C++ Tutorial
Let’s get to coding

W. Deconinck\textsuperscript{1}   V. Gray\textsuperscript{2}

\textsuperscript{1}Assistant Professor  
Department of Physics  
College of William & Mary

\textsuperscript{2}Graduate Student  
Department of Physics  
College of William & Mary

C++ Tutorial, May 29, 2014
Outline

C++ An Overview
  Data Types, Variables, etc
  Development Environments

C++ 101
  Set Up
Things C++ can do

Data Types

- `bool`
- `int`
- `char`
- `float`
- `double`
- `std::string`

Declaration & Initialization

```cpp
int a;
a = 1;
or
int a = 1;
int b(a);
```

Type Casting

```cpp
unsigned int b = 1;
int a = dynamic_cast<int>(b);
double a = static_cast<double>(b);
```

Scope

Variable can only be used in their scope: the smallest block of braces ({ /*...*/ }) that surrounds them.
Arrays

// array of 3 integers with initialization
int a[3] = {0, 1, 2};

// without explicit size (as it is redundant)
int b[] = {0, 2};

// 2 x 2 matrix without initialization
int M[2][2];

cout << a[0] << endl;  // first element in a

cout << M[1][1] << endl; // second diagonal element in M

Indices

Array indices in C++ start counting at 0: a[0], a[1],...
Vectors

```cpp
#include <vector>
vector<int> a(3); // vector of integers with size 3
// Unfortunately, the initializer lists {0,1,2}
// does not work with these vectors.
cout << a.size() << endl; // print size of a
vector<vector<int> > M; // vector of vectors of integers
M.resize(2); // resize M to have two rows
M[0].resize(2); // two elements in first row
M[1].resize(3); // three elements in second row

// print size of first row of M
cout << M[0].size() << endl;
// print size of second row of M
cout << M[1].size() << endl;
```
Things C++ can do

Control structure

```c
if (condition)
{
    /* do something sometimes*/
} else
{
    /* do something the other times */
}
```
Things C++ can do

Control structure

```cpp
for (int i = 0; i < 10; i++)
{
    /* do something 10 times */
}
```
Things C++ can do

Control structure

```c
switch (a)
{
    case 1:
        /* do something */
        break;

    case 2:
        /* do something different */
        break;

    default:
        /* do something in all other cases */
        break;
}
```
Things C++ can do

Functions

//functions are defined like
type name (parameter1, parameter2,...)
{
   /* statements */
}

//they are called like
type name (parameter1, parameter2,...);

Signature

Function can (but don't have to) have a **signature** before they are defined. The signature for a general function (above) is

```cpp
type name (parameter1, parameter2,...);
```

This signature will typically be placed in the header file, as it provides the **interface** to using the functions that are defined in the source file.
Call by Value

When a function such as `int addition (int a, int b)` is called, the two arguments are copied and the function operates on the copy only.

**The original values of the arguments, a and b are not changed.**

Call by Reference

It is also possible to call the function as `int addition (int & a, int & b)`. The additional `&` characters indicate that the values for a and b are not copies but are the original values.

**The variables a and b are now references to the original variables, and any changes to a and b will affect the original values.**
Things C++ can do

Pointers

//functions are defined like
type functionname (parameter1, parameter2,...) {
  /* statements */
  return something;
}

//they are called like
functionname (parameter1, parameter2,...);
Things C++ can do

Classes
They are extensions of the different basic data types

    //class of a rectangle with sides x and y
    class CRectangle
    {
        int x, y;

        public:
            void set_values (int, int);
            int area (void);
    };

This is just the abstract object that represents a rectangle.
Things C++ can do

Classes
They are extensions of the different basic data types

```cpp
//class of a rectangle with sides x and y
class CRectangle
{
   int x, y;

   public:
      void set_values (int, int);
      int area (void);
};
```

There are **private** and **public** blocks of class data members (or fields) and functions (or methods).
The **private** members and functions can only be accessed by the objects of the class itself. The **public** members and functions can be accessed by everyone.
Classes
To create a variable of the type CRectangle you can use the same syntax as for the declaration of basic data types.

