Welcome to CSci 1113

Introduction to C/C++ Programming for Scientists and Engineers
Instructor (me)

James Parker
Lind 354

Primary contact:
jparker@cs.umn.edu
TAs

Zhiheng Deng, Jangyoon Kim
Apurva Badithela, Thomas Crumrine,
Abdelrahman Elfaramawy, Allie Funk,
Jon Garbe, Reid Kleinschmidt, James Kolar,
Justin Koo, Nikki Kyllonen, Christian Loven,
Max Marchionda, Timothy Nelson,
Joseph Rice, Reed Schelitzche, Gaurav Singh,
Liam Tyler, Mina Yacoup, Laura Ziegelski
Questions?

Direct questions to:
csci1113@cs.umn.edu
Moodle forum discussion
Problem Solving
With C++,
Walter Savitch,
9th edition
You need a CSELabs account to participate in labs in this course

Lab attendance is mandatory (please make an account!)
http://cselabs.umn.edu

College of Science and Engineering Public Computer Classrooms and Labs

The College of Science and Engineering operates several public computer labs that are reserved for CSE students. Students must open a CSE labs account in order to use these labs.

This site provides information regarding CSE lab account creation, access, and use. If you are unable to find what you are looking for on this site or have comments, suggestions, or questions, please send e-mail to operator@cselabs.umn.edu.

Door Access: The Mechanical Engineering building now has 24 hour access for CSE Labs students. If you have an active CSE Labs account, your U Card will now have access to the NIE building after hours. If you would like to test your U Card to ensure it works, please use the door in the far southwest corner of the courtyard between the old Mechanical Engineering building and the new Mechanical Engineering building. This is NOT the glass door next to the elevator.

Quick links for new students
- Account Information
- Account Authorization form
- Classroom and Lab Information
- Other Locations
http://cselabs.umn.edu
Welcome to the Fall2012 CSE Labs Account Creation Form.

Use this form to initiate or change your CSE Labs account for the Fall2012 semester. CSE Labs use is open to any student currently enrolled in the College of Science and Engineering.

Please enter the following information:

- Your student email **username**.
- Your **password** for your general UMN email account. (To verify your eligibility for a CSE Labs account.)

Username: park0580@umn.edu
Password: [redacted]

If you do not know what your username is, or you are having problems see the [U of M Student Internet Account Initiation Form](http://www.cs.umn.edu). For further information send email to operator@cselabs.umn.edu or stop by the Systems Staff Office in Keller Hall 1-213.

For a list of our hours see [Systems Staff Contact Information and Hours](http://www.cs.umn.edu).

Submit
CSE Labs account

CSE Labs account used in lab (first lab ensures account working)

Register ASAP

Problems?
Bug operator@cselabs.umn.edu
Class website

www.cs.umn.edu/academics/classes
Or google “umn.edu csci class”

Syllabus, schedule, other goodies

Moodle page will have grades and homework submissions
Class website

Moodle also has a link to the website:

Announcements
Moodle Resources and Self-Help Guides
Discussion/Questions

<----- Main webpage

Questions? Email: csci1113@cs.umn.edu

Office Hours
1113 Office Hours

<table>
<thead>
<tr>
<th>Sun 1/15</th>
<th>Mon 1/16</th>
<th>Tue 1/17</th>
<th>Wed 1/18</th>
<th>Thu 1/19</th>
<th>Fri 1/20</th>
<th>Sat 1/21</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6am</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7am</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CSci 1113: C++ Programming

Class Announcements

- 09/08/2015
  ALL YOUR BASE ARE BELONG TO US!
Syllabus

15% Labs
30% Homework (due Fridays)
   5% Quiz (Feb. 14)
10% Midterm 1 (Feb. 28)
15% Midterm 2 (April 11)
25% Final (May 9, 6:30-8:30pm)
Syllabus

Each week there will be either a homework due or a test.

Homework is due Fridays at 9:00 P.M. on Moodle.

Late homework is not accepted, but we will drop the lowest one.
Syllabus

Labs can be checked off up until a week after the lab

Homework must be coded individually

Don't cheat
Really... don't cheat
Homework will be both a creative and problem solving endeavor:

Lego example
Build a castle with:
- 4 walls enclosing
- Door
- At least one tower (higher than wall)
Homework
Syllabus

Grading scale:
93% A  77% C+
90% A-  73% C
87% B+  70% C-
83% B   67% D+
80% B-  60% D
Below F
Schedule

Ch. 1: Introduction, Programs, Compilers
Ch. 2: Input/Output, Data, Expressions
Ch. 3: Control Flow (if and loops)
Ch. 4, 5: Functions (return values)
Ch. 6: File I/O
Ch. 7, 8: Arrays and Strings
Ch. 9: Pointers and Dynamic Arrays
Ch. 10&11: Classes and Operator Overloading
Ch. 14&15: Recursion & Inheritance
Syllabus

Any questions?
What can I program?

