Basic **Coding**
Tutorial & Guideline

Find This Document At:
www.nevis.columbia.edu/~kazuhiro/Summer2015_CPPTutorial_00.pdf

Kazuhiro Terao @ Nevis, Columbia

Introduction to Programming
&
Overview of What We Learn

What is “Programming?”

- Also called “coding”
- It is to provide a set of instructions for your computer to perform
- Three steps to achieve in programming
  - “Write”, “Compile”, and “Execute” your program

How to “Write” a Program

- Write in a “programming language”
  - C/C++ is one of the most popular & basic language
- Save it in a text file
  - Pick a text file editor of your choice: emacs, vim, nano, etc. etc.
  - This matters to your coding efficiency! Learn your editor VERY WELL!

How to “Compile” a Program

- What does it mean to “compile”?
  - What you wrote is a text file
  - Computer doesn’t speak English (it is made by aliens)
  - Compile = convert your text file (English) into a computer’s language (byte)
- Use a “Compiler” to compile your program
  - Made by those awesome dudes who speak alien’s language
  - It translates various programming language into byte code.
  - C++ compiler: “g++” and “clang++” as most typical choice
- What is a “compiler” output? … roughly 2 types
  - “Executable” … a program that “runs” or “execute tasks”
  - “Library” … a byte-code description of toolkits, like “functions”

Last Topic: Compiled vs. Interpreted Language

- Compilation comes with pro and cons
  - Pros:
    - Compiler checks your program and finds mistakes for you
    - Compiled byte-code gets executed very fast
  - Cons:
    - You have to learn about a compiler apart from the language
    - Everything requires a compilation: not easy to try a simple task

- Interpreted language
  - … does not require an explicit compilation step
  - Your program is “compiled” line-by-line when you execute them
  - Pros
    - You don’t have to compile! (i.e. compilation is a part of execution)
    - Very easy to try a simple thing quickly
  - Cons
    - No grammar check: you will find a problem in your code @ run-time
    - Line-by-line compilation ends up slow execution speed

So… What Are We Going to Cover?

You tell me what you want to cover!

- Basics of C++
  - "Hello World" program … learn how to compile a simple program
  - C++ class & functions
  - C++ class inheritance & templates
- Basics of Python
  - "Hello World" program
  - Python class & functions
  - "Everything-is-object"
- Advanced topics
  - C++ std libraries (STL containers, std algorithms, etc)
  - Scientific python libraries (sci/numpy, matplotlib, pandas)
  - "C++ in Python" (accessing C++ class/functions from Python)

What I Do NOT Plan To Cover

- How to use a text editor
- How to use “terminal” (i.e. shell language)
- How to use ROOT (this, we can negotiate…)
  - Very well written tutorial here

C++ Introduction

~ Link: “if programming languages were vehicle” ~

C++ introduction

- Checkout the example from my web space

  www2.lns.mit.edu/~kazuhiro/Summer2015_CPPIntro.tar.gz

- If you are on uboonegpvm machine:

  > source /grid/fermiapp/products/uboone/setup_uboone.sh

  > setup gcc v4_9_2

- Find 7 sub-directories under playground/Introduction
  - simplest … simplest "main" program
  - example_00 ... simple "hello world" program
  - example_01 ... "hello world" using a function
  - example_02 ... "hello world" using a class
  - example_03 ... separation of class/function from driver code
  - example_04 ... class inheritance
  - example_05 ... template
- Recommended C++ resource ... a lot can be found online
  - cplusplus.com
  - cppreference.com

C++ introduction ... simplest

- Compile simplest.cc

```
int main()
{
  return 1;
}
```

If you are on Unix/Linux with LLVM (OSX, Ubuntu 14)

```
kazuhiro$ clang++ simplest.cc -o simplest
```

Other Unix/Linux (Ubuntu 12, SL6)

```
kazuhiro$ g++ simplest.cc -o simplest
```

Execute simplest

```
bash-3.2$ ./simplest
bash-3.2$
bash-3.2$ echo $?
1
```

Compilation methods same for example_0X

C++ introduction ... example_00

- Compile example_00.cc

```
#include <iostream>

int main() {

  std::cout << "hello world... I think 1 + 1 = " << 1+1 << std::endl;

  return 0;
}
```

