Welcome to CSci 1113

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

James Parker
Shepherd Laboratories 391

Primary contact:
jparker@cs.umn.edu
TAs

Karthik Unnikrishnan, Prashanth Venkatesh, Jackson Benning, Yanjun Cui, Mitchell Dillon, Skye Gagnon, Jacob Hammer, Samuel Highbargin, Lin Huynh, Shane Jung, Jin Hong Kuan, Jan-Wei Lim, Haoran Liu, Ying Lu, Sophia Manicor, Andrew McCullough, Adam McCune, Kyle Meng, Brandon Nee, Tanner Skluzacek, Antonio Turley, Ruobing Wang, Kaiwei Wu, Yuyang Xiao, Songyu Yan, Lei Zhang, Xintong Zhang
Questions?

Direct questions to:
Canvas forum discussion
jparker@cs.umn.edu
Problem Solving With C++, Walter Savitch, 10th edition
Sister course: CSci 1115

This course is an “introduction” (from start), but many find it difficult.

We started to run a supplementary course to provide additional help: CSci 1115(Th 6pm)
Sister course: CSci 1115

This course is an “introduction” (from start), but many find it difficult.

We started to run a supplementary course to provide additional help: CSci 1115(Th 6pm)
Sister course: CSci 1115

This course is an “introduction” (from start), but many find it difficult

We started to run a supplementary course to provide additional help: CSci 1115(Th 6pm)

Daniel

Me
Sister course: CSci 1115

This course is an “introduction” (from start), but many find it difficult.

We started to run a supplementary course to provide additional help: CSci 1115(Th 6pm)

-group problem solving
Sister course: CSci 1115

This course is an “introduction” (from start), but many find it difficult.

We started to run a supplementary course to provide additional help: CSci 1115 (Th 6pm)

- group problem solving
- free food!
You need a CSE Labs account to participate in labs in this course.

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

Note: You must be logged in to the University of Minnesota to view these forms.

**General**

- Software or Hardware Assistance
- OS Configuration
- Laptop Wired Access Form
- Access Request
- Networking Request

**Printing**

- Paper and/or Toner Request
- Computer Science Poster Printing
- Earth Sciences Poster Printing

**CSE Labs**

- CSE Lab Room Reservation
- CSE Labs Account Creation

**Other**

- Service Suggestions
- Computer Science Labs UCard Access
CSE Labs Room Reservation

Use this form to request a CSE Labs room reservation.

After submitting this form, a confirmation e-mail will be sent from the CSE Labs Help Desk to your university e-mail address (unless reserving Lind 150). This confirmation will include a job number in the subject line. Please refer to this number in any future correspondence.

You should receive an approval or denial of your room request within 2 business days. If you have not received an answer after that time, please reply to the e-mail you received from the Help Desk (preserving the original subject line). If you submit a reservation request for the Taylor Center, you won't receive a submission confirmation from the Help Desk. In this case, if you haven't received a response in 2 business days, please send an email to csehelp@umn.edu explaining the situation.

For a CSE Labs Account, visit the CSE Labs Account Creation Page.
CSE Labs Account Creation

CSE Labs accounts no longer closing every term

If you have had a previous CSE Labs account, you do not need to reopen it every term. Accounts will now only be closed after a year of inactivity.

Welcome to the CSE Labs Account Creation Site

Use this site to initiate your CSE Labs account. CSE Labs use is open to any student currently enrolled in the College of Science and Engineering.

If you do not know what your username is, or you are having problems see the U of M Student Internet Account Initiation Form.

Create CSE Labs Account

For further information send email to operator@cselabs.umn.edu or stop by the Systems Staff Office in Keller Hall 1-201.

For a list of our hours see Systems Staff Contact Information and Hours.

Changing your Password

If you want to change your password, you will need to use the U of M Internet Account Options web page.
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.

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.

