CSC 200: Problem Solving for Science and Engineering

Unit Readings and Assignments

Instructor: Victoria C. Chávez

Last modified on: 11/15/2020

Unit 0: Intro and setup

Read/Watch/Review

- Read: the syllabus

Assignment

Description

- Choose one of the following online IDEs to work with for the semester:
    1. Repl.it
    2. GDB
    3. Ideone
    4. Something else if you're comfortable with it
- Create an account on the IDE of your choice, sign in, start a new program or file (no need to write anything, you can leave it as is)
    1. Make sure the language is set to C++
    2. Run your program or file
    3. Take a screenshot
- Upload and submit your screenshot
    1. Make sure we can see the whole screen

Overview: What to submit

- Screenshot of your online IDE while you are signed in and have run the default program

Requirements

1. Screenshot shows user logged in
2. Screenshot shows code was executed successfully

Unit 1: Basics of Programming and C++

Essential Questions

- What is an algorithm?
- What is pseudocode?
- What is a syntax error?
- What is a C++ program made up of?
- What is the purpose of cout and cin?
- What are some common escape sequences?
- How do variables work in C++?
- What are the different data types in C++ and when should each be used?

Read/Watch/Review

Intro to Programming

- Read: 1.3-1.5
- Watch: Computer Science is Changing Everything [6 min]
- Watch: How Computers Work: Hardware and Software [6 min]

C++ Input and Output

- Read: 2.1-2.3
- Read: 3.1
- Watch: C++ Program Input and Output [10 min]

C++ Data Types and Variables

- Read: 2.5-2.15
- Watch: First C++ Program [12 min]
- Watch: Variables, Assignment & Data Types [11 min]
- Optional Watch: (more in-depth video on data types): Primitive Data Types [17 min]
- Watch: Introduction to Strings in C++ [6 min]

Best Programming Practices

- Read: 1.6
- Read: 2.16-2.17
- Watch: Devising an Algorithm C++ Program, Part 1 [15 min]
- Watch: Devising an Algorithm C++ Program - Part 2 [13 min]

Lab 1: Programming Exercises

Part 1 Description

The program in lab1p1.cpp will not compile because the lines have been mixed up. When the lines are properly arranged the program should display the following output: `Dogs are awesome.`

1. Start a new program on your editor of choice
2. Download lab1p1.cpp and either upload it to your editor, or copy and paste the code

    a. To copy and paste it, right click on the downloaded file and select "Open with." Windows users choose "NotePad" and Mac users choose "TextEdit"
3. Rearrange the lines in the correct order
4. Test the program by compiling it and running it.
5. Download the corrected code and make sure to **save it as lab1p1.cpp.**

Part 2 Description

1. Start a new program on your editor of choice
2. Download lab1p2.cpp and either upload it to your editor, or copy and paste the code
    a. To copy and paste it, right click on the downloaded file and select "Open with." Windows users choose "NotePad" and Mac users choose "TextEdit"
3. Add the necessary code but **do not change, remove, or otherwise modify** any of the code I've given you to swap the values entered by the user.
4. Test the program by compiling it and running it.
5. Download the corrected code and make sure to **save it as lab1p2.cpp.**

Part 3 Description

1. Start a new program on your editor of choice
2. Write a program that calculates the average monthly rainfall for three months.
    a. The program should ask the user to enter the name of each month and the amount of rain (in inches) that fell that month.
    b. The program should display a message saying how much the average rainfall for the given three months was.
    c. For example, if I enter the months "June" "July" and "August" with average rainfalls of "3.3", "6.43", and "2", your program should output something like:
    ```
    The average monthly rainfall for June, July, and
    August was 3.91 inches.
    ```
3. Test your program by compiling it and running it.
4. Download and **save your code as lab1p3.cpp.**

Overview: What to submit

- You should have 3 files: lab1p1.cpp, lab1p2.cpp, lab1p3.cpp
- Upload your three files

Requirements

1. Part 1 compiles and runs with no errors
2. Part 1 returns expected output
3. Part 2 compiles and runs with no errors
4. Part 2 returns expected output
5. Part 3 compiles and runs with no errors
6. Part 3 returns expected output

Unit 2: Expressions and Interactivity

Essential Questions

Programming Practices

- What's the purpose of the Problem Solving recipe?
- What is the purpose of memory diagrams?
- How do memory diagrams differentiate variable creation/instantiation vs variable updating?