CRectangle rect; // the variable rect of type CRectangle

On the object rect you can now call all public functions and access public data members:

rect.set_values (3,4); // set the values on object rect
myarea = rect.area(); // get the area
Things C++ can do

Classes

Notice that we have not yet specified the implementation for the functions `void set_values(int,int)` and `int area (void)`. For example

```cpp
void CRectangle::set_values (int a, int b)
{
    x = a;
    y = b;
}
int CRectangle::area (void)
{
    return (x*y);
}
```

The notation `::` indicates that the functions `set_values` and `area` are inside the class `CRectangle`. 
Things C++ can do

**Constructor & Destructor**
We have to tell C++ how to initialize objects of classes that we define, and how to free the memory again. For example:

```cpp
CRectangle::CRectangle ()
{
    a = 0;
    b = 0;
}

CRectangle::~CRectangle ()
{
    std::cout << "he's dead, Jim" << std::endl;
}
```

Destructors are often used to release explicitly created pointer objects (with `delete`).
Integrated development environments

An integrated development environment (IDE) the compiler, editor and debugger are integrated in the same software program. This typically makes for a more productive environment when you can spend the time in setting it up to work according to your needs.

- **Eclipse** (with C/C++ Development Tooling), http://www.eclipse.org/cdt/ This is an integrated development environment originally developed for Java programming but excellent support for C/C++ is available as well. It has syntax highlighting, auto-completion and even auto-generation of code, and it includes a debugger.
Integrated development environments

An integrated development environment (IDE) the compiler, editor and debugger are integrated in the same software program. This typically makes for a more productive environment when you can spend the time in setting it up to work according to your needs.

- **Kdevelop**, [http://www.kdevelop.org/](http://www.kdevelop.org/) This integrated development environment focuses on C/C++ development, in particular for open source projects (e.g. KDE). It includes a similar set of features as Eclipse above.
An integrated development environment (IDE) the compiler, editor and debugger are integrated in the same software program. This typically makes for a more productive environment when you can spend the time in setting it up to work according to your needs.

- **Visual C++ Express** (part of Visual Studio Express), [http://www.microsoft.com/visualstudio/eng/team-foundation-service](http://www.microsoft.com/visualstudio/eng/team-foundation-service) This is an integrated development environment for Microsoft Windows.
C++ 101
Set Up

Open a Terminal Window

- Windows users: this will be through a program like Putty or SSH Secure Shell
- Mac and Linux users: find the Terminal program

Login

```bash
$ ssh -X RootTutorial@sporades.physics.wm.edu
```

Password

`Rootisfun2013`
See what is there
List the contents the home directory, via `ls` command

```
[RootTutorial@sporades ~]$ ls
cplusplus_tutorial_files
cplusplus_tutorial_link.txt
cplusplus_tutorial_files.tar
root_tutorial_files
root_tutorial_files.tar
users
```
Create your directory

Create the your own directory in the users directory

[RootTutorial@sporades ~]$ cd users
[RootTutorial@sporades users]$ ls
adorasmith dasalmon Henderson ...
an smith eahenderson hnguyen ...
cmkaramitsos Eahenderson itgordon ...
crhaufe eemikh janderson ...
[RootTutorial@sporades users]$ mkdir <yourWMusername>
[RootTutorial@sporades users]

This directory is only for you. Please don’t delete anything in someone elses directory
C++ 101
Set Up

Copy the C++ tutorial files

[RootTutorial@sporades users]$ cd <yourWMusername>
[RootTutorial@sporades <yourWMusername>]$ mkdir cplusplus_tutorial_files
[RootTutorial@sporades <yourWMusername>]$ cp ˜/cplusplus_tutorial_files/*
  ˜/users/<yourWMusername>/cplusplus_tutorial_files/*
Double Check
Make sure the files are there

[RootTutorial@sporades  <yourWMusername>]$
   cd cplusplus_tutorial_files/
[RootTutorial@sporades  <yourWMusername>]$ ls
addition.cpp  addition.h  arrays  arrays.cpp
CRectangle.cpp  datatypes.cpp  helloworld.cpp
helloworld.h  pointers.cpp  subtraction.cpp
subtraction.h  switch  switch.cpp
Ready, Set, Go!

text editors

- **nano**: a bare-bones text-based editor with basic syntax highlighting (after some effort). Exit by typing Ctrl-x
- **vi**: a text-based editor with a fairly steep learning curve. Exit by typing :q
- **emacs**: a graphical or text-based editor with a fairly steep learning curve. Exit by typing Ctrl-x, Ctrl-c
- **nedit**: a basic graphical editor with syntax highlighting (after some effort). Nedit runs smoothly over ssh connections.
- **kate**: a graphical editor with syntax highlighting. Kate supports opening files over ssh connections using the fish://server/path/to/file syntax.
Need help?

- C++: www.cplusplus.com
- Google: www.google.com