Introduction to Computers and C++

1.1 Introduction

- C++—a powerful computer programming language that's appropriate for technically oriented people with little or no programming experience, and for experienced programmers to use in building substantial information systems.
- You'll write instructions commanding computers to perform those kinds of tasks.
- *Software* (i.e., the instructions you write) controls *hardware* (i.e., computers).
- You'll learn *object-oriented programming*—today's key programming methodology.
- You'll create many *software objects* in the real world.

1.1 Introduction (Cont.)

- C++ is one of today's most popular software development languages.
- This text provides an introduction to programming in C++11&C++14—standardized through the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).
- In use today are more than a billion general-purpose computers and billions more cell phones, smartphones and handheld devices (such as tablet computers).

C++ and Facebook Moments: Facebook code blog, Techworld

By Blog Staff | Jun 16, 2015 07:02 AM | News, Articles & Books | Tags: *None*

Here are two notable articles related to C++'s central role in Facebook's Moments app, released yesterday.

The first is the announcement on the Facebook code blog:

Under the Hood: Building Moments

by Ashwin Bharambe, Zack Gomez, and Will Ruben

From the article:

> ... There are many alternatives for sharing code between mobile platforms. We wanted to optimize for fast iteration, app performance, and native look and feel. After weighing the alternatives, we chose to write the UI in platform-specific code and the business logic in shared code using C++. Traditionally, C++ is known for providing high performance while lacking easy memory management and higher-level abstractions. However, using modern C++11 features such as std::shared_ptr reference counting, lambda functions, and auto variable declarations, we were able to quickly implement highly performant, memory-safe code...

The growing use of C++ for cross-platform shared code in mobile apps is not a new technical story in itself – last year's CppCon had multiple sessions about doing this including from Dropbox and Microsoft Office – but even the mainstream press is starting to notice this is happening more often:

1.5 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation steps*.
- These may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages

1.5 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

Machine Languages

- Any computer can directly understand only its own machine language (also called machine code), defined by its hardware architecture.
- Machine languages generally consist of numbers (ultimately reduced to 1s and 0s). Such languages are cumbersome for humans.

1.5 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

Assembly Languages

- English-like *abbreviations* to represent elementary operations. These abbreviations formed the basis of assembly languages.
- *Translator programs* called assemblers were developed to convert early assembly-language programs to machine language.

1.5 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

High-Level Languages

- To speed up the programming process further, high-level languages were developed in which single statements could be written to accomplish substantial tasks.
- Translator programs called compilers convert high-level language programs into machine language.
- Allow you to write instructions that look more like everyday English and contain commonly used mathematical expressions.

1.5 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

- Compiling a high-level language program into machine language can take a considerable amount of computer time.
- Interpreter programs were developed to execute high-level language programs directly (without the need for compilation), although more slowly than compiled programs.
- Scripting languages such as the popular web languages JavaScript and PHP are processed by interpreters.

Performance Tip 1.1

Interpreters have an advantage over compilers in Internet scripting. An interpreted program can begin executing as soon as it's downloaded to the client's machine, without needing to be compiled before it can execute. On the downside, interpreted scripts generally run slower than compiled code.

1.6 C++

- C++ evolved from C, which was developed by Dennis Ritchie at Bell Laboratories.
- C
  - Available for most computers and is hardware independent.
  - It's possible to write C programs that are portable to most computers.
  - The widespread use of C with various kinds of computers (sometimes called hardware platforms) led to many variations.
  - American National Standards Institute (ANSI) cooperated with the International Organization for Standardization (ISO) to standardize C worldwide.
  - Joint standard document was published in 1990 and is referred to as *ANSI/ISO 9899: 1990*.

1.6 C++ (Cont.)

- C11
  - Latest ANSI standard for the language.
  - Developed to evolve the C language to keep pace with increasingly powerful hardware and ever more demanding user requirements.
  - Makes C more consistent with C++.
- C++, an extension of C, was developed by Bjarne Stroustrup in 1979 at Bell Laboratories.
- C++ provides a number of features that "spruce up" the C language, but more importantly, it provides capabilities for object-oriented programming.

1.6 C++ (Cont.)

- C++ Standard Library
  - C++ programs consist of pieces called classes and functions.
  - Most C++ programmers take advantage of the rich collections of classes and functions in the C++ Standard Library.
  - Two parts to learning the C++ "world."
  - The C++ language itself, and
  - How to use the classes and functions in the C++ Standard Library.
  - Many special-purpose class libraries are supplied by independent software vendors.

