocessor directive:
#include <iostream>

// main function
int main() {
// standard library functions are in
// "std::" namespace
  std::cout<<"ciao\n";
  return 0;
}
```

```
g++ main.cpp -o main.x
./main.x
```

Some reference

https://github.com/asartori86/advanced_programming-2018-19

https://www.learncpp.com/

google.com

No easy way... try to learn the basic and then edit existing codes!

Variables

```
// variables (as functions) should be declared!
#include <iostream>
int main() {
  const int a=3; // a is int
  const auto b=a+1; // b is same type as a+1
  std::cout<<"this is a:”<<a<<"\n"
  std::cout<<"this is b:”<<a<<"\n";
  return 0;
}


// pay attention to uninitialized variables!
int a;
// a has no value yet
a=5;
// now a has a value
```

Types

```
// integer types
int a; unsigned b;
a=3; b=a/2; // b=1!!!

// floating point numbers
double a; double b;
a=3; b=a/2; // b=1.5

// char stores one single letter!
// to use strings check std::string on google
char a='c'; // single quote for character
int b=3+a; // converted to int!
std::cout<<a<<" "<<b<<"\n";

// use const when you know a variable will not change!
```

Input/output

```
#include <iostream>

int main() {
  int a;
  std::cin>>a;
  const int b=a*10;
  std::cout<<b<<"\n";
}
```

```
#include <iostream>

int main() {
  int a;
  while(std::cin>>a) {
    const int b=a*10;
    std::cout<<b<<"\n";
  }
}
```

Loops and conditionals

```
#include <iostream>
#include <cmath>

int main() {
  for(unsigned i=0;i<1000;i++) {
    const auto x=i*0.01*3.14;
    const auto y=std::sin(x);
    if(y>0.5) std::cout<<x<<" "<<y<<"\n";
  }
}
```

```
#include <iostream>
#include <string>
int main() {
  bool cont=true;
  while(cont) {
    std::string in;
    if(!(std::cin>>in)) break; // parenthesis!
    if(in=="stop") cont=false;
  }
}
```

Functions

```
#include <iostream>
#include <cmath>
// note: these functions can be inlined!
void square(int & p) { // pass by reference
  p=p*p;
}
int square2(const int p) { // pass by value
  return p*p;
}
// overload (same name, different function)
double square2(const double p) {
 return p*p;
}
int main() {
  int number;
  while(std::cin>>number) std::cout<<square2(number)<<"\n";
  while(std::cin>>number) {
    square2(number);
    std::cout<<number<<"\n";
  }
}
```

const variables

```
void square(const int & p) { // pass by reference
  p=p*p; // error!!
}

int main() {
  const double a=3.0;
  a+=3; // error!
}
```

Good habit: always tell the compiler what you know, it will help you in finding errors!

Classes

```
#include <iostream>
#include <cmath>

class MyVector {
public:
  double x;
  double y;
  double z;
  double modulo() const { // const method
    return x*x+y*y+z*z;
  }
};

int main() {
  MyVector v;
  v.x=2.0; v.y=3.0; v.z=4.0;
  std::cout<< v.modulo()<<"\n";
}
```

A more advanced view

```
// preprocessor directive:
#include <iostream>

// main function
int main() {
// standard library functions are in
// "std::" namespace
  std::cout<<"ciao\n";
  return 0;
}
```

std::cout is an object of class std::ostream (an "output stream")
operator "<<" is "overloaded" so as to print different types
with different formats

Containers

```
// if you come from C, do not use pointers

#include <vector>
std::vector<double> v;
for(unsigned i=0;i<10;i++) v.push_back(2*i);

for(unsigned i=0;i<10;i++)
    std::cout<<v[i]<<"\n";

// initialize with size 10 and all elements 0.5
std::vector<double> vec(10,0.5);
```

std::vector<T> is a template class

For SimpleMD (tomorrow afternoon)

```
std::vector<Vector> positions;

// std::vector of Vector
// Vector is a custom class with 3 components
// ([0], [1], and [2])


positions[10][2];
// is the third (z) coordinate of eleventh atom
```

Standard library

Standard library:

https://en.cppreference.com/w/
http://www.cplusplus.com/reference/

```
#include <vector>
#include <string>
#include <iostream>
#include <map>
int main(){

  std::string s="this is";
  s=s+" a string";
  std::cout<<s<<"\n";

  std::vector<std::string> vec;
  vec.push_back("first");
  vec.push_back("second");

  std::map<int,double> map;
  map[3]=5.0;
  return 0;
}
```

Linear algebra: armadillo

http://arma.sourceforge.net/

```
#include <iostream>
#include <armadillo>

using namespace std;
using namespace arma;

int main()
  {
  mat A = randu<mat>(4,5);
  mat B = randu<mat>(4,5);

  cout << A*B.t() << endl;

  return 0;
  }
```

See also Eigen: http://eigen.tuxfamily.org

C++ vs AWK

```
# test file
awk 'BEGIN{for(i=0;i<1000000;i++)print(i/10)}' > oo
```

```
#include <iostream>
int main() {
  double sum=0.0;
  double num;
  while(std::cin>>num) sum+=num;
  std::cout<<sum<<"\n";
}
g++ -O3 sum.cpp -o sum.x
```

```
awk '{sum+=$1}END{print sum}'
```

```
time ./sum.x < oo          4.6s
time bash sum.awk < oo     1.0s !!
```

C++ vs AWK

```
#include <iostream>
#include <vector>
int main() {
  std::vector<double> vec;
  double num;
  while(std::cin>>num) vec.push_back(num);
  double sum; for(const auto n : vec) sum+=n;
  std::cout<<sum<<"\n";
  return 0;
}

awk '{n[NR]=$1}END{for(i in n)sum+=n[i]; print sum}'
```

```
time ./a.out < oo          4.8s
time bash sum2.awk < oo     2.8s
```

C++ vs AWK

```
awk 'BEGIN{for(i=0;i<1000;i++)print(i/10)}' > pp

#include <iostream>
#include <vector>
#include <cmath>
int main() {
  std::vector<double> vec;
  double num;
  while(std::cin>>num) vec.push_back(num);
  double sum;
  for(const auto n : vec) for(const auto m : vec) sum+=std::cos(n-m);
  std::cout<<sum<<"\n";
  return 0;
}

awk '{n[NR]=$1}END{
  for(i in n) for(j in n) sum+=cos(n[i]-n[j]); print sum}'
```

```
time ./sum3.x < oo         0.05s
time bash sum3.awk < oo    0.7s
```

Summary

Very complex language

Very powerful (can do anything)

No need to learn everything, take your time and learn looking at examples.

Try to start with C++11 (although most books still teach you first old-fashioned syntax unfortunately...)