CS103 Lecture 1 Slides

Introduction

Mark Redekopp

What is Computer Science

- All science is computer science
  - It is very interdisciplinary: Math, Engineering, Medicine, Natural sciences, Art, Linguistics, Social Sciences etc.
- It is about developing algorithms to solve information-based problems
  - How do I recognize objects in a photograph
  - How do I determine the best web page to return given a search query
  - Identify the function of this protein given it structure
- Computer science is no more about computers than astronomy is about telescopes
  - Computers are the primary tool

Computer Science Is…

- Essential to economic growth and development
- Dealing with society's problems
  - Health and E-Science
  - Big Data / Analytics
    - Conservation & the environment
    - Developing personalized learning
  - Who you might want to date ☺
- A great way to make a living
  - ***Maria Klawe, et. al. - To the age-old question -- "What do you want to do when you grow up?" -- children today give many modern answers: "Help feed hungry families." "Prevent and cure diseases." "Find sources of renewable energy." "Understand the universe."***
    One clear path leads to each of these aspirations: the study of computer science (& engineering).
    http://www.huffingtonpost.com/maria-klawe/computing-our-childrens-f_b_388874.html

More Applications

- 3D-Printables
- Music
- Math Art
- Visual Effects
- Self-Driving Vehicles
- Virtual Surgery

What Computer Scientist Do…

- Find methods to solve a problem (algorithms) **[this is truly CS]**
  - Observe, organize, transform and discover useful information from data
  - Use math and logic to solve problems
  - Work in (cross-discipline) groups
- Convert these methods/algorithms to a form a computer can execute **[this is programming]**
- **We generally do both at the same time**

What Is this Course About

- Introduction to Programming
  - Introduction: Doesn't require prior programming experience
    - However, we will move at a good pace so you must be prepared to put in some extra time if you've never coded before
    - However, those without any programming may want to consider CS 102 as a slower-paced on ramp to programming
  - Programming
    - We'll try to teach good coding practices and how to find efficient solutions (not just any solution)
    - We'll focus on concepts present in most languages using C/C++ as the primary language (not Java)

High Level Languages

Mother Tongues

Tracing the roots of computer languages through the ages

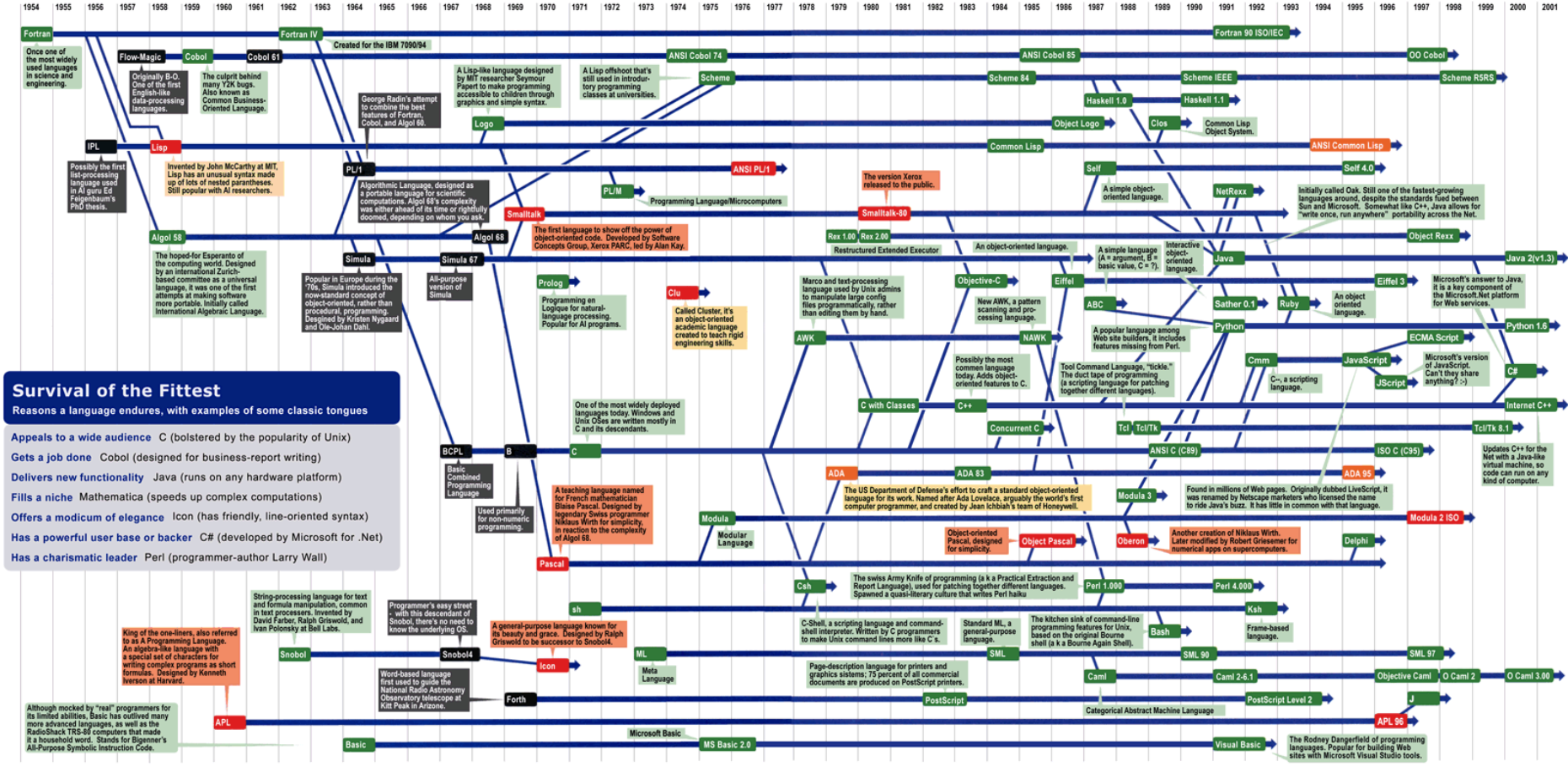
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html.* - Michael Mendeno

Key

- 1954 Year Introduced
- Active: thousands of users
- Protected: taught at universities; compilers available
- Endangered: usage dropping off
- Extinct: no known active users or up-to-date compilers
- Lineage continues



Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

http://www.digibarn.com/collections/posters/tongues/ComputerLanguagesChart-med.png

Why C/C++

- A very popular language in industry
- C/C++ is close to the actual hardware
  - Makes it fast & flexible (Near direct control of the HW)
  - Makes it dangerous (Near direct control of the HW)
  - Most common in embedded devices (your phone, car, wireless router, etc.)
- C/C++ is ubiquitous
  - Used everywhere, even to implement other programming languages (i.e. Python, Matlab, etc.)
- Principles learned in C/C++ will allow you to quickly learn other programming languages
  - C/C++ is extremely broad and thus covers concepts present in most other languages
- Not Java

Syllabus

Course Advice

- Catch the wave!
  - Overestimate the time you will need and your ability to get your work done
  - Limit extracurricular activities in the 1st semester
  - Don't let shame or embarrassment keep you from the help you need
- Experiment and fail
  - Learning to "debug" your programs is JUST AS important as learning to code correctly the first time
- Practice, practice, practice!
  - Computer science and programming require practice
  - It's like learning a musical instrument
- Let's have fun!