Software Engineering Observation 1.1

Use a “building-block” approach to create programs. Avoid reinventing the wheel. Use existing pieces wherever possible. Called software reuse, this practice is central to object-oriented programming.

Software Engineering Observation 1.2

When programming in C++, you typically will use the following building blocks: classes and functions from the C++ Standard Library, classes and functions you and your colleagues create and classes and functions from various popular third-party libraries.

Performance Tip 1.2

Using C++ Standard Library functions and classes instead of writing your own versions can improve program performance, because they're written carefully to perform efficiently. This technique also shortens program development time.

Portability Tip 1.1

Using C++ Standard Library functions and classes instead of writing your own improves program portability, because they're included in every C++ implementation.

1.7 Programming Languages

- In this section, we provide brief comments on several popular programming languages (Fig. 1.5).

Programming language	Description
Fortran	Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s to be used for scientific and engineering applications that require complex mathematical computations. It's still widely used and its latest versions support object-oriented programming.
COBOL	COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users based on a language developed by Grace Hopper, a career U.S. Navy officer and computer scientist. COBOL is still widely used for commercial applications that require precise and efficient manipulation of large amounts of data. Its latest version supports object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 1 of 7.)

Programming language	Description
Pascal	Research in the 1960s resulted in *structured programming*—a disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than large programs produced with previous techniques. One of the more tangible results of this research was the development of Pascal by Professor Niklaus Wirth in 1971. It was designed for teaching structured programming and was popular in college courses for several decades.
Ada	Ada, based on Pascal, was developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s. The DOD wanted a single language that would fill most of its needs. The Pascal-based language was named after Lady Ada Lovelace, daughter of the poet Lord Byron. She's credited with writing the world's first computer program in the early 1800s (for the Analytical Engine mechanical computing device designed by Charles Babbage). Ada also supports object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 2 of 7.)

Programming language	Description
Basic	Basic was developed in the 1960s at Dartmouth College to familiarize novices with programming techniques. Many of its latest versions are object oriented.
C	C was implemented in 1972 by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most of the code for general-purpose operating systems is written in C or C++.
Objective-C	Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads).

Fig. 1.5 | Some other programming languages. (Part 3 of 7.)

Programming language	Description
Java	Sun Microsystems in 1991 funded an internal corporate research project led by James Gosling, which resulted in the C++-based object-oriented programming language called Java. A key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-control devices. This is sometimes called "write once, run anywhere." Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (e.g., smartphones, tablets, television set-top boxes, appliances, automobiles and more) and for many other purposes. Java is also the key language for developing Android smartphone and tablet apps.
Visual Basic	Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its latest versions support object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 4 of 7.)

Programming language	Description
C#	Microsoft's three object-oriented primary programming languages are Visual Basic (based on the original Basic), Visual C++ (based on C++) and C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).
PHP	PHP is an object-oriented, "open-source" (see Section —) "scripting" language supported by a community of users and developers and is used by numerous websites including Wikipedia and Facebook. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including MySQL.
Perl	Perl (Practical Extraction and Report Language), one of the most widely used object-oriented scripting languages for web programming, was developed in 1987 by Larry Wall. It features rich text-processing capabilities and flexibility.

Fig. 1.5 | Some other programming languages. (Part 5 of 7.)

Programming language	Description
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam (CWI), Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces.
JavaScript	JavaScript is the most widely used scripting language. It's primarily used to add programmability to web pages—for example, animations and interactivity with the user. It's provided with all major web browsers.

Fig. 1.5 | Some other programming languages. (Part 6 of 7.)

Programming language	Description
Ruby on Rails	Ruby—created in the mid-1990s by Yukihiro Matsumoto—is an open-source, object-oriented programming language with a simple syntax that's similar to Perl and Python. Ruby on Rails combines the scripting language Ruby with the Rails web application framework developed by 37Signals. Their book, *Getting Real* (available free at `gettingreal.37signals.com/toc.php`), is a must read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications. Ruby on Rails was used to build Twitter's user interface.
Scala	Scala (`www.scala-lang.org/node/273`)—short for "scalable language"—was designed by Martin Odersky, a professor at École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. Released in 2003, Scala uses both the object-oriented programming and functional programming paradigms and is designed to integrate with Java. Programming in Scala can reduce the amount of code in your applications significantly. Twitter and Foursquare use Scala.

Fig. 1.5 | Some other programming languages. (Part 7 of 7.)

1.9 Typical C++ Development Environment (Cont.)

- C++ systems generally consist of three parts: a program development environment, the language and the C++ Standard Library.
- C++ programs typically go through six phases: edit, preprocess, compile, link, load and execute.

