**Ekaba Bisong**
**Programming in C++**
**University of Calabar**

Tutorial Notes #6
May 08, 2015

# What is C++

The goal of this tutorial is to provide an introduction to programming using C++ as well as good software engineering principles.

*"Learning to program computers unlocks the full power of computer technology in a way that is both liberating and exciting" – Mehran Sahami.*

The purpose of a programming language is to help express ideas in code.

**The Design of C++**
The design of C++ has focused on programming techniques dealing with fundamental notions such as memory, mutability, abstraction, resource management, expression of algorithms, error handling, and modularity. Those are the most important concerns of a ***systems programmer*** and more generally of programmers of resource-constrained and high-performance systems.

C++ is a general-purpose programming language with a bias toward systems programming. C++ has indeed been used for an incredible variety of uses (from microcontrollers to huge distributed commercial applications), but the key point is that C++ is not deliberately specialized for any given application area.

**Programming Style**
The C++ language features most directly support four programming styles:

- Procedural programming
- Data abstraction
- Object-oriented programming
- Generic programming

- *Procedural programming:* This is programming focused on processing and the design of suitable data structures.
- *Data abstraction:* This is programming focused on the design of interfaces, hiding implementation details in general and representations in particular. C++ supports concrete and abstract classes.
- *Object-oriented programming:* This is programming focused on the design, implementation, and use of class hierarchies.
- *Generic programming:* This is programming focused on the design, implementation, and use of general algorithms. Here, 'general' means that an algorithm can be designed to accept a wide variety of types as long as they meet the algorithm's requirements on its arguments.

**Learning C++**

***The only way to learn how to program is to write programs.*** You'll learn a lot more by writing and debugging programs than you ever will by reading any book or this handout.

The task of learning a language should focus on mastering the native and natural styles for that language – not on understanding of every little detail of every language feature. ***Practical application of ideas is necessary.***

Many of the most fundamental programming concepts are represented in the standard library. Thus, learning the standard library is an integral part of learning C++. The standard library is the repository of much hard-earned knowledge of how to use C++ well.

C++ is a language that you can grow with.

The purpose of learning a programming language is to become a better programmer, that is, to become more effective at designing and implementing new systems and at maintaining old ones. For this, an appreciation of programming and design techniques is far more important than understanding all the details. The understanding of technical details comes with time and practice.

**What is C++ Used for?**

C++ is used just about everywhere: it is in your computer, your phone, your car, probably even in your camera. You don't usually see it. C++ is a systems programming language, and its most pervasive uses are deep in the infrastructure where we, as users, never look.

Early applications tended to have a strong systems programming flavor. For example, several early operating systems have been written in C++: [Campbell, 1987] (academic), [Rozier, 1988] (real time), [Berg, 1995] (high-throughput I/O). Many current ones (e.g., Windows, Apple's OS, Linux, and most portable-device OSs) have key parts done in C++. Your cellphone and Internet routers are most likely written in C++.

Some of today's most visible and widely used systems have their critical parts written in C++. Examples are Amadeus (airline ticketing), Amazon (Web commerce), Bloomberg (financial information), Google (Web search), and Facebook (social media).

Many other programming languages and technologies depend critically on C++'s performance and reliability in their implementation. Examples include the most widely used Java Virtual Machines (e.g., Oracle's HotSpot), JavaScript interpreters (e.g., Google's V8), browsers (e.g., Microsoft's Internet Explorer, Mozilla's Firefox, Apple's Safari, and Google's Chrome), and application frameworks (e.g., Microsoft's .NET Web services framework).

C++ wasn't specifically designed with numerical computation in mind. However, much numerical, scientific, and engineering computation is done in C++.

**[Note]**

- Contents of this handout are adapted from book:

  ***The C++ programming language by Bjarne Stroustrup. Fourth edition. Addison-Wesley. 2013. ISBN 978-0-321-56384-2***