Submit
CSELabs account

CSELabs 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

Canvas page will have grades and (maybe) homework submissions
Class website

Canvas also has a link to the website:
## CSci 1113: C++ Programming

### Schedule

This is an approximate schedule. It will be updated as the class progresses.

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Lecture Materials (020)</th>
<th>Readings</th>
<th>Exams</th>
<th>Lab</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Jan 16</td>
<td>Introduction, computers, algorithms, programs, compilers</td>
<td>1/16</td>
<td>Ch. 1</td>
<td>Unix tutorial Remote connect (no lab this week)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Jan 23</td>
<td>Variables, expressions, assignment, console I/O, predefined functions</td>
<td>Ch. 2, Section 4.2</td>
<td>Lab 1: Basic C++ programs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Jan 30</td>
<td>Selection, boolean expressions, if-else, multiway-if, switch</td>
<td>Sections 3.1, 3.2</td>
<td>Lab 2: Sequence and Selection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Feb 6</td>
<td>Iteration, while loops, for loops, loop paradigms</td>
<td>Sections 3.3, 3.4</td>
<td>Lab 3: Iteration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Feb 13</td>
<td>User-defined functions, procedural abstractions</td>
<td>Ch. 4, 5</td>
<td>Quiz Covers Ch 1-3.2 (up to week 3: if-statements)</td>
<td>Lab 4: User defined functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Feb 20</td>
<td>Basic file I/O</td>
<td>Ch. 6</td>
<td>Lab 5: Reference parameters and basic file I/O</td>
<td>HW 3. Thursday Feb 22 at 11:55 P.M.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Feb</td>
<td></td>
<td>Ch. 7</td>
<td>Midterm 1, Covers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Additional Resources

- [CSci 1113: C++ Programming](http://www.users.cselabs.umn.edu/classes/Spring-2018/csci1113-night/)

---

**University of Minnesota**

**College of Science & Engineering**

[Home](#)  | [Office Hours](#)  | [Syllabus](#)  | [Moodle (grades and hw submission)](#)

---

**CSci 1113: C++ Programming**
15% Labs
30% Homework (due Fridays)
   5% Quiz (Feb. 18)
10% Midterm 1 (March 3)
15% Midterm 2 (April 14)
25% Final (May 12, 6:30-8:30pm)
Each week there will be either a homework due or a test.

Homework is due Fridays at 11:55 P.M. (more details to come)

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 (warm-up questions must be in your lab)

Homework must be done by yourself

Don't cheat
Really... don't cheat
Homework

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
Exams

All exams will be open book/notes
Electronic notes okay
(no memorization)

You **cannot**:
1. Use the internet (no typing)
2. Compile/run programs
3. Talk to or copy from others
## Syllabus

### Grading scale:

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>93%</td>
<td>A</td>
</tr>
<tr>
<td>90%</td>
<td>A-</td>
</tr>
<tr>
<td>87%</td>
<td>B+</td>
</tr>
<tr>
<td>83%</td>
<td>B</td>
</tr>
<tr>
<td>80%</td>
<td>B-</td>
</tr>
<tr>
<td>77%</td>
<td>C+</td>
</tr>
<tr>
<td>73%</td>
<td>C</td>
</tr>
<tr>
<td>70%</td>
<td>C-</td>
</tr>
<tr>
<td>67%</td>
<td>D+</td>
</tr>
<tr>
<td>60%</td>
<td>D</td>
</tr>
<tr>
<td>Below 50%</td>
<td>F</td>
</tr>
</tbody>
</table>
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
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
ATMs

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 restaurant; 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**.
7. **Bake** for 1 hour. **Cool** on a **rack**.
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 tl;dr

directions

cook

meal

eat

satiated

code

compile

1's and 0's (program)

run

pretty colors
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)
#include <iostream>
using namespace std;

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

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!
Magic 8 ball

```cpp
#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

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 the your brain will ignore the bottom third of this slide (this is why we need a semi-colon)