If you can think of an explicit process (of simple steps) to solve your problem, then it can be programed.
Banana Nut Bread

Directions
1. Preheat the oven to 350°F (175°C).
2. Mix butter into the mashed bananas in a large mixing bowl.
3. Mix in the sugar, egg, and vanilla.
4. Sprinkle the baking soda and salt over the mixture and mix in.
5. Add the flour and nuts last, mix.
6. Pour mixture into a buttered 4x8 inch loaf pan.
Repetitive tasks
How do you get change for $18.26 with the least amount of bills and coins?
Repetitive tasks

If you feel like a mindless zombie when you do it a lot, you can probably program it.
Repetitive tasks
Repetitive tasks
Auto leveling?
Software vs Hardware

Software - the more intangible code on a computer

Hardware - the physical Parts of the computer
Hardware interaction

Input → CPU → Memory → Output
Memory addressing

Data is stored in “addresses” inside the memory

Later in this class, we will use these addresses to manipulate and share data
Memory addressing
Object oriented programming

OOP - focus on data and how they interact

To make algorithms for OOP, it is often useful to identify the data you are working with and their relationships before programming.
Object oriented programming

Data for...

Banana nut bread?
ATM?
Ball game?
Object oriented programming

Data for...

Banana nut bread? Ingredients
ATM?
Ball game?
Object oriented programming

Data for...

Banana nut bread? Ingredients
ATM? Dollars & coins
Ball game?
Object oriented programming

Data for...

Banana nut bread?  Ingredients
ATM?  Dollars & coins
Ball game?  Balls & mouse
Object oriented programming

Data for...

Banana nut bread?  Ingredients
ATM?  Dollars & coins
Ball game?  Balls & mouse

Lots of pixels (tiny color dots)
Break time!

How many programmers does it take to change a light bulb?

None. It's a hardware problem.
Object Oriented

Main focus is on objects and how they interact (represented by me as boxes)

Reusable groups of actions (verbs) between objects are called functions (squiggly boxes)

These actions can take additional information called arguments,
(an analogy is ordering at a restraunt; the ordering format is the same, different food)
Object Oriented

One format is:
object.function(argument, argument...);

Example:
James.teaches(CSci 1113);
teach(James, CSci 1113);

The dot (period) shows that “teaching” is an action done by “James”
Banana Nut Bread

Ingredients
* 3 or 4 ripe bananas, smashed
* 1/3 cup melted butter
* 1 cup sugar
* 1 egg, beaten
* 1 teaspoon vanilla
* 1 teaspoon baking soda
* Pinch of salt
* 1 1/2 cups of all-purpose flour
* 1 cup of nuts
Banana Nut Bread

Directions
1. Preheat the oven to 350°F (175°C).
2. Mix butter into the mashed bananas in a large mixing bowl.
3. Mix in the sugar, egg, and vanilla.
4. Sprinkle the baking soda and salt over the mixture and mix in.
5. Add the flour and nuts last, mix.
6. Pour mixture into a buttered 4x8 inch loaf pan.
Banana Nut Bread

Directions
1. Preheat the oven to 350°F (175°C).
2. Mix butter into the mashed bananas in a large mixing bowl.
3. Mix in the sugar, egg, and vanilla.
4. Sprinkle the baking soda and salt over the mixture and mix in.
5. Add the flour and nuts last, mix.
6. Pour mixture into a buttered 4x8 inch loaf pan.
Banana Nut Bread

Directions
1. Preheat the oven to 350°F (175°C).
2. Mix butter into the mashed bananas in a large mixing bowl.
3. Mix in the sugar, egg, and vanilla.
4. Sprinkle the baking soda and salt over the mixture and mix in.
5. Add the flour and nuts last, mix.
6. Pour mixture into a buttered 4x8 inch loaf pan.
Banana Nut Bread

Pseudo code directions

1. oven.preheat(350);
2. bowl.mix(butter, bananas);
3. bowl.mix(sugar, egg, vanilla);
4. bowl.sprinkle(baking soda, salt);
5. bowl.mix(flour, nuts);
6. bowl.pour(pan);
7. pan.bake(60);
8. pan.cool();
Banana Nut Bread