1.9 Typical C++ Development Environment (Cont.)

- Phase 1 consists of editing a file with an *editor* program, normally known simply as an editor.
  - Type a C++ program (source code) using the editor.
  - Make any necessary corrections.
  - Save the program.
  - C++ source code filenames often end with the `.cpp`, `.cxx`, `.cc` or `.C` extensions (note that `C` is in uppercase) which indicate that a file contains C++ source code.

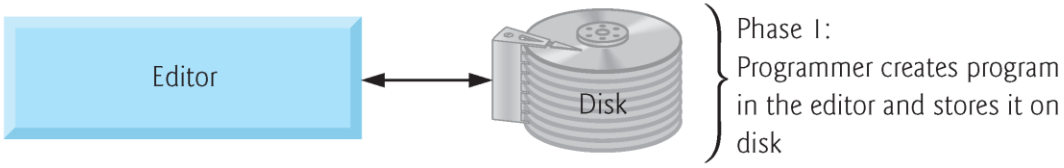

Fig. 1.6 | Typical C++ development environment—editing phase.

1.9 Typical C++ Development Environment (Cont.)

- Linux editors: `vi` and `emacs`.
- C++ software packages for Microsoft Windows such as Microsoft Visual C++ (`microsoft.com/express`) have editors integrated into the programming environment.
- You can also use a simple text editor, such as Notepad in Windows, to write your C++ code.
- integrated development environments (IDEs)
  - Provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating logic errors—errors that cause programs to execute incorrectly.

1.9 Typical C++ Development Environment (Cont.)

- Popular IDEs
  - Microsoft® Visual Studio (Express Edition)
  - Dev C++
  - NetBeans
  - Eclipse
  - Apple's Xcode
  - CodeLite
  - Qt

1.9 Typical C++ Development Environment (Cont.)

- In phase 2, you give the command to compile the program.
  - A **preprocessor** program executes automatically before the compiler's translation phase begins (so we call preprocessing Phase 2 and compiling Phase 3).
  - The C++ preprocessor obeys commands called preprocessing directives, which indicate that certain manipulations are to be performed on the program before compilation.
  - These manipulations usually include other text files to be compiled, and perform various text replacements.
  - The most common preprocessing directives are discussed in the early chapters; a detailed discussion of preprocessor features appears in Appendix E, Preprocessor.

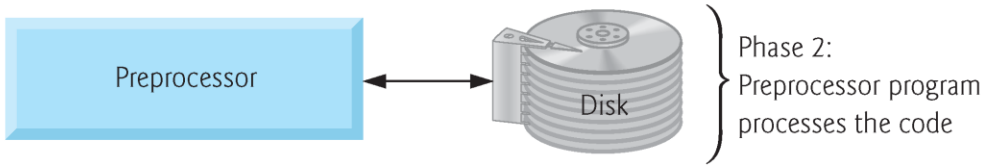

Fig. 1.7 | Typical C++ development environment—preprocessor phase.

1.9 Typical C++ Development Environment (Cont.)

- In Phase 3, the compiler translates the C++ program into machine-language code—also referred to as object code.

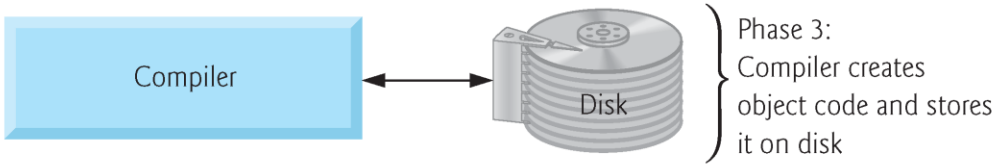

Fig. 1.8 | Typical C++ development environment—compilation phase.

1.9 Typical C++ Development Environment (Cont.)

- Phase 4 is called linking.
  - The object code produced by the C++ compiler typically contains "holes" due to these missing parts.
  - A linker links the object code with the code for the missing functions to produce an executable program.
  - If the program compiles and links correctly, an executable image is produced.

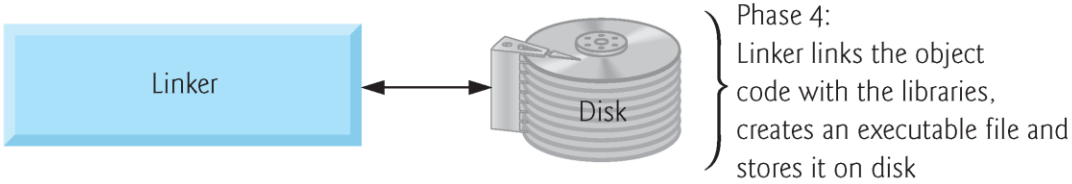

Fig. 1.9 | Typical C++ development environment—linking phase.