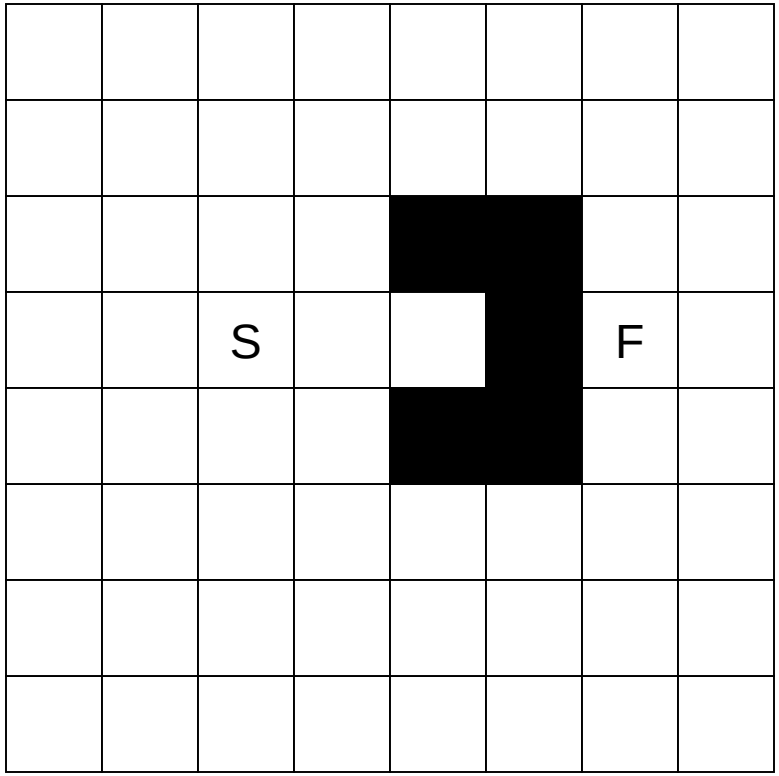
20-Second Timeout

- Who Am I?
  - Teaching faculty in CENG
  - Undergrad at USC in CECS
  - Grad at USC in EE
  - Work(ed) at Raytheon
  - Learning Spanish (and Chinese?)
  - Sports enthusiast!
    - Basketball
    - Baseball
    - Ultimate Frisbee?