If you are on Unix/Linux with LLVM (OSX, Ubuntu 14)

```
kazuhiro$ clang++ example_00.cc -o example_00.exe
```

Other Unix/Linux (Ubuntu 12, SL6)

```
kazuhiro$ g++ example_00.cc -o example_00.exe
```

Execute example_00.exe

```
kazuhiro$ ./example_00.exe
hello world... I think 1 + 1 = 2
```

Compilation methods same for example_0X

C++ introduction ... example_01

- example_01.cc

```cpp
#include <iostream>

// Define a function
void HelloWorldFunc()
{ std::cout << "hello world from function!" << std::endl; }

// Use that function
int main() {

  HelloWorldFunc();

  return 0;
}
```

- Introduced C++ function, “HelloWorldFunc”
- Nope, nothing more than that. Moving on...

C++ introduction ... example_02

- example_02.cc

```cpp
#include <iostream>

// Define a class
class HelloWorldClass {

public:
  /// Default constructor
  HelloWorldClass(){}
  /// Default destructor
  ~HelloWorldClass(){}
  /// Greeting function
  void Hello() const
  { std::cout << "Hello world from class-func!" << std::endl; }
};

// Use that class
int main() {

  HelloWorldClass obj;

  obj.Hello();

  return 0;
}
```

- Introduced C++ class, “HelloWorldClass”
  - Read (a lot) more about classes here

C++ introduction

- Why those 3 trivial (and boring) examples?
  - remind you about a simple C++ executable
  - make sure you know about class/functions
  - write code with class/functions without a dedicated build system

- Introduction to reusable code structure
  - “main” function is a driver function to be executed
  - Other functions/classes can be re-used for various “main” functions
  - To achieve this, we use a pre-processor command “#include”

C++ introduction ... example_03

```h
#include <iostream>

// Define a class
class HelloWorldClass {

public:
  /// Default constructor
  HelloWorldClass(){}
  /// Default destructor
  virtual ~HelloWorldClass(){}
  /// Greeting function
  void Hello() const
  { std::cout << "Hello world from class-func!" << std::endl; }
};

// Define a function
void HelloWorldFunc()
{ std::cout << "hello world from function!" << std::endl; }
```

example_03.h

```cc
#include "example_03.h"

int main()
{
  HelloWorldClass obj;

  obj.Hello();

  HelloWorldFunc();

  return 0;
}
```

example_03.cc

- Defined class/functions in “example_03.h”
  - now various *.cc can call #include example_03.h and share code
- Nope, nothing more than that. Moving on...

C++ introduction ... example_04

- Class inheritance
  - Children classes inherit various features from parent class. Read here.
  - Greatly helps re-usable code design

```
class Polygon{
public:
  Polygon(){}

  virtual int area()
  { return -1; }

  void SetParams(int width, int height)
  { w = width; h = height; }

protected:
      int w, h;
};

class Rectangle : public Polygon{
public:
  Rectangle(){}
  virtual int area() { return w * h; }
};

class Triangle : public Polygon{
public:
  Triangle(){}
  virtual int area() { return w * h / 2.; }
};
```

polygon.h

Base class: Polygon
- defines “width” and “height”
- defines a setter function
- defines useless “area” function

Child class: Rectangle
- overrides “area” function

Child class: Triangle
- overrides “area” function

C++ introduction ... example_04

```cpp
#include <iostream>
#include "polygon.h"

int main() {

  Polygon   obj1;
  Rectangle obj2;
  Triangle  obj3;

  obj1.SetParams(2,2);
  obj2.SetParams(2,2);
  obj3.SetParams(2,2);

  std::cout
    << std::endl
    << "Area of (w,h) = (2,2) Polygon   : " << obj1.area() << std::endl
    << "Area of (w,h) = (2,2) Rectangle : " << obj2.area() << std::endl
    << "Area of (w,h) = (2,2) Triangle  : " << obj3.area() << std::endl
    << std::endl;

  return 0;
}
```

example_04.cc

```
Area of (w,h) = (2,2) Polygon   : -1
Area of (w,h) = (2,2) Rectangle : 4
Area of (w,h) = (2,2) Triangle  : 2
```

