CSci 123 – Intro. to Programming in C++

What?

This course is an introduction to programming in C++. We'll learn about statements; types, variables, and expressions; control structures; classes and object-oriented design; and the C++ standard library.

Where is...

Course website?	http://staffwww.fullcoll.edu/aclifton/courses/cs123/
Syllabus?	*on the website or Canvas*
Grades?	*on Canvas*

Who?

Professor	Andy Clifton
Office hours	Mon,Wed 10:00 - 11:35 AM Thurs 12:45 - 3:00 PM
Office	611-02
Email	aclifton@fullcoll.edu
Campus phone	714-992-7418

The FCCsci server

Hostname	fccsci.fullcoll.edu
Port	5150
Username	First initial, followed by last name
Password	Banner ID, without the @

Assignments and projects are completed and submitted on the server.

- On Windows, download and connect using PuTTY with the above details.
- On a Mac, connect using SSH in the Terminal app: `ssh username@fccsci.fullcoll.edu -p 5150`
- The computers in the CSci lab already have PuTTY installed.

Assignments

Posted to the website (almost) every Sun. or Mon, (usually) due the following Monday. Download them on the server with `do-assignment`, edit with `micro`. You don't need to do anything to submit; just make sure you are saving as you work. Graded on effort, not correctness.

Projects

Complete two ('C'), three ('B') , or all four ('A') stages of an exciting project!

Midterms

Four midterms, each covers three *new* modules, plus any modules from the previous midterms. You only need to work modules you haven't already passed.

Final

Final exam is comprehensive, averaged (rounding up) with the grade for the rest of the class.

Grading

I used *specifications grading* which means that your grade is based on the amount of material you can *prove* you have mastered.

Pick the grade you want in the first row of the table, and then read *down* to see what you need to do to get it:

To earn an:	**A**	**B**	**C**
Assignments	90%	80%	70%
Midterm topics	All topics	Core + 3 adv.	Core
Project stages	All four	1,2,3	1,2

You can pick any three of the advanced modules to earn a B. Your grade on the final exam is averaged with the grade from this table, and then rounded up.

A simple C++ program

For the first few weeks, all our programs will look something like this:

```
#include <iostream>
using namespace std;

int main() {

    ...
}
```

with the details of your particular program replacing the "..." in the middle.

Reference

Shell commands

Command	Description
`cd` *path*	Change directory
`cd ~`	Change to home
`ls`	List current dir.
`tree`	Show dir. tree
`mkdir` *path*	Make directory
`cp` *file1 file2*	Copy files
`mv` *file1 file2*	Rename files
`rm` *path*	Delete file (permanent!)
`compile` *file.cpp*	Compile program
`g++` *file.cpp*	Manually compile to `.o`
`g++ -o` *prog* `f1.o f2.o`	Link `f1`, `f2` into *prog*
`./`*program*	Run *program*
`do-assignment` *num*	Download assignment
`micro` *file*	Edit *file*
`micro` *f1 f2 …*	Edit multiple files (tabs)

Micro keyboard commands

Command	Description
Ctrl-g	Display help
Ctrl-s	Save
Ctrl-q	Quit
Ctrl-o *file*	Open file
Ctrl-t	New tab
select with mouse	Select text
Ctrl-c	Copy
Ctrl-v	Paste
Ctrl-x	Cut
Ctrl-z	Undo
Ctrl-b *cmd*	Run shell command
Ctrl-e `vsplit`	Split vertically (side-by-side)
Ctrl-e `hsplit`	Split horizontally (top/bottom)
Ctrl-e `term`	Open shell terminal
Ctrl-l *num*	Go to line number *num*

When you run `do-assignment` *num*, blank files for all of the worked examples and problems will open in Micro, as tabs across the top of the window. Click on a tab to switch to that file. A good workflow for writing programs is

1. Edit a file (e.g., `file.cpp`)
2. Save (Ctrl-s). Any errors will be highlighted in the left margin.
3. If there are no errors, press Ctrl-b and type `compile file.cpp` and press Enter
4. Press Ctrl-b and type `./file` to run your program and see if it works. If it doesn't, go back to step (1).

C++ Syntax

Syntax	Description
`/* … */`	Block comment
`// …`	Single-line comment
`#include <iostream>`	Include directive
`#pragma once`	Pragma once dir.
`int`, `char`, `bool`, `float`, `double`	Built-in types
`string`	String type
`vector<`*T*`>`	Vector-of-*T*s type
T`&`	Reference to type *T*
T`*`	Pointer to type *T*
`using namespace std;`	using-declaration
type name`;`	Variable declaration
type name `=` *expr*`;`	Variable decl.
type $name_1$`,` $name_2$`, …;`	Variable decl.
type name`(`*params…*`);`	Function decl.
`class` *name*`;`	Class decl.

type name`(`*params…*`) {`
  `…`
`}`
	} Function def.

`class` *name* `{`	Class def.

`public:`
`private:`
`protected:`
	} Access spec.

name`(`*params…*`) { … }`	Constructor
`~`*name*`() { … }`	Destructor
type name`(`*params…*`) { … }`	Function member
type name`;`	Data member
`};`	(end class def.)

Include files

`iostream`	Terminal input/output (`cin`, `cout`, etc.)
`string`	The `string` type
`vector`	The `vector` type
`fstream`	File input/output
`stdexcept`	Standard exception types
`cmath`	Math functions (`sin`, `log`, etc.)
`cassert`	Assertions (`assert`)
`cctype`	Classification of `char`'s (letter, number, etc.)

String escapes

`\n`	Newline (`endl`)
`\t`	Tab
`\"`	Double-quote
`\'`	Single-quote
`\b`	Backspace (may not work)
`\a`	"Alert" (flashes window)
`\x`*nn*	Any character code *nn*