THINK LIKE A COMPUTER

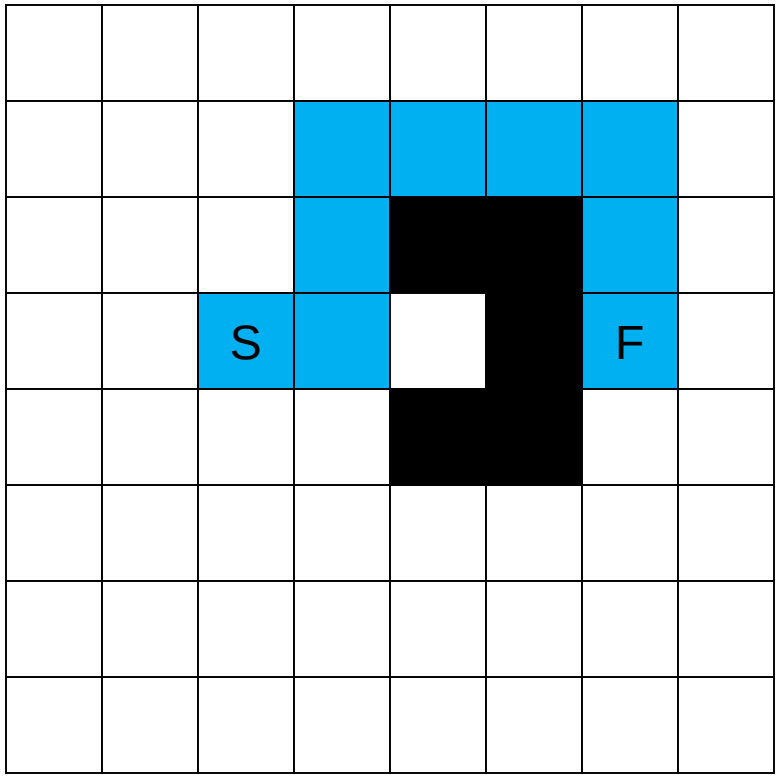
Path Planning

- Find shortest path from S to F



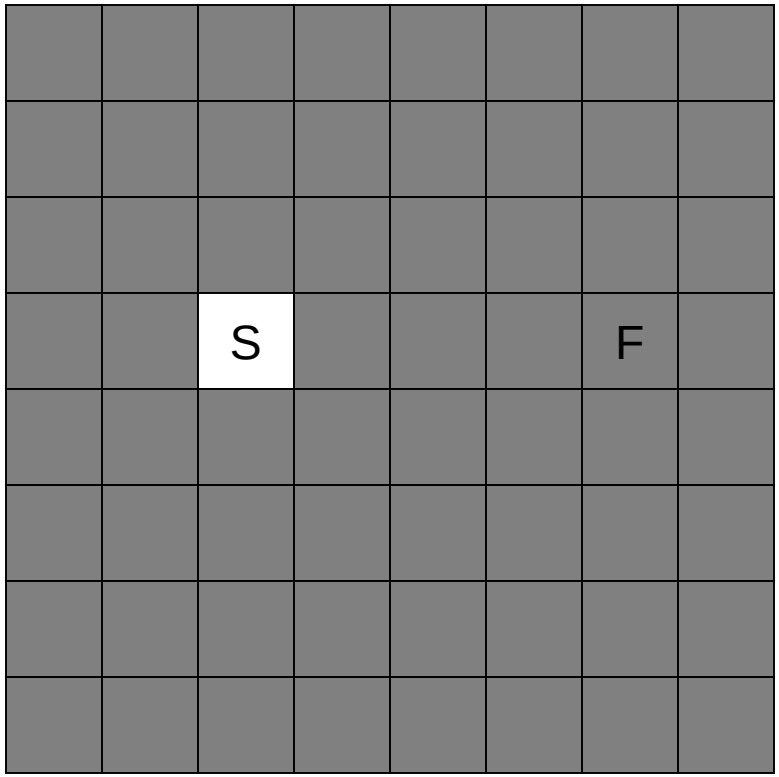
Path Planning

- Find shortest path from S to F



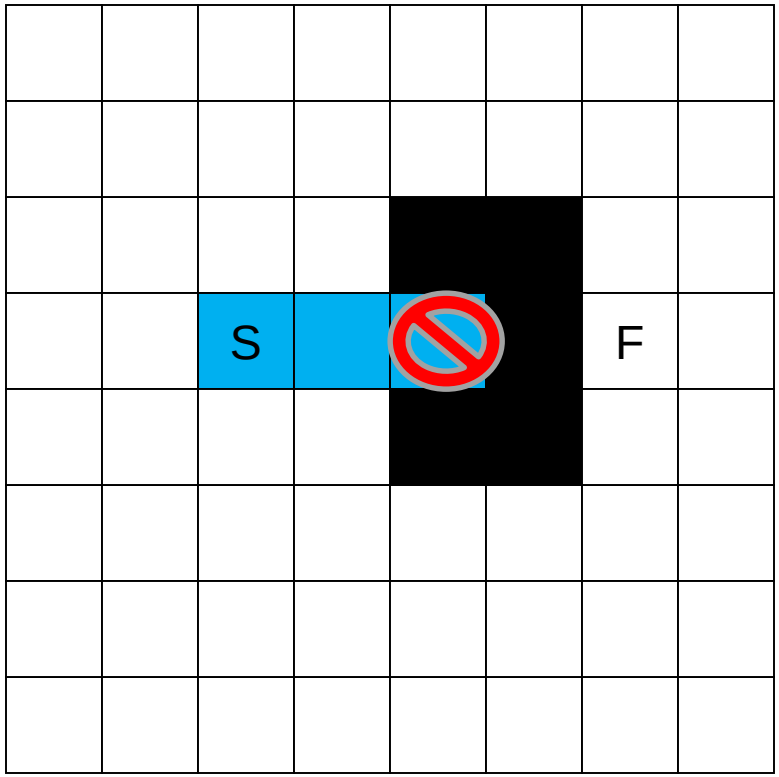
Path Planning

- Let's say you are a computer controlled robot. A computer usually can only process (or "see") one or two data items (a square) at a time

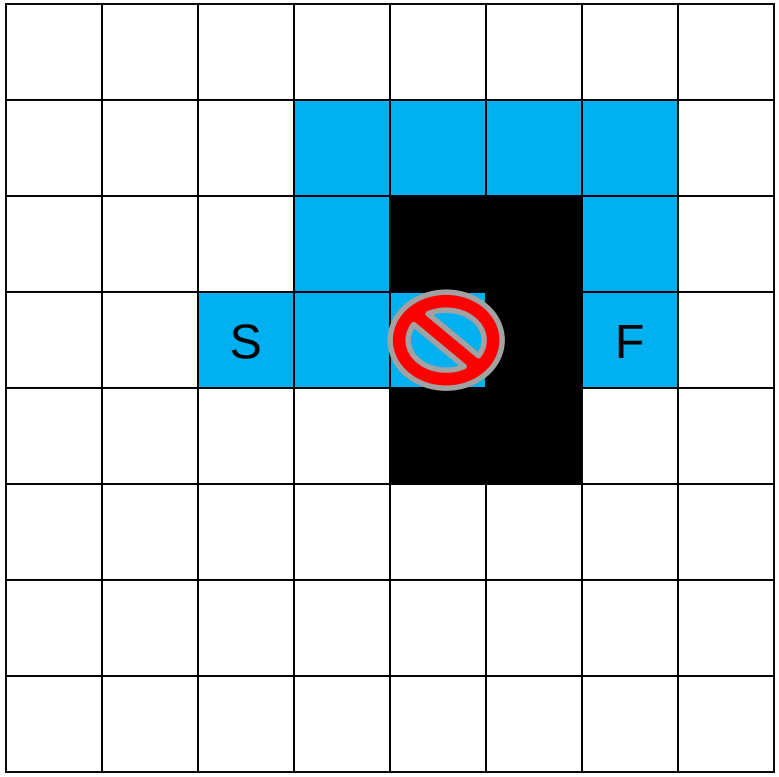


May just compute a straight line path from ‘S’ to ‘F’

