C++ Lecture 1

* Introduction to Computers and C++ Programming
* CIS 251 • Shelby-Hoover Campus

Computer Basics

* Hardware
  + Input Devices: Keyboard, Mouse
  + Output Devices: Monitor, Printer
  + Processor (CPU)
  + Main Memory (RAM)
  + Secondary Memory (Storage)
* Software
  + Program: a set of instructions for the computer to perform, usually on a set of data
  + Most application software communicates with the computer through the operating system

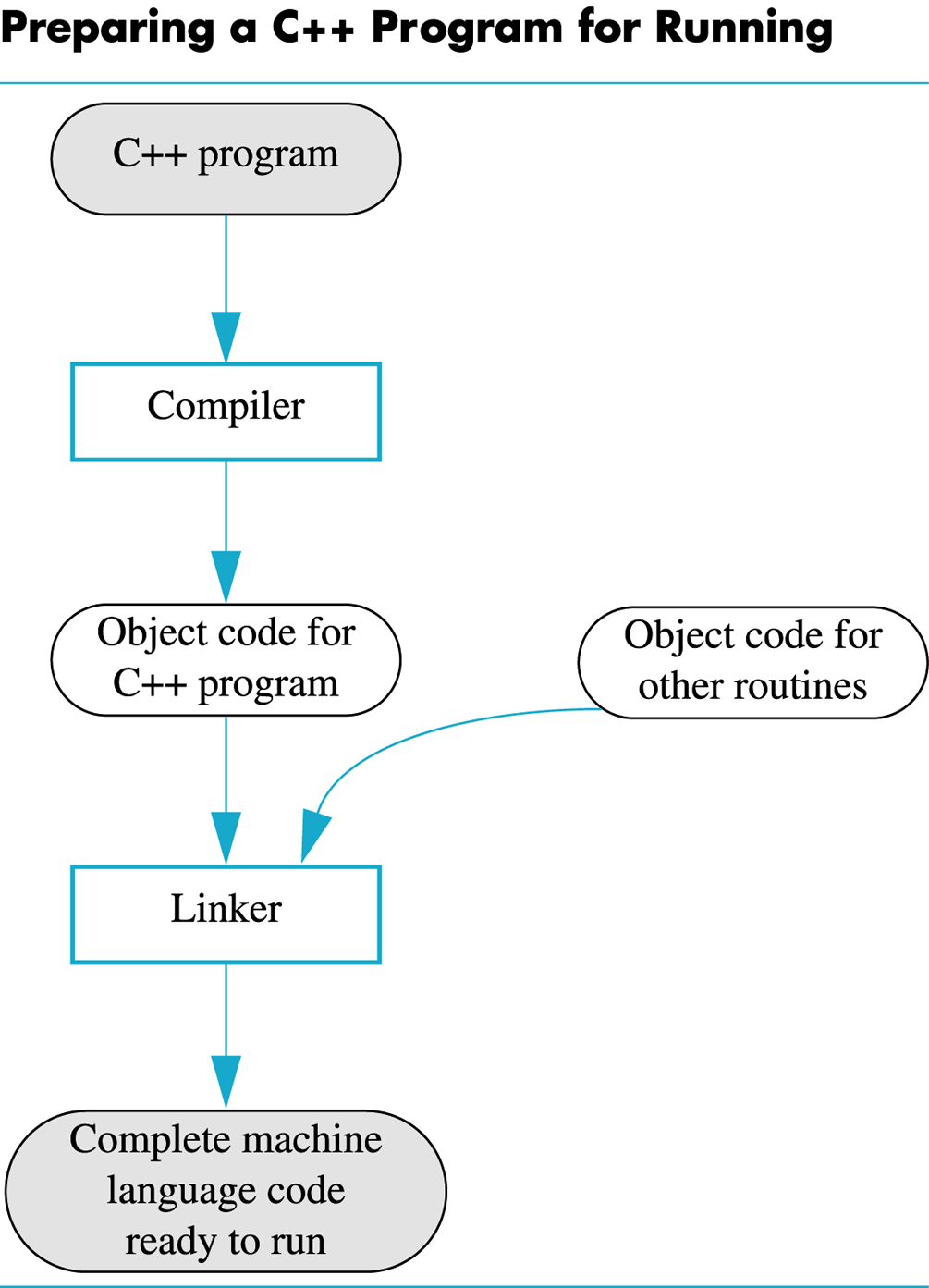
Programming Basics

* Underlying information is stored / manipulated in **binary** (0 for off, 1 for on)
* Processors understand rudimentary memory operations written in a **low-level language**
  + machine language (binary)
  + assembly language (simple conversion to binary)
* Programmers usually write programs in a **high-level language**
  + C++, C#, Java, Basic, Pascal, FORTRAN, COBOL
  + Easier for humans to write, understand

High Level to Low Level

* Code written in a high-level language must be converted to a low-level language form before the instructions can be executed
* A **compiler** converts a set of high-level language instructions into a low-level language all at once
  + The high-level language code is known as **source code**
  + The low-level language code that results is **object code**
* In C++, an additional tool, the **linker**, combines the object code from the compiler with the object code for predefined functions to generate the **executable** form
* Many environments combine the compile and link commands into a single operation (build)

Compiling and Linking



Language Components

* **Keywords / Reserved Words**: words with predefined meaning (cannot be used for other purposes)
* **Operators**: symbols or words that perform operations
* **Punctuation**: characters that mark the beginning or end of a statement or block, separate items in a list, etc.
* **Programmer-Defined Names / Identifiers**: identify memory locations, code units, new data types
* **Syntax**: rules that govern how the various language elements are combined to form statements and programs

The Programming Process

* Remember: the point at which you enter code in a specific language should not be the beginning point of the programming process
* There are several different methods that may be applied to a problem to generate a solution
* Problem-Solving Phase
  + Define the problem
  + Craft an algorithm (flowchart, pseudocode)
  + Test the algorithm and refine if necessary
* Implementation Phase
  + Express the algorithm in a specific language