Expressions

- What is an expression?
- What purpose do operators and operands serve?
- What's the difference between a unary and a binary operator?
- What's the purpose of the C++ library?
- What's purpose does implicit type conversion/type coercion serve?
- When and why should type casting be used?
- Why can overflows and underflows be dangerous?
- When should the const keyword be use?
- When should multiple assignment be used?
- What are compound operators and what purpose do they serve?

Interactivity

- What's the purpose of formatting?
- What are some common stream manipulators?
- How do stream manipulators differ from escape sequences?
- How should user's input be Read: if want:
    - A word
    - A sentence
    - A letter
    - A single whitespace character
- What is a buffer overrun and how should it be avoided?
- What is the purpose of a seed?

Read/Watch/Review

Programming Practices

- Read: Problem-Solving Recipe
- Read: Memory Diagrams - A Tool For Code Tracing [Variables]
- Watch: memDiagram Variables [5 min]

Expressions

- Read: 3.2-3.7
- Watch: C++ Arithmetic Operators [6 min]
- Watch: C++ Increment and Decrement Operators [8 min]
- Watch: C++ Modulus, Short-Hand Operators [5 min]
- Watch: Casting in C++ [5 min]
- Watch: Overflow and Underflow in C++ Tutorial [2 min]
- Watch: Ethical Hacking: Buffer Overflow Basics [4 min]

Interactivity

- Read: 3.8-3.11
- (Re)Watch: Introduction to Strings in C++ [6 min]
- Watch: Formatting Output - fixed, showpoint, setprecision [5 min]
- Watch: Formatting Output - setw [3 min]
- Watch: C++ Extraction Operator, getline and cin.ignore() [12 min]
- Watch: Random Numbers in C++ [playlist, 20 min]

Lab 2 Assignment

Part 1 Description

- Modify the program in smiley.cpp so that it uses **no whitespace characters** while still printing out the same smiley face.
- Leave both the original code and your modified code in the program so the smiley is printed it out twice: once by the given code and another by your code.
    1. Note: You'll need an escape sequence to print out the \ of the mouth but that should be **the only escape sequence** in the second version of your code.
    2. Hint: While you could do this through trial and error, having a clear understanding of **formatting** will make your task much easier.
- Your output should look as so:

```
^   ^
  *
\___/

^   ^
  *
\___/
```

- **Download and save your program as lab2p1.cpp**

Part 2 Description

- Use a memory diagram to trace the two programs below.
- Take a picture or screenshot of your memory diagrams.

Program 1 (assume the user enters `54321`):

```
int main() {

    double salary, monthly;

    cout << "What is your annual salary? ";
    cin  >> salary;
    monthly = static_cast<int>(salary) / 12;
    cout << "Your monthly wages are " << monthly << endl;
    return 0;
}
```

Program 2 (assume the user enters your name):

```
int main() {

    string userInput;

    cout << "What is your name? ";
    cin  >> userInput;
    cout << "Hello " << userInput << endl;
    return 0;

}
```

Overview: What to submit

- Submit the following 3 files:
    - lab2p1.cpp
    - A picture of the memory diagram for program 1
    - A picture of the memory diagram for program 2

Requirements

1. Part 1 compiles and runs with no errors
2. Part 1 returns expected output, following description
3. Memory diagram for Program 1 is correct
4. Memory diagram for Program 2 is correct

Unit 3: Making Decisions

Essential Questions

- What purpose do relational operators serve?
- Why can characters be compared using relational operators?
- What's the relationship between relational and boolean expressions?
- What purpose does the **if** statement serve?
- What's the difference between a sequence structure and a decision structure?
- Why is it important that C++ considers any non-zero value true?
- What purpose do flags serve?
- How should equality of floating point values be checked?
- What purpose does the **if/else** statement serve?
- What purpose does the **if/else if** statement serve?
- What purpose does a **trailing else** statement serve?
- What purpose does a menu-driven program serve?
- How do nested if and nested if/else statements work?
- What purpose do logical operators serve?
- How do each of &&, ||, and ! evaluate with different inputs? (Hint: create a table)
- Why is input validation important?
- Why should we care about the scope of a variable?
- What purpose do switch statements serve?