Path Planning

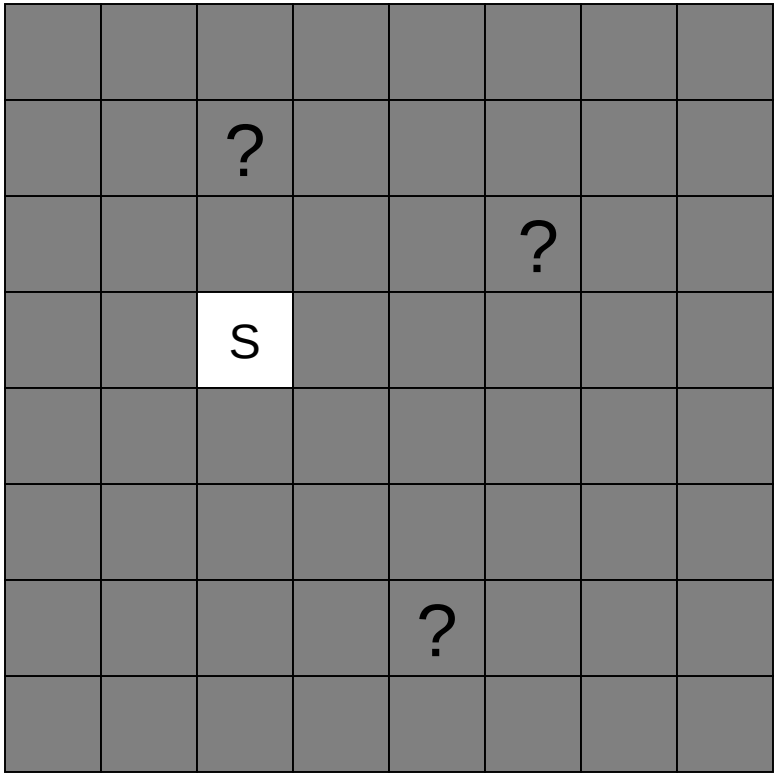


Path Planning



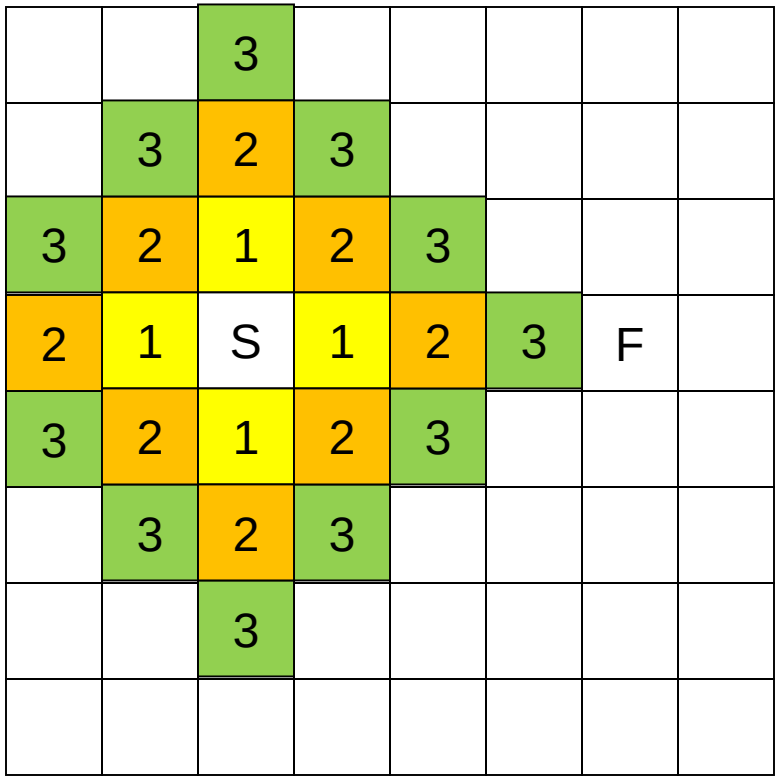
Path Planning

- What if I don't know where the Finish square is? You can examine any square in any order (no longer a robot). Can you devise a general search order to find the shortest path to 'F' while examining the minimum number of squares as possible.



Path Planning

- Examine all closer squares one at a time before progressing to further squares.



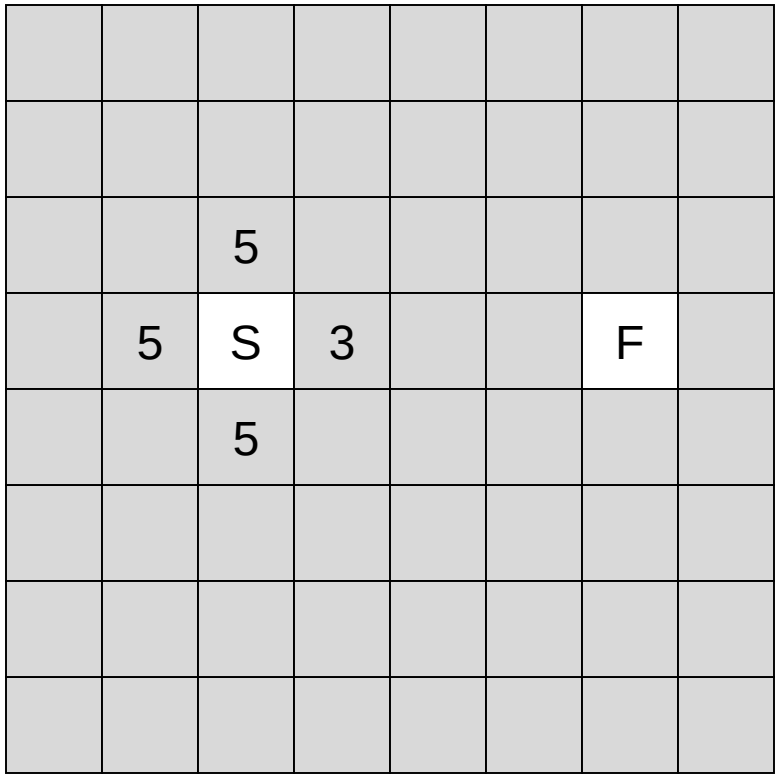
If you don't know where F is and want to find the shortest path, you have to do it this way

Uninformed search for shortest path:

Breadth-first

Path Planning

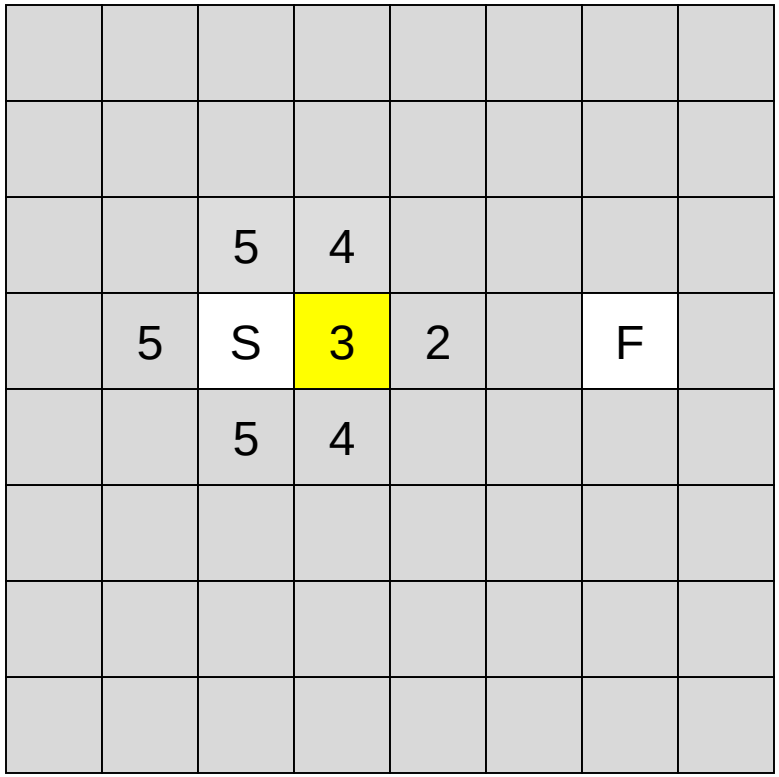
- Now I'll tell you where F is
- Can that help you reduce the number of squares explored?



Select a square to explore with minimum distance to the finish

Path Planning

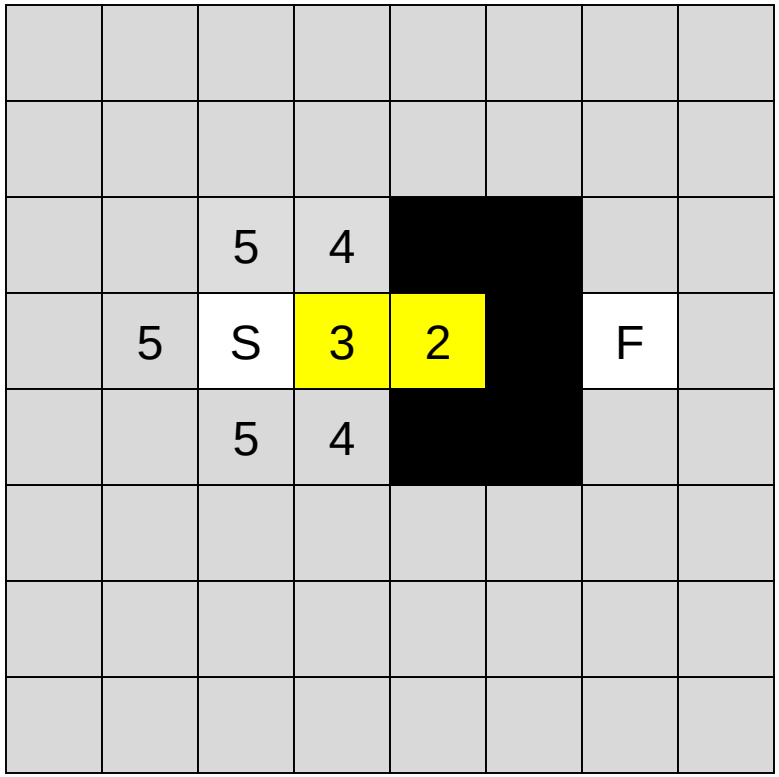
- Now I'll tell you where F is
- Can that help you reduce the number of squares explored?