Output of executing example_04.exe

C++ introduction ... example_05

- Class/Function template
  - You write abstract description without specifying the subject type. Read here.
  - Greatly helps re-usable code design

```
template <class T>
class Polygon{
public:
  Polygon(){}

  virtual T area()
  { return -1; }

  void SetParams(T width, T height)
  { w = width; h = height; }

protected:
  T w, h;
};

template <class T>
class Rectangle : public Polygon<T>{
public:
  Rectangle(){}
  virtual T area() { return Polygon<T>::w * Polygon<T>::h; }
};

template <class T>
class Triangle : public Polygon<T>{
public:
  Triangle(){}
  virtual T area() { return Polygon<T>::w * Polygon<T>::h / 2.; }
};
```

Class structures same as example_04
But this uses a template type “T” instead of “int”

polygon.h

C++ introduction ... example_05

```cpp
#include <iostream>
#include "polygon.h"

int main() {

  Polygon<float>   obj1;
  Rectangle<float> obj2;
  Triangle<float>  obj3;

  obj1.SetParams(5,5);
  obj2.SetParams(5,5);
  obj3.SetParams(5,5);

  std::cout
    << std::endl
    << "Area of (w,h) = (5,5) Polygon   : " << obj1.area() << std::endl
    << "Area of (w,h) = (5,5) Rectangle : " << obj2.area() << std::endl
    << "Area of (w,h) = (5,5) Triangle  : " << obj3.area() << std::endl
    << std::endl;

  return 0;
}
```

Template specialized to float type
(classes can be re-used for various type)

example_05.cc

```
Area of (w,h) = (5,5) Polygon   : -1
Area of (w,h) = (5,5) Rectangle : 25
Area of (w,h) = (5,5) Triangle  : 12.5
```

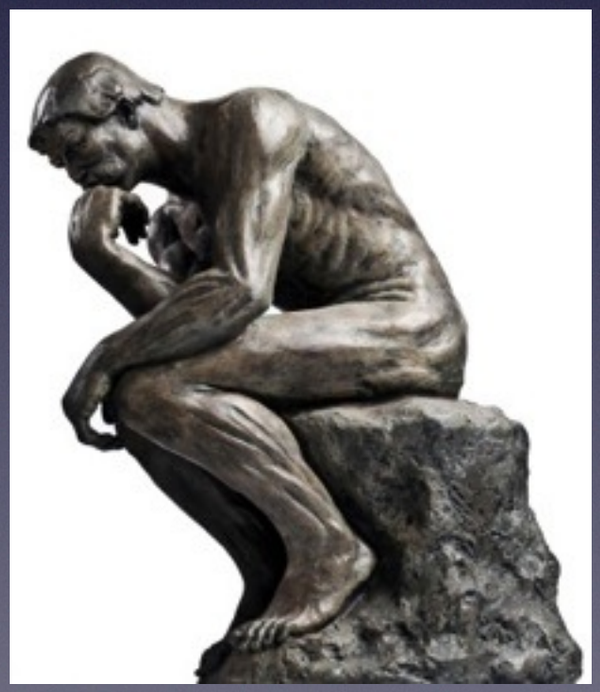
Output of executing example_05.exe

C++ introduction

- Hope those 6 examples taught/remind you...
  - How to write a simple C++ code quickly, compile, and test-run it
  - C++ class and functions
  - Inheritance and class/function template for re-usable code design

- Example code design for finding “hit” from an waveform

- I need to read an waveform from a data file
- I want to fill some histograms of “hit”
- I want to try writing two algorithms
  - Gaussian shaped waveform hit
  - Landau shaped waveform hit



C++ introduction

- Hope those 6 examples taught/remind you...
  - How to write a simple C++ code quickly, compile, and test-run it
  - C++ class and functions
  - Inheritance and class/function template for re-usable code design

- Example code design for finding “hit” from an waveform

I/O Class
Read data file

Manager Class
Use I/O class and executes algorithm

Algorithm base class
Defines analysis histograms

GausHit Algorithm
Find “gaussian hit”

LandauHit Algorithm
Find “landau hit”

- Writing an algorithm requires minimal effort
  - Focus on “find hit from an waveform”
  - Nothing else. No “data read”. No “make TH1”.
  - Algorithm becomes simple and readable
- Algorithm base class for common features
  - All algorithms benefit from changes here
    - e.g.) Add a new histogram
- I/O class decouples data product dependency
  - Simply replace I/O class for each fmwk
  - No need to change the rest of the code

OK... let’s take a break here...

Any question?

Any coffee/donuts left?