Read/Watch/Review

- Read: Gaddis 4.1-4.10
- Watch: C++ Relational and Boolean Operators [10 min]
- Watch: if Statements in C++ [8 min]
- Watch: if/else Statements in C++ [4 min]
- Optional Watch: (more if/else practice): C++ Conditionals [playlist, 32 min]
- Read: Gaddis 4.12
- Watch: Switch Statements in C++ [7 min]

Lab 3: Assignment

Part 1 Description

- Create a memory diagram for each of the 3 programs below
- Remember to include the outputs (whatever gets printed)
- Answer the questions about the programs below:
    1. What value would x have to be in **Program 1** for “X is big” to print out?
    2. What value would x have to be in **Program 2** for “X is 5 or less” to print out?
    3. What value would x have to be in **Program 3** for nothing to print out?

Program 1:
```
int x = 5;

if (x < 10) {
	cout << “X is less than 10” << endl;
}

if (x <=5) {
	cout << “X is 5 or less” << endl;
}

cout << “X is big” << endl;
```

Program 2:
```
int x = 5;

if (x < 10) {
	cout << “X is less than 10” << endl;
} else if (x <= 5) {
	cout << “X is 5 or less” << endl;
} else {
	cout << “X is big” << endl;
}
```

Program 3:
```
int x = 5;

if (x <= 5) {
	cout << “X is 5 or less” << endl;
} else if (x < 10) {
	cout << “X is less than 10” << endl;
} else {
	cout << “X is big” << endl;
}
```

Part 2 Description

- Follow one of the following two storylines to create a “Choose Your Own Adventure” program: two friends hanging out or a day in the life of Garfield
- **Save your program as “cyoa.cpp”**

Overview: What to submit

- Pictures of all three memory diagrams
- Answers to three questions from Part 1
- cyoa.cpp

Requirements

1. Memory diagram for Program 1 is correct (including output box)
2. Memory diagram for Program 2 is correct (including output box)
3. Memory diagram for Program 3 is correct (including output box)
4. Answers to all 3 questions are correct
5. cyoa.cpp runs, builds, and gives the expected output-- per storyline followed

Unit 4: Loops

Essential Questions

- What purpose do ++ and -- serve?
- What's the difference between prefix and postfix modes for the ++ and -- operators?
- What's the purpose of a loop?
- What's an infinite loop and how do we avoid them?
- What does it mean for a loop to be pretest?
- What does it mean for a loop to be posttest?
- How do while loops work?
- How can we perform input validation with a while loop?
- How do do-while loops work?
- How can do-while loops be used for user interaction?
- How do while and do-while loops differ?
- How do for loops work?
- How do for and while loops differ?
- How can for loops be used for user interaction?
- What's the purpose of an accumulator?
- What's the purpose of a sentinel?
- How can sentinels be used for user interaction?
- How do nested loops work?
- What's the purpose of the keyword break?
- What's the purpose of the keyword continue?
- Why should using break and continue be avoided when working with loops?
- When should one use a while vs a do-while vs a for loop?

Read/Watch/Review

- Read: Gaddis 5.1
- (Re)Watch: C++ Increment and Decrement Operators [8 min]
- Read: Gaddis 5.2
- Watch: C++ while Loop [6 min]
- Read: Gaddis 5.3 – 5.4
- Watch: C++ while Loop Types [13 min]
- Watch: C++ While Loop Input Validation [8 min]
- Read: Gaddis 5.5
- Watch: C++ do while Loop [4 min]
- Read: Gaddis 5.6 - 5.8 and 5.10 – 5.13
- Watch: Repetition/Looping Structures in C++ [42 min]

- Read: Memory Diagrams - A Tool For Code Tracing [Loops]
- Watch: memoryDiagram For Loops [2 min]

Lab 4 Assignment

Part 1 Description

- Write a program that generates a **x** random numbers, **inclusive**, between **low** and **high**
- Your program should be completely user-led (that is, the user decides the range and how many random numbers to generate)
- Download and save your program as **lab4Rand.cpp**

Part 2 Description

- Modify your cyoa.cpp to validate user input and allow replays:
    1. Your program should not crash or end if a user inputs incorrect input, regardless of how many times they do it
    2. Your program should allow the user to restart the game at the end (if they want to)
- Download and save your program as **cyoaUpdated.cpp**

Part 3 Description

- Write the pseudocode for a program that checks whether a string is a palindrome (a palindrome is a word that is the same backward and forward, such as mom and racecar)