Select a square to explore with minimum distance to the finish

Path Planning

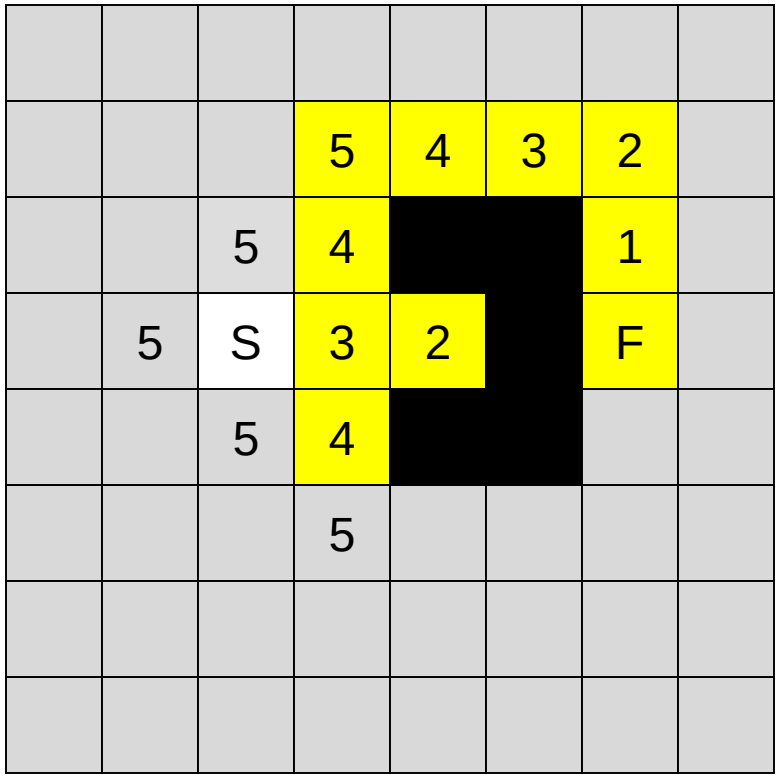
- But what if we run into a blockage?
  - Now we would pick the best among the remainder.



Select a square to explore with minimum distance to the finish

Path Planning

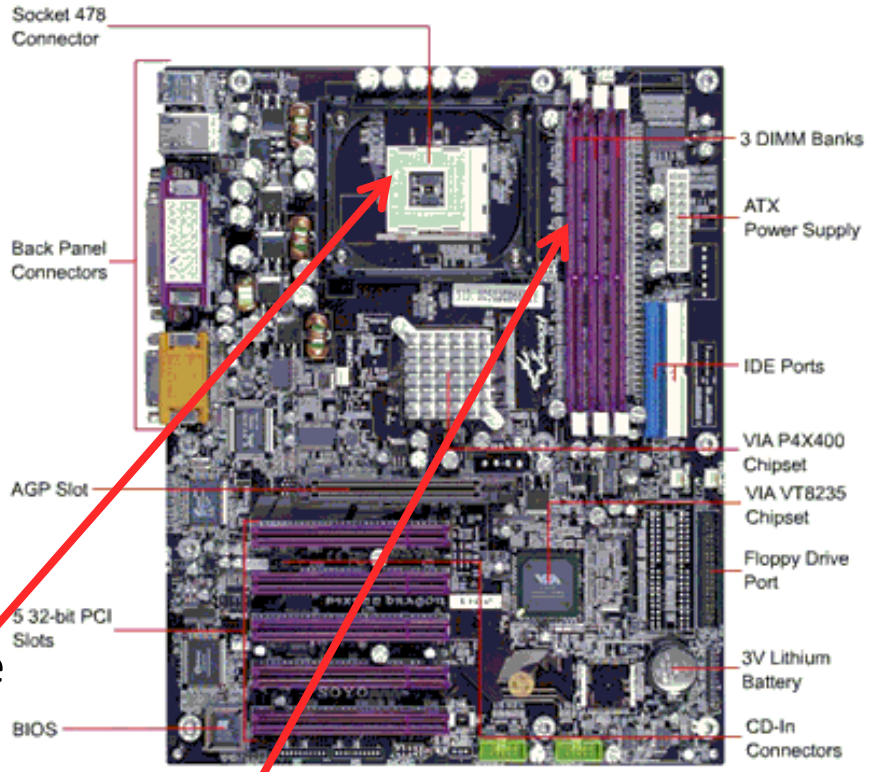
- But what if we run into a blockage?
  - Now we would pick the best among the remainder.

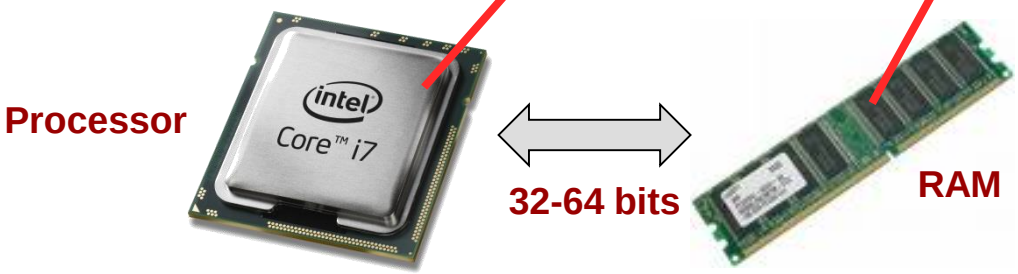


Select a square to explore with minimum distance to the finish

But Why?