1.9 Typical C++ Development Environment (Cont.)

- Phase 5 is called loading.
  - Before a program can be executed, it must first be placed in memory. This is done by the loader, which takes the executable image from disk and transfers it to memory.
  - Additional components from shared libraries that support the program are also loaded.

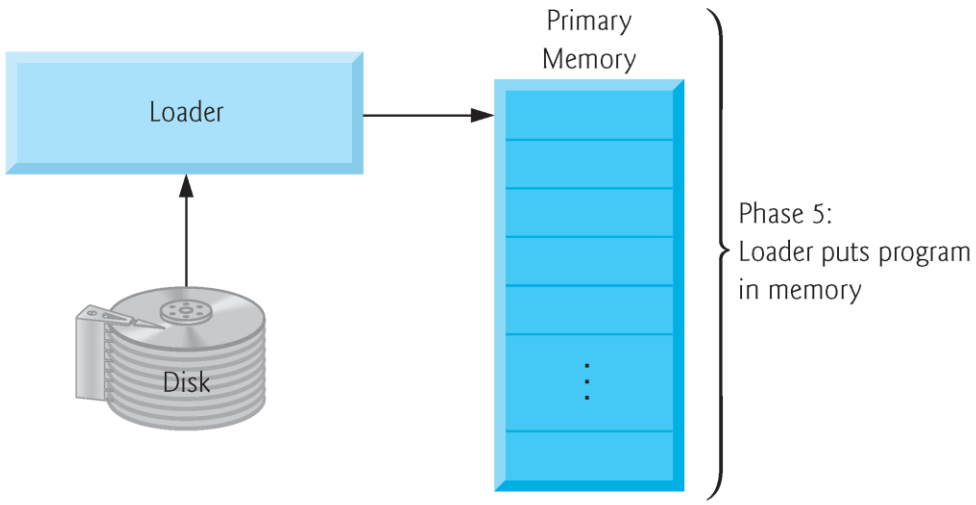

Fig. 1.10 | Typical C++ development environment—loading phase.

1.9 Typical C++ Development Environment (Cont.)

- Phase 6: Execution
  - Finally, the computer, under the control of its CPU, executes the program one instruction at a time.
  - Some modern computer architectures can execute several instructions in parallel.

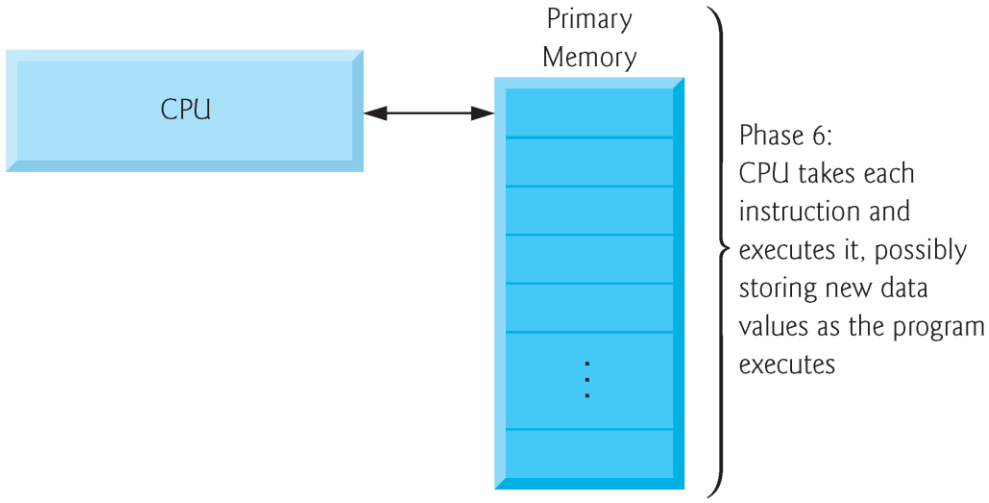

Fig. 1.11 | Typical C++ development environment—execution phase.

1.9 Typical C++ Development Environment (Cont.)

- Data may be output to other devices, such as disks and hardcopy printers.
- There is also a standard error stream referred to as cerr. The `cerr` stream is used for displaying error messages.

Common Programming Error 1.1

Errors such as division by zero occur as a program runs, so they're called runtime errors or execution-time errors. Fatal runtime errors cause programs to terminate immediately without having successfully performed their jobs. Nonfatal runtime errors allow programs to run to completion, often producing incorrect results.