Overview: What to submit

- lab4Rand.cpp
- cyoaUpdated.cpp
- Pseudocode for program from Part 3

Requirements

1. Part 1 compiles, runs, and returns expected output
2. Part 2 compiles, runs, and returns expected output
3. Part 3 correct

Unit 5: Functions

Essential Questions

- What is modular programming?
- What is a function and what purpose does it serve?
- What does the divide and conquer approach entail?
- What's the difference between a function definition and a function call?
- What's a parameter and what purpose does it serve?
- What purpose do void functions serve?
- What does it mean for a function to be called in a hierarchical or layered fashion?
- What's the relationship between an argument and a parameter?
- What does it mean to pass data by value?
- What purpose does a return statement serve?
- What purpose do value-returning functions serve?
- What types of data may a function return?
- How can functions help with user interaction?
- What's the difference between local and global variables?
- Why should global variables be avoided?
- What purpose do global constants serve?
- What purpose do default arguments serve?
- ~~What does it mean to pass data by reference?~~
- ~~When should reference variables be used?~~
- What is function overloading and why is it useful?

Read/Watch/Review

- Read: Gaddis 6.1
- Watch: Introduction To Structured Programming [6 min]
- Read: Gaddis 6.2-6.3
- Watch: C++ Void Functions [9 min]
- Read: Gaddis 6.4 - 6.5
- Watch: C++ - Passing Parameters to a Function [12 min]
- Watch: C++ - Passing Data to a Function [11 min]
- Read: Gaddis 6.6 – 6.8
- Watch: C++ - Value-Returning Functions [11 min]
- Read: Gaddis 6.9
- Watch: Menu Driven Program with Functions - Revised [4 min]
- Read: 6.10 - 6.11
- Watch: C++ - Scope [18 min]
- Read: Gaddis 6.12

- Watch: Default Arguments / Parameters [6 min]
- Read: Gaddis 6.14
- Watch: Function Overloading [6 min]

Lab 5: Assignment

Description

- We will be solving **a modified** Bridge Problem using modular programming (i.e., divide and conquer):
    1. Your program should ask the user for how long they want the bridge to be (in feet)
    2. Your program should ask the user how many **5ft boards** they have
    3. Your program should ask the user how many **1ft boards** they have
    4. Tell the user whether it is possible to build the bridge given the boards they have.
        - The bridge needs to be EXACT size (can't cut boards)
    5. Tell the user how many of each board they will need.
    6. Hint: assume you use large boards first
    7. This is more logic and less programming (figure out the mathematical equation first)
    8. You should do this without finding each possible board combination.
    9. No need for loops
- Write a comment, for each function, telling us what you expect the inputs and outputs to be (i.e., what does this function do?)
- Use layered function calls, where appropriate, to achieve desired results (this means you shouldn't be copy/pasting or rewriting any of your code.
- There are many ways to approach this-- some better than others. Think about how to break the problem down into smaller components. Each function you write should only be doing one thing.
- Test your program
    1. Call all your functions in your program's `main()` function, passing in arguments to test the output. See class slide on testing.
- Download and save your program as **bridgeFunctions.cpp**

Overview: What to submit

- bridgeFunctions.cpp

Requirements

1. Modular programming guidelines followed (i.e., each function has only one job)

2. No repeated code (i.e., correct use of layered function calls)
3. Each function has a comment describing its functionality
4. Each function created was tested in `main()`, as instructed in slides/in class, with various arguments, to test for correct and expected behaviour and output

Unit 6: Arrays and Vectors

Essential Questions

Arrays

- What purpose does an array serve?
- How does one instantiate an array?
- What does it mean for arrays to be zero-indexed?
- How are elements of an array assigned?
- How are elements of an array accessed?
- What is the danger of trying to access an array beyond its bounds?
- How does one initialize an array?
- How should we compare arrays?
- What are two ways to access all the elements of an array?
- How can arrays be passed as arguments to functions?
- What does it mean to pass data by reference?

2D Arrays

- What is a two-dimensional array and what purpose does it serve?
- How does one instantiate a two-dimensional array?
- How does one initialize a two-dimensional array?
- How can two-dimensional arrays be passed as arguments to functions?
- How does one access any given element in a two-dimensional array?
- How does one access all the elements of a two-dimensional array?

Vectors

- How do vectors differ from arrays?
- How are elements removed from a vector?
- How are elements added to a vector?