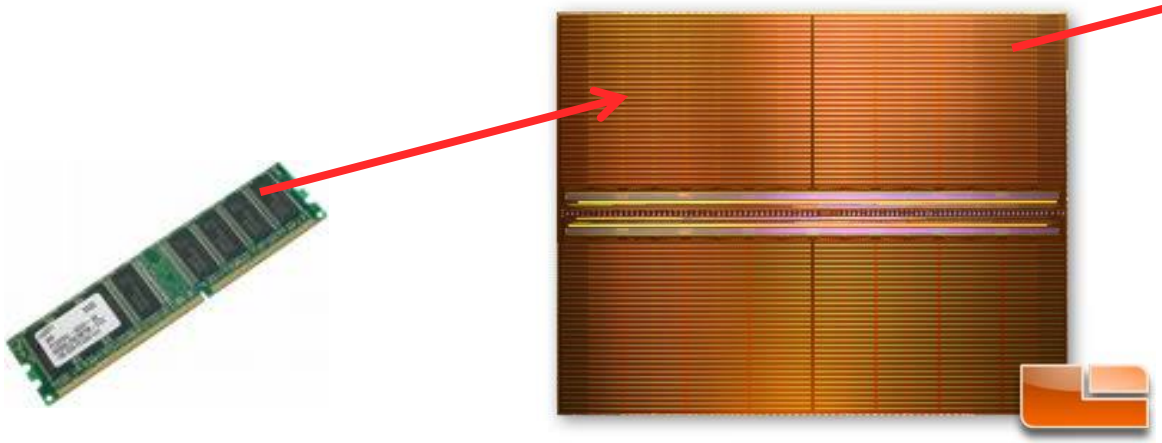
- Why can't computer just "look" at the image
  - Computer store information as numbers
  - These numbers are stored as units of 8-, 32- or 64-bits and the processor is only capable to looking at 1 or 2 numbers simultaneously
  - Each pixel of the image is a separate piece of data





Memory

- Set of cells that each store a group of bits (usually, 1 byte = 8 bits)
- Unique address assigned to each cell
  - Used to reference the value in that location

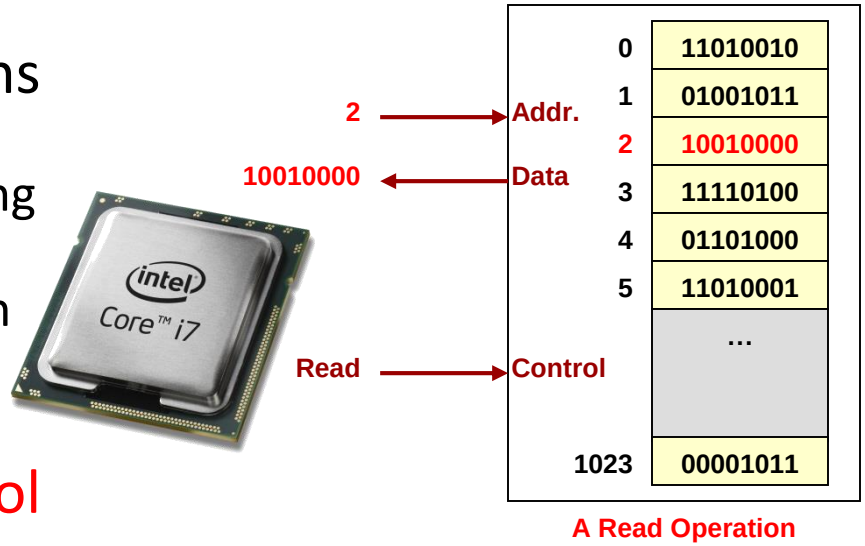


Address	Data
0	11010010
1	01001011
2	10010000
3	11110100
4	01101000
5	11010001
	…
1023	00001011

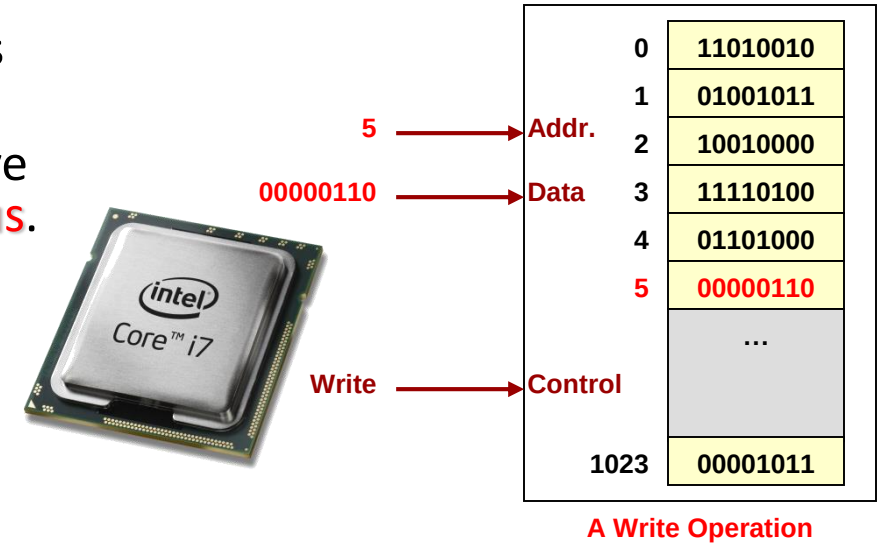
Memory Device

Memory Operations

- Memories perform 2 operations
  - Read: retrieves data value in a particular location (specified using the address)
  - Write: changes data in a location to a new value
- To perform these operations a set of address, data, and control inputs/outputs are used
  - Note: A group of wires/signals is referred to as a 'bus'
  - Thus, we say that memories have an address, data, and control bus.