Pseudo code directions #2

1. oven.preheat(350);
2. bowl.add(butter, bananas);
3. bowl.mix();
4. bowl.add(sugar, egg, vanilla);
5. bowl.mix();
6. bowl.sprinkle(baking soda, salt);
7. bowl.add(flour, nuts);
8. bowl.mix();
9. pan.pour(bowl);
10. pan.bake(60);
11. pan.cool();
mashedBananas = bananas.mashed();
bowl.add(butter, mashedBananas);

same as:
bowl.add(butter, bananas.mashed());

Kitchen.bowl.add(butter, bananas.mashed());

hand.mix(butter, mashedBananas);
bowl.add(hand.mix(butter, mashedBananas));
Compiling

Converting **code** to binary is called **compiling**
Compiling

Often this compiled code will not work on other computers.
Compiling
C++ is a high level language (human readable)

Compiling changes a high level language into a low level language that is easier for the computer (computer cannot run high level)
Compiling

Your **source code** is the original language you wrote your program in (the C++ code for us)

You must **recompile** the **source code** every time you save a change before running the program again
Compiling

In labs, the computers will come with a program called “geany” (which I will use too)

This program is where you can write code and easily compile simple programs

To run it either click the terminal icon ( ] on the left bar or press Ctrl+Alt+T

Then type:   geany   (enter)
Compiling tl;dr
directions
cook
directions
cook
directions
cook

meal
code
compile
1's and 0's

eat
compile
run
pretty colors

satiated

DO NOT COOK in microwave ovens below 1100 watts as pot pie may not cook thoroughly. Conventional oven preparation is recommended.

3 LET STAND 3 MINUTES in microwave to complete cooking. CAREFULLY REMOVE as product will be hot.

4 CHECK that pot pie is cooked thoroughly.
- Internal temperature needs to reach 165° F as measured by a food thermometer in several spots.
- Crust is golden brown and steam rises from filling.

CONVENTIONAL OVEN

1. PREHEAT oven to 400°F. Place pot pie on cookie sheet, slit top crust.
2. BAKE in oven 33 to 34 minutes. CAREFULLY REMOVE in product will be hot.
3. LET STAND 5 MINUTES to complete cooking.

CHECK that pot pie is cooked thoroughly.
- Internal temperature needs to reach 165° F as measured by a food thermometer in several spots.
#include <iostream>
using namespace std;

int main ()
{
    cout << "Hello World! ";
    return 0;
}

(See: helloWorld.cpp)
MODEL SMALL
IDEAL
STACK 100H

DATASEG
MSG DB 'Hello, World!', 13, '$'

CODESEG
Start:
MOV AX, @data
MOV DS, AX
MOV DX, OFFSET MSG
MOV AH, 09H ; output ascii string
INT 21H
MOV AX, 4C00H
INT 21H
END Start
Ease of use
Why C++?

Speed

Control

Libraries
Speed

Not all programming languages need to compile code as C++ (Java, Python)

Compiling can greatly increase speed of a program
C++ allows you great control over your data (and its interpretation)

This comes with a burden of responsibility to properly manage your data

If you mismanage your data, you are likely to cause an error in your program
C++ is an old language (older than me) and this comes with pros and cons...

Some aspects are quirky to enable backwards compatibility (and are honestly out of date)

Since it has been around for a long time, there are lots of supporting libraries (and the language continues to develop...)
Java vs C++

Java
- Goes anywhere
- Comfy

C++
- Fast
- Fine tuned
Magic 8 ball
Magic 8 ball

What a rip off!
#include <iostream>
using namespace std;

int main()
{
    cout << "Maybe.";
    return 0;
}
Keyboard input

`cout << “word”`
- prints “word” to the screen

`cin >> x`
- store what is typed into “x”
  (x is some object or data)

Can also do arithmetic using +, -, / and *
(See: inputOutput.cpp)
Types of errors

Syntax error - code will not compile
e.g. `cout(“hi”);`

Runtime error - code crashes after starting
(see: runtimeError.cpp)

Logic error - code runs but doesn't return the correct answer
(see: logicError.cpp)
Syntax is a fancy word for the “grammar” of programming languages.

The basic English syntax is:
(subject) (verb) (noun)
“I eat bananas” not “Bananas I eat”

The computer is VERY picky (and stubborn) about grammar, and will not understand you unless you are absolutely correct!
Avoid errors

To remove your program of bugs, you should try to test your program on a wide range of inputs.

Typically it is useful to start with a small piece of code that works and build up rather than trying to program everything and then debug for hours.
Comments

Comments are ignored pieces of code (computer will pretend they do not exist)

// denotes a single line that is commented // (everything before hitting enter)

/* denotes the beginning of a comment and the end of a comment is denoted by */
Additional facts

Braces denote a block of code `{ ... }` (belonging to a method, class, etc.)

“White space” is ignored, just as your brain will ignore the bottom third of this slide (this is why we need a semi-colon)