Read/Watch/Review

- Read: Gaddis 6.13
- Watch: C++ Value and reference parameters [2 min]
- Read: Gaddis 8.1-8.6
- Watch: C++ Programming: Array Traversals [10 min]
- Read: Gaddis 8.8-8.9
- Watch: C++ Pass an Array to a Function [4 min]
- Read: Gaddis 8.11
- Watch: Vectors in C++ [16 min]

Lab 6: Assignment

Description

- Write a function `getMax()` that has an **array of integers and the array's size as its parameters**. Your function should **return the largest number** in the array. You must **manually calculate** the maximum of the array.
- Write a function `getMin()` that has an **array of integers and the array's size as its parameters**. Your function should **return the smallest number** in the array. You must **manually calculate** the minimum of the array.
- Write a function `getTotal()` that has a **vector of integers** as its parameter. Your function should **return the sum** of all the values in the vector. You must **manually calculate** the sum.
- Write a function `getAverage()` that has **a vector of integers** as its parameter. Your function should **return the average** of all the values in the vector. You must **manually calculate** the average using a function call to `getTotal()`.
- Call all your functions in your program's `main()` function, passing in arguments to test the output. See class slide on testing.
- In your `main()` function, write a **menu-driven program** with the following options:
    1. Get the minimum of all the inputted numbers
    2. Get the maximum of all the inputted numbers
    3. Get the sum of all the inputted numbers
    4. Get the average of all the inputted numbers
- For options 1 and 2, the user should tell you how many numbers they're going to input before actually inputting the numbers themselves.
- For options 3 and 4, the user should be able to input as many numbers as they'd like until they are done (that is, they won't tell you in advance how many numbers they're inputting)
- Your program should print the correct output for each option

Overview: What to submit

- A single .cpp file with all 4 functions and a working main()

Requirements

1. Each function has the correct input/output contract (parameters and function-type)
2. Each function has a comment describing its functionality
3. Each function created was tested in `main()`, as instructed in slides/in class, with various arguments, to test for correct and expected behaviour and output
4. Menu-driven program behaves as expected

Unit 7: Classes and Objects

Essential Questions

- What is a class?
- How do you create your own class?
- What are access specifiers and what purpose do they serve?
- What is an object?
- What are objects created from?
- How do objects store data and how can we access said data?
- What purpose do member variables serve?
- What purpose do member functions serve?
- How are member functions defined?
- What are accessors (getters) and mutators (setters)?
- How do we avoid stale data?
- What is a constructor?
- What is a UML diagram?
- What do the + and – symbols denote in a UML diagram?

Read/Watch/Review

- Read: Gaddis Chapter 7.2-7.6
- Read: Gaddis Chapter 7.8
- Read: UML Class Diagram Tutorial
- Read: UML Diagrams

Lab 7: Assignment

Description

- Create a UML Class Diagram for the Pharmacy and Patient Account classes

Overview: What to Submit

- Submit an image of your diagram with a short explanation of how the two classes interact with each other

Requirements

1. UML Diagram correctly identifies all functions and fields
2. UML Diagram correctly identifies Private vs Public methods and fields
3. UML Diagram correctly identifies Private vs Public functions
4. UML Diagram correctly identifies relationship between classes

Unit 8: Classes and Objects cont'd

Essential Questions

- What does it mean to overload a function?
- What does it mean to overload a constructor?
- What does it mean to pass an object as an argument to a function?
- What's the difference between passing an object by value vs passing it by reference?
- How do you return an object from a function?
- What is object composition and what purpose does it serve?
- How should code be organized into its respective files?
- How should you determine what classes to create for a project?
- How should you determine what member variables a class should have?
- How should you determine what member functions a class should have?

Read/Watch/Review

- Read: Gaddis 7.15
- Watch: Object Oriented Design: Object and Classes [10 min]
- Watch: Introduction to classes and objects for beginners [12 min]
- Watch: Introduction to Classes and Objects [8 min]
- Read: Gaddis 7.6
- Watch: Overloading Class Constructors [9 min]
- Read: Gaddis 7.9-7.11
- Watch: C++ Value and reference parameters [2 min]
- Watch: Header Files in C++ [12 min]

Lab 8: Assignment

Description

- Define and implement a class called UltimateBridgeSolver. **Use a header file** for your class declaration **and a separate .cpp file** for your implementation. Review 7.11 as needed.
- UltimateBridgeSolver should contain the following variables and functions, note what needs to be private vs public:

UltimateBridgeSolver
- smallLen : int - bigLen : int

+ UltimateBridgeSolver()
+ UltimateBridgeSolver(b : int)
+ getBigLen() : int

```
+ getSmallLen() : int
- canBuild(smalls : int, bigs : int, n : int) : boolean
- howManyBigs(smalls : int, bigs : int, n : int) : int
- howManySmalls(smalls : int, bigs : int, n : int) : int
+ solve(smalls : int, bigs : int, n : int) : void
```

- Below are descriptions for each function:
    1. **UltimateBridgeSolver()** is a no argument constructor that sets smallLen to 1 and bigLen to 5
    2. **UltimateBridgeSolver(b : int)** is a constructor that sets bigLen to b
    3. **getBigLen()** returns the value of bigLen
    4. **getSmallLen()** returns the value of smallLen
    5. **canBuild(smalls : int, bigs : int, n : int)** returns true if we can build a bridge of exactly of length n using at most smalls boards of length smallLen and at most bigs boards of bigLen. For example, if bigLen is 8, then canBuild(1, 3, 5) would return false
    6. **howManyBigs(smalls : int, bigs : int, n : int)** returns exactly how many boards of length bigLen we need to build a bridge of length n or -1 if it can't be built. Hint: Use a layered call to canBuild()
    7. **howManySmalls(smalls : int, bigs : int, n : int)** returns exactly how many boards of length smallLen we need to build a bridge of length n or -1 if it can't be built. Hint: Use layered calls to canBuild() and howManyBigs()
    8. **solve(smalls : int, bigs : int, n : int)** prints how many boards of each length are needed to build a bridge of length n or prints that the bridge cannot be made. Hint: Use layered calls to canBuild(), howManyBigs(), and howManySmalls()
- Create a **new file** to serve as your **main program file** and save it as **UBSTester.cpp** This file should only have one function, the main() function. In your main():
    1. Create an instance of your UltimateBridgeSolver using the no argument constructor
    2. Call getBigLen() and getSmallLen() as shown below to test that they are correct (in this example, the reference variable to my UltimateBridgeSolver object is called mySolver). If they print "false" (i.e., 0) then there is a bug in your code:
        i. cout << (mySolver.getBigLen() == 5);

        ii. cout << (mySolver.getSmallLen() == 1);

3. Perform the following calls testing what your functions return, as shown above and in the testing slides. I've given you partial answers, the rest you should figure out on your own:
    i. solve(3, 1, 8) //can build with 1 big and 3 smalls
    ii. solve(3, 1, 9) //can't build
    iii. solve(3, 5, 14) //can't build
    iv. solve(14, 5, 14) //can build with 14 smalls
    v. solve(4, 3, 7) //can build with 1 big and 2 smalls
4. Create an instance of your UltimateBridgeSolver using the overloaded constructor.
5. Create your own set of tests to ensure your functions work as expected for both instances of the class (default and overloaded constructor)

Overview: What to Submit

- UltimateBridgeSolver.h
- UltimateBridgeSolver.cpp
- UBSTester.cpp

Requirements

1. Code was divided into appropriate files, as specified, and was done so correctly
2. Member variables are correctly labeled public or private, per UML diagram
3. Member functions are correctly labeled public or private, per UML diagram
4. Member function signatures match UML diagram
5. Functions work correctly, as specified
6. Correct use of layered calls, as instructed
7. Tests performed correctly and as instructed for default constructor object
8. Tests performed correctly and as instructed for overloaded constructor object

Unit 9: Files

Essential Questions

Chapter 3

- What are the steps necessary to be able to Read: or write files using C++?
- Which file stream should be used if we are only reading the contents of a file?
- Which file stream should be used if we are only writing the contents of a file?
- Which file stream should be used if we are both reading and writing the contents of a file?
- How does one read data from a file using a file stream object?
- How does one write data to a file using a file stream object?

Chapter 4

- What's the purpose of checking if an error occurred when trying to access a file?
- How should one check if an error occurred?

Chapter 5

- How does one Read: data from a file when the file size is unknown?