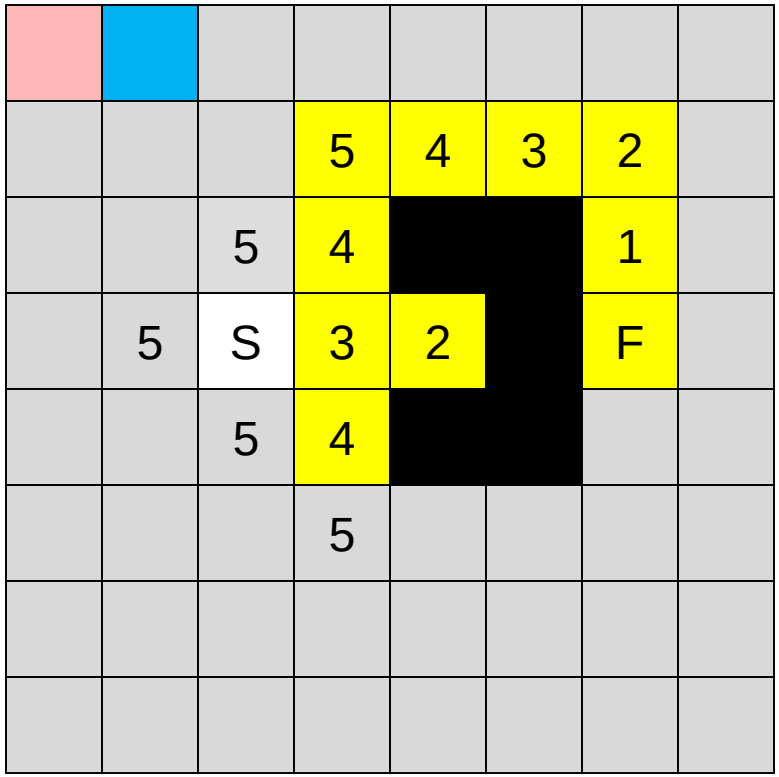
A Read Operation

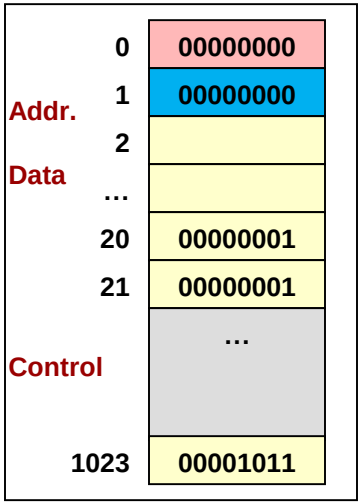


A Write Operation

Programming vs. Algorithms

- Programming entails converting an algorithm into a specific process that a computer can execute





20-Second Timeout: CS/CENG True or False

- Control Question: USC ~~basketball~~ baseball will win the NCAA championship this year
- True or False: The following achievements were performed here at USC in CS and EE depts.
  - Algorithmic basis of JPG, MPG, and MP3 formats developed here
  - A CS faculty won an Academy Award for Motion Pictures in 2010
  - THX audio was partly developed here
  - Network security has its roots in the research of a professor at USC

Another Example: Image Compression

- Images are just 2-D arrays (matrices) of numbers
- Each number corresponds to the color or a pixel in that location
- Image store those numbers in some way

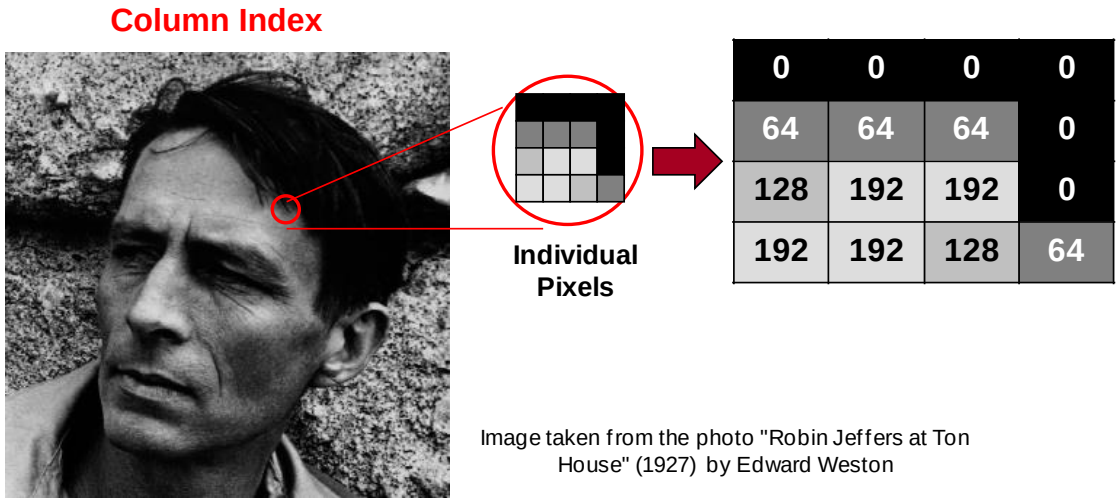


Image taken from the photo "Robin Jeffers at Ton House" (1927) by Edward Weston

Image Compression

129	131	130	133	132	132	130	129	128	130	131	129
130	130	131	129	131	132	131	133	130	129	129	131
132	131	130	132								
134	132	131	132								
133	131										
156	157										
153	155										
154	152										
207	204										
208	205										

Image Compression

129	131	130	133	132	132	130	129	128	130	131	129
130	130	131	129	131	132	131	133	130	129	129	131
132	131	130	132								
134	132	131	132								
133	131										
156	157										
153	155										
154	152										
207	204										
208	205										

1. Break Image into small blocks of pixels

129	131	130	133
130	130	131	129
132	131	130	132
134	132	131	132

129	2	1	4
2	1	2	0
3	2	1	3
5	3	2	3

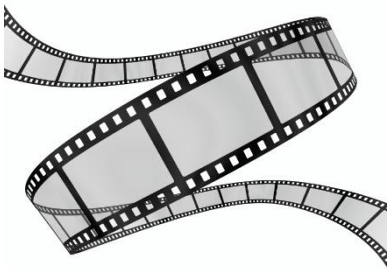
2. Store the difference of each pixel and the upper left (or some other representative pixel)

129	2	1	4
2	1	2	0
3	2	1	3
5	3	2	3

129	2	0	4
2	0	2	0
2	2	0	2
4	2	2	2

3. We can save more space by rounding numbers to a smaller set of options (i.e. only even # differences)

Video Compression



- Video is a sequence of still frames
  - 24-30 frames per second (fps)
- How much difference is expected between frames?
- Idea:
  - Store 1 of every K frames, with other K-1 frames being differences from frame 1 or from previous frame

Your Environment

GETTING STARTED

Development Environment

- To write and run software programs in C you will need
  - A text editor to write the code
  - A ‘C/C++’ compiler, linker, etc. to convert the code to a program
- Different OS and platform combinations have different compilers and produce “different version” of a program that can only run on that given OS/platform.
  - Mac XCode (Mac only)
  - MS Visual Studio (Windows only)
  - CodeBlocks (cross-platform)

Ubuntu VM Image

- We are providing a virtual machine appliance (An Ubuntu Linux image that you can run on your Mac or Windows PC)
  - Requires installation of Oracle VirtualBox and download of the Ubuntu Image
  - https://www.virtualbox.org/wiki/Downloads
  - Requires download of the VM Image from http://bytes.usc.edu/files/cs103/install/
- Video walkthrough
  - http://ee.usc.edu/~redekopp/Streaming/fa13_vm_walkthru/fa13_vm_walkthru.html

