Introduction to Programming
Learning Goals

• Understand at a conceptual level
  – What is a computer?
  – What is a program?
  – Why learn to program?
  – What is hard about learning to program?
  – What is hard about teaching programming?
  – Strategies for teaching programming
What is a Computer?

- A device that performs high-speed mathematical and/or logical operations or that assembles, stores, correlates, or otherwise processes information.
- The first *computers* were *people* – who did computations
What is Programming?

• Creating detailed instructions that a computer can execute to accomplish some task.
  – Like writing a recipe for your favorite dish
  – Or giving someone directions to your house
  – Or making a robot do what you want
Early Programming

• Early computers required the programmer to set switches and move wires
  – Which represented a series of 1’s and 0’s

• Later computers were programmed using punched cards
Language Evolution

• Early languages were based on how to do instructions on each machine
  – 1’s and 0’s to add, subtract, read, store, etc

• Assembler allowed you to write programs using names for instructions and memory
  – But still translated into machine language

• Higher-level languages
  – Are compiled into machine language or virtual machine language
Java

- Developed at Sun in the early 1990s
  - Invented by James Gosling
- Similar to C++ in syntax but easier to use
  - Less likely to crash
- Cross-platform, object-oriented language
- Used in business, science, and education
- One of the fastest adopted technologies of all time
Which Language?

• All high-level languages are eventually translated into machine language
• You can write the same program in any language
  – The computer doesn’t care what high-level language you use
• The language matters to the programmer
  – How long does it take to write the program?
  – How hard is it to change the program?
  – How long does it take to execute?
Why Don’t We Just Use English?

• English is good for communication between two intelligent humans
  – Even then we sometimes don’t understand

• Computers are very stupid
  – They basically know how to add, compare, store, and load
  – Programs are very detailed instructions
    • Everything must be precise and unambiguous
Programming Exercise

• Write down instructions for how to make a sandwich
• Have another group read the directions and do the actions
  – stop anytime anything isn’t clear and ask for clarification
Why Learn to Program?

- Alan Perlis, first head of Carnegie Mellon's Computer Science Department, made the claim in 1961 that computer science, and programming explicitly, should be part of a liberal education.
- Seymour Papert claimed in the 70’s and 80’s that learning to program is “learning to think, and debug one’s own thoughts.”
  - If you learned to program, you learned to plan, to debug, to handle complexity, etc.
- Twenty years of research found that that is simply not true.
- Most people don’t learn to program.
What CS Education Research Tells Us

• Most people don’t learn to program in one semester
  – Alan Perlis, “Most people find the concept of programming obvious, but the doing impossible.”

• Many people find CS classes boring and irrelevant

• Most people can’t transfer what they learn in programming

• Students have a hard time putting statements together to accomplish a task

• Students learn much less than teachers think they will
So, Why Learn to Program?

• The computer is the most amazingly creative device that humans have ever conceived of. If you can imagine it, you can make it “real” on a computer.

• Computers will continue to have a major impact on modern life
  – Movies, games, business, healthcare, science, education, etc
Computers Are Commonplace

- Computers, or at least processors, are in many common devices
Programming is Communicating

• Alan Perlis, “You think you know when you can learn, are more sure when you can write, even more sure when you can teach, but certain when you can program.”
What is Hard About Programming?

• It is easier to write an incorrect program than understand a correct one – Alan Perlis

• Beginners have a hard time understanding some of the core concepts
  – Boolean expressions with more than two items
    • if \((a < b)\) is okay
    • if \((a < b \&\& c > d)\) is hard
  – Iteration (loops)

• Beginners have a hard time putting statements together to accomplish a task
Strategies for Teaching Programming

• Do Live Programming
• Choose Depth over Breadth
• Assign Interesting Programs
• Use Pair Programming
• Start by Modifying Programs
• Have Lots of Small Projects
• Go from Concrete to Abstract
Do Live Programming

• Program in front of the students and talk about what you are doing and why you are doing it
  – Programming is a new and strange activity for most people
  – Talk through the algorithms and how to translate them to code
  – Let them see you make mistakes and fix them
  – Learn by example
Choose Depth over Breadth

- Cover difficult topics in depth
- Cover topics in more than one way
- Try to get people to connect the concept to something they already know to reduce “brittle knowledge”
Assign Interesting Programs

• A basic problem is that students find programming projects irrelevant

• Use motivating, relevant examples and projects
  – Games, digital video special effects, animation, graphics, pictures, sound, web pages, simulations
Use Pair Programming

- Have two students work together on programs
- They can explain what they are thinking to each other
  - Learn from each other
- One may be better at typing
  - Reducing frustration
Start by Modifying Programs

• Students won’t have to remember everything at once
  – Just focus on the part they are trying to understand
  – Less daunting than starting with nothing
  – Just getting the syntax right is time consuming
  – Reading code is a valuable skill

• The given code provides a sample of good coding practice
  – Teach by example
  – Gives advanced students something to learn from
Lots of Small Projects

• Programming is a skill
  – It takes practice

• Getting the program to work
  – Is frustrating while you are working on it
  – Rewarding when it works

• Students take much longer to program than teachers estimate
  – Expert programmers are much, much faster than beginners
Go from Concrete to Abstract

- Use props, live demonstrations, role playing
  - People learn better when you start with concrete things and later introduce abstract ideas
- Try to link concepts to students lives
  - To make it relevant
  - Reduce brittle knowledge
Summary

• Computers are fairly simple machines
  – Fancy calculators with lots of storage
  – But incredibly fast

• Computers have changed modern life

• Programs are instructions to a computer to accomplish a task

• Programs written in high-level languages are translated to machine or assembly language by a compiler

• Programming is hard to learn
  – But there are some ways to make it easier