Read/Watch/Review

- Read: Gaddis 3.12: Introduction to Files
- Read: Gaddis 4.14: Testing for File Open Errors
- Read: Gaddis 5.9: Using a Loop to Read: Data from a File

Lab 9: Assignment

Description

- Download these 4 files and either upload them to your editor. Write all of your code in a file called **main.cpp**.
- Write a function called `countCharacters()` that takes in **one string** parameter called `file`. Your function should **return** an integer representing the number of characters in the file.
- Write a **void** function called `mergeFiles()` that takes in **two string** parameters: `fileOne` and `fileTwo`. Your function should take the contents of `fileOne` and `fileTwo`, in that order, and write them to a file called "combined.txt".
- In your `main()` function, **and in the following order:**
    1. **Call and print** the result of `countCharacters()` on "groceries.txt"
    2. **Call and print** the result of `countCharacters()` on "names.txt"
    3. **Call and print** the result of `countCharacters()` on "Bradbury1.txt"

4. **Call** `mergeFiles()` on "Bradbury1.txt" and "Bradbury2.txt"
5. **Call and print** the result of `countCharacters()` on "combined.txt"
    - If both your functions work correctly, this should print 716

Overview: What to submit

- Submit **main.cpp** with your two functions and your five function calls

Requirements

1. Correct function header for countCharacters()
2. Correct function header for mergeFiles()
3. countCharacters() and mergeFiles() are called and printed as instructed
4. countCharacters() returns expected output
5. mergeFiles() has expected behavior

Unit 10: Pointers

Essential Questions

- What is a pointer and what purpose does it serve?
- How do pointer variables work?
- What is the relationship between arrays and pointers?
- What do arithmetic operations do to pointers?
- How should pointers be initialized?
- How should pointers be compared?
- What does it mean to pass a pointer as a function parameter?
- What does it mean for a function to return a pointer?
- What needs to be true in order for a function to return a pointer?
- How do pointers work with class objects?

Read/Watch/Review

- Watch: Introduction to Pointers in C++ [12 min]
- Read: Gaddis 10.1-10.7
- Watch: Introduction to C++ pointers Programming Tutorial [11 min]
- Watch: How to use pointers and arrays [13 min]
- Read: Gaddis 10.10-10.11
- Read: Pointers Tutorial

Lab 10: Assignment

Description

- Write a function, bookEnds(), that takes in an array of strings, **arr**, the array's size, **len**, and a pointer to an array of strings of size 2 called **ends**. You function should make **ends** contain the first and last elements of **arr**.
- In your main(), call your function twice, making sure to check for the expected output.
  - Note that this comparison should be of the value of the array. You may need to revisit Unit 6 to remember how to compare arrays.
- Write a function, switcharoo(), that takes in a pointer to an array of strings, **arr**, an index to replace, **idx**, and the string it should be replaced with, **str**. Your function should replace the value in **arr** at **idx** with **str**. You may assume **i** will be within bounds, that is, assume that **idx** is not bigger than the **arr**'s last index.
- In your main(), call your function twice, making sure to check for the expected output.
  - Note that this comparison should be of the value of the array. You may need to revisit Unit 6 to remember how to compare arrays.

Overview: What to submit

- Submit a single .cpp file containing both functions as well as your test calls (in main()) to them

Requirements

1. bookEnds() has correct function signature
2. bookEnds() returns expected output
3. bookEnds() is called in main() as instructed
4. switcharoo() has correct function signature
5. switcharoo() returns expected output
6. switcharoo() is called in main() as instructed

-

Final Project Requirements

- **Scope**: Scope of project must be significantly broader than that of an assigned lab.
- **Code**: Code must compile and execute without any errors (no crashes).
- **Modular**: Functions and objects must be used appropriately to abstract and divvy-up your code
- **Concepts**: Should demonstrate a clear understanding of basic concepts (i.e., conditionals, loops, functions) as well as more advanced ones (i.e., files, classes, pointers)
- **README**: Project should include a README.txt so users know how to use your program. README should answer:
    a. **Purpose overview**: What's the purpose of this project?
    b. **Instructions for use**: Describe how the user will interact with your program (is there anything the user shouldn't do? Tell us)
    c. **Design overview**: How did you choose to represent the aspects of your program?
    d. **Code overview**: An overview of how your program works, including how all of the pieces fit together
    e. **Potential bugs**: A description of any possible bugs or problems with your program
    f. **Easter Eggs**: A description of any extra features you chose to implement, if any
    g. **Group**: The names of all group members