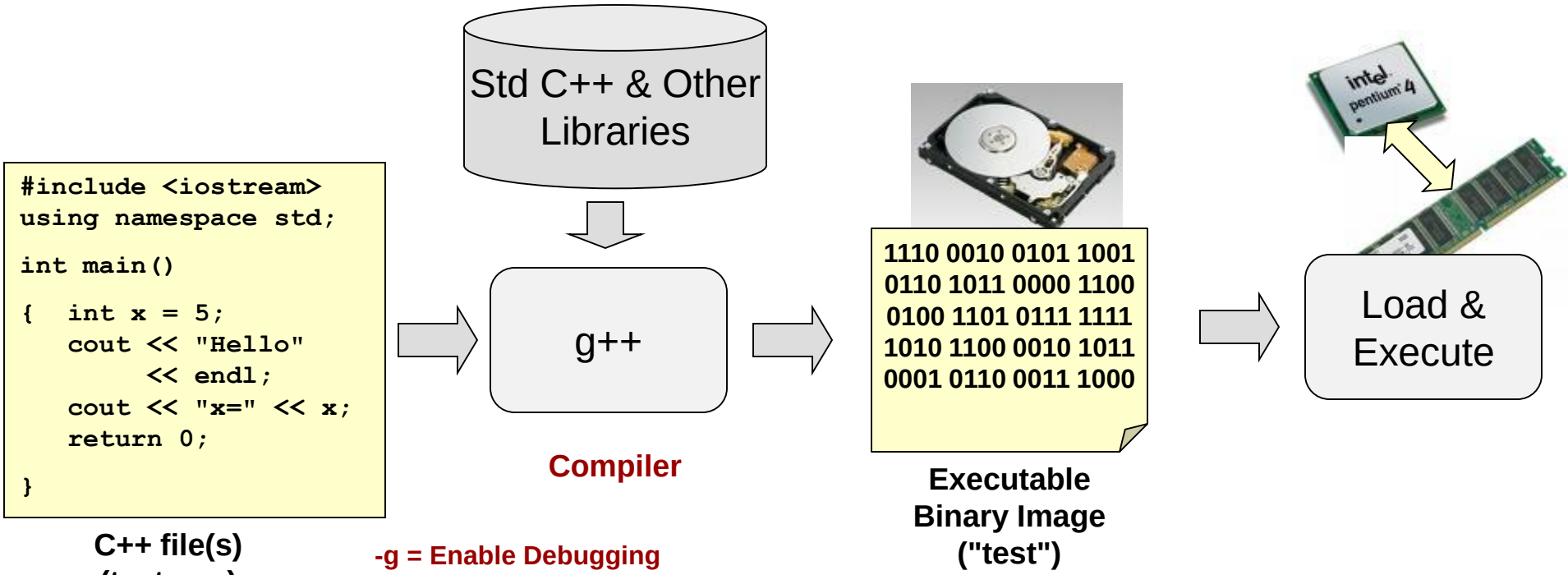
C OVERVIEW AND DEMO

C Program Format/Structure

- Comments
  - Anywhere in the code
  - C-Style => "/*" and "*/"
  - C++ Style => "//" (Single-line comments)
- Compiler Directives
  - #includes tell compiler what other library functions you plan on using
  - "using namespace std;" -- Just do it for now!
- Global variables (more on this later)
- main() function
  - Starting point of execution for the program
  - Variable declarations often appear at the start of a function
  - All code/statements in C must be inside a function
  - Statements execute one after the next
  - Ends with a 'return' statement
- Other functions

```
/* Anything between slash-star and
   star-slash is ignored even across
   multiple lines of text or code */
// Remainder of line after "//" is ignored

/*----Section 1: Compiler Directives ----*/
#include <iostream>
#include <cmath>
using namespace std;

/*------------ Section 2 ----------------*/
/*Global variables & Function Prototypes */
int x=5;
void other_unused_function();

/*----Section 3: Function Definitions ---*/
void other_unused_function()
{
  cout << "No one uses me!" << endl;
}

int main(int argc, char *argv[])
{  // anything inside these brackets is
   // part of the main function
  int y;    // a variable declaration stmt
  y = x+1; // an assignment stmt
  cout << y << endl; // print stmt
  return 0;
}
```

Software Process



-g = Enable Debugging
-Wall =Show all warnings
-o test = Specify Output executable name

```
$ gedit test.cpp &
```

1 Edit & write code

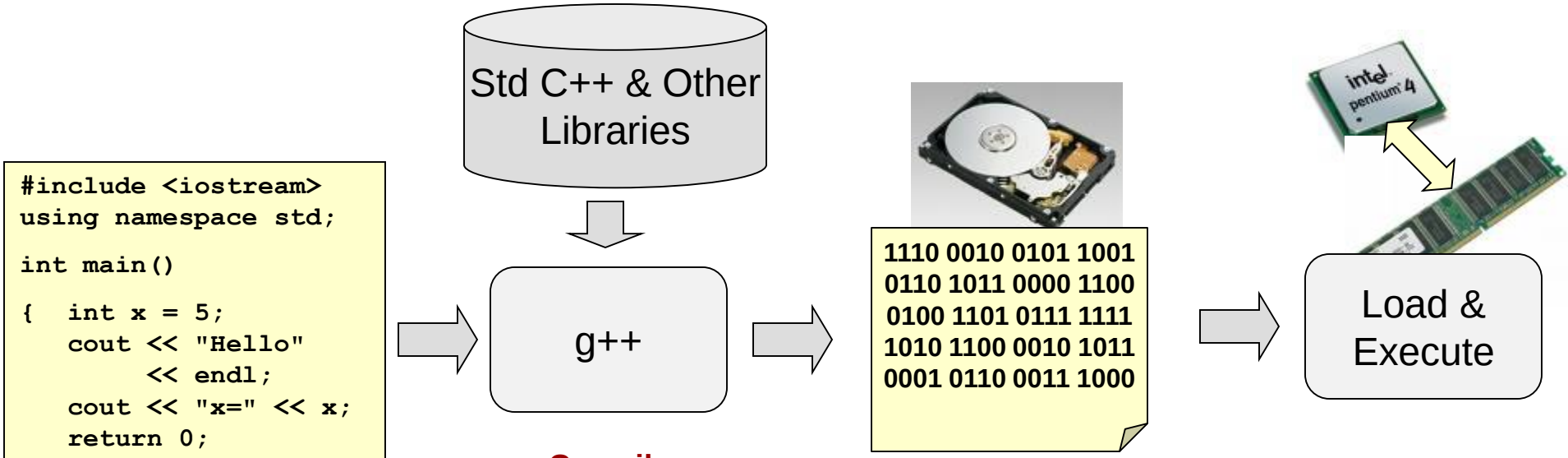
```
$ gedit test.cpp &
$ g++ -g -Wall -o test test.cpp
or
$ make test
```

2 Compile & fix compiler errors

```
$ gedit test.cpp &
$ g++ -g -Wall -o test test.cpp
$ ./test
```

3 Load & run the executable program

Software Process



C++ file(s) (test.cpp)

-g = Enable Debugging
-Wall =Show all warnings
-o test = Specify Output executable name

Executable Binary Image (test)

```
$ gedit test.cpp &
```

1 Edit & write code

```
$ gedit test.cpp &
$ g++ -g -Wall -o test test.cpp
or
$ make test
```

2 Compile & errors

Fix compile-time errors w/ a debugger

```
$ gedit test.cpp &
$ g++ -g -Wall -o test test.cpp
$ ./test
```

3 Load & run th executable program

Fix run-time errors w/ a debugger