Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-one)

1.1: Prologue

When writing a program, especially in the Internet age, there are several factors that determine its success. Since the Internet is composed of a variety of operating systems and platforms, a program is expected to run on most of them flawlessly. Also, a program should be robust -- it should allow the user to be productive by not crashing. Another big issue is security. Since programs can come from anywhere or anyone, a certain amount of security control is now necessary.

A key to a program's usability is its portability or the ability to have it run on multiple platforms. Imagine writing a program in a language such as C or C++, the most widely used computer languages prior to Java. Chances are this program will run only on one type of machine or platform. If you want your program to be used by as large an audience as possible, you will want to make it portable. C code is compiled or translated from what the programmer writes into a stream of machine-specific instructions that a specific microprocessor can understand. Thus, a C program that is targeted for the Unix platform will not run on a Windows machine. While not impossible, making C code portable is time-consuming and difficult. It will require specific code changes for each platform. This results in increased complexity as the code grows significantly.

Complex code is often not robust or bug free. It often behaves in an erratic fashion that could possibly cause damage. Who has not experienced running a program with bugs? Just when you are about to save your work, the program freezes, forcing a reboot of the machine. Often, the work you had just performed is lost. In C and C++, the use of pointers contributes greatly to this problem. A pointer is a language construct that lets the contents of one variable "point" to another piece of data. Bugs in pointers can cause the program to crash, or at worst can cause the entire computer to crash, sometimes with permanent loss of data.

In addition to not being robust, a program may over step its bounds and do things that are unauthorized. When you have widely distributed applications, many programs come from unknown sources. Without any assurances, a user should be rightfully concerned about a program's behavior. C and C++ do not enforce security and are given free reign on a machine. Without restrictions, it is very easy to write viruses and other destructive programs. For example, it would be fairly simple to create a program that sends your private files such as your tax returns to a hacker without your knowledge. Another example is a game that is actually a disguised virus that reformats your hard drive. Security features built into a language prevent this kind of horseplay and ease the user's mind.