End of Fall 2020 Semester Material

Removed Material

Functions Lab

Part 1 Description

- Write a value-returning function `calculatePerimeter()` that takes in three parameters: `shape` (a string), `firstSide` (an int), and `secondSide` (an int)
- The function should return the perimeter of `shape` as follows:
    1. If `shape` is "equilateral triangle" then the perimeter is firstSide * 3
    2. If `shape` is "isosceles triangle" then the perimeter is firstSide * 2 + secondSide
    3. If `shape` is "rectangle" then the perimeter is firstSide * 2 + secondSide * 2
    4. If `shape` is "square" then the perimeter is firstSide * 4
    5. If `shape` is none of the above, your program should return -1
- Call `calculatePerimeter()` in you program's `main()` function, passing in the arguments below and printing out the result using `cout`
    1. "equilateral triangle", 5, 5 //should return and print 15
    2. "isosceles triangle", 3, 5 //should return and print 11
    3. "rectangle", 12, 6 //should return and print 36
    4. "square", 3, 3 //should return and print 12
    5. "circle", 1000, 1000 //should return and print -1

Part 2 Description

- Write a void function `numberMatrix()` that takes in two int parameters: `numPerLine` and `lastNum`
- The function should print the numbers 1 through `lastNum` with `numPerLine` numbers per line (the last line may have fewer numbers)
- Call `numberMatrix()` in you program's `main()` function, passing in the arguments below
    1. 0, 12 //should print a blank line
    2. 1,1 //should print 1 on a single line
    3. 15, 25 //should print 1-15 on first line and 16-25 on second line
    4. 11, 12 //should print 1-11 on first line and 12 on second line
    5. 12, 11 //should print 1-11 on a single line
- Call `numberMatrix()` in you program's `main()` method, passing in user input as the arguments

Overview: What to submit

- Create a single project/file and have both parts in the same file
- Submit only Lab5.cpp

Requirements

- Part 1 function header correct
- Part 1 function body correct
- Part 1 function called in main() function as instructed
- Part 2 function header correct
- Part 2 function body correct
- Part 2 function called in main() function as instructed

Lab 8: Classes (due 4/10, 11:59pm)

- Continue working on Lab 8.

Mini-Lab 9: Classes and Objects (due 4/10, 11:59pm)

Description

- Create a UML diagram (or several UML diagrams) for a representation of library system
  1. Think about what a library does—how would you represent library branches, book collections, checking out books, returning books, public computers?
  2. How do you account for the fact that different library branches can share their books? (Hint: Think about how Ocean State Libraries allows you to check out books from anywhere in RI and how CCRI has different campuses that all share the same library system.)

Overview: What to submit

- A Word Document or PDF with:
  - One or more UML diagrams
  - A brief explanation (5-10 sentences) of major decisions you made when determining your representation of a library system

Requirements

- UML diagram(s) properly formatted (denotes private vs public, denotes variables vs functions)
- UML diagram(s) account(s) for library branches
- UML diagram(s) account(s) for book collections
- UML diagram(s) account(s) for checking out books
- UML diagram(s) account(s) for returning books
- UML diagram(s) account(s) for public computers
- UML diagram(s) account(s) for shared system of books
- Explanation makes sense for UML diagram(s) handed in

Grade Breakdown

- 100/100: All (8) requirements met
- 80/100: 6-7 requirements met
- 70/100: 5 requirements met
- 60/100: 1-4 requirement met

Final Project Proposal (due 3/6, 11:55pm)

Your final assessment for this course is a final project of you own choosing. You will have the option to work individually or in groups of up to 3 for your final project. Because this course does not meet in person often, you will have to figure out how to coordinate collaboration for the project. You can and should begin working on your projects as soon as you receive my feedback on your proposal.

What's final project worthy? Your final project should demonstrate what you've learned in this class. It's worth 400 points, which is the equivalent of **4-5 lab assignments** (i.e., all the labs we've done up to this point).

Description

- Write a project proposal that answers the following questions:
    1. What's the purpose of your project? (e.g., fun, educational, research, other?)
    2. What will the end product be? (e.g., a game, a data structure, a representation of something in the real world, other?)
    3. What concepts that we have covered (and will cover) will be used for this project?
    4. Who, if anyone, will you be working with, what will each person be responsible for, and how do you plan on working together?
    5. Create a timeline of milestones for your project (what do you want to have done by when. Keep in mind your project is due 4/24)

Overview: What to submit

- Answers to the all (7) questions above

Requirements

- Answers all (7) questions above
- Project scope is challenging and reasonable

Grade Breakdown

- 100/100: All (2) requirements met
- 70/100: 1 requirement met