  + Correct syntax errors that inhibit compilation
  + Test the resulting executable and refine the code if necessary

Programming Approaches

* **Procedural Programming** separates the data from the instructions performed on the data; the focus is on the sets of instructions (procedures, functions, methods)
* **Object-Oriented Programming** unites individual data elements into units with predefined interaction
  + **Object**: a group of variables representing some entity
  + **Method**: a procedure that can be used to access or modify the data stored in the object
  + **Class**: the code that defines the makeup of a particular type of object, including the data each object contains and what operations can be performed on that object

An Introduction to C++

* Derived from the C language
  + Developed at AT&T Bell Laboratories in the 1970s
  + Used for writing and maintaining UNIX
  + Contains both high-level and low-level attributes
* C++ is a variation on the C language
  + Developed at AT&T Bell Laboratories in the early 1980s
  + “One better” than C (the ++ operation adds one)
  + Supports object-oriented programming
  + Instructions written in C are also compatible with C++ (though the reverse is not always true)

C++ Structure

* Your initial programs will follow this pattern:

#include <iostream>

using namespace std;

int main()

{

🡨 Your code goes here

return 0;

}

Some Basic C++ Syntax

* In C++, a statement (a single instruction) ends with a semicolon
* Declaration
  + Listing the data type and name of a variable to be used
  + Example: int studentCount; (int is used for whole numbers)
* Input and Output
  + The key word cout represents generating output; cin represents obtaining input
  + cout << "Hello!\n"; displays Hello! and starts a new line
  + cin >> studentCount; stores the next item the user enters as input in the variable studentCount (note the difference in the brackets)
* Assignment
  + Storing a predetermined value in a variable
  + The variable must be on the left side of the equal sign; the value goes on the right
  + studentCount = 15; assigns 15 to the variable
  + studentCount = studentCount + 1; adds 1 to the existing value of the variable and assigns the sum as the new value

C++ Compilers

* The code presented in this course conforms to the C++ standard
* Older compilers may use a pre-standard version of C++ syntax
  + #include <iostream.h> instead of <iostream>
  + using namespace std; omitted entirely
  + void main() instead of int main()
  + return 0; omitted entirely