CS A250
C++ Programming Language 2
Syllabus – Spring 2020

Instructor: Gabriela Ernsberger

E-mail: gernsberger@occ.cccd.edu

Office: MBCC 116
Office Hours:

- **Tuesday & Thursday:** 5:30 pm - 6:30 pm
- **Thursday (online):** 10:00 am - 12:00 pm
    - To chat with me during this time, log into **Google Hangouts**, using your gmail account, and send a message to: **mrs.e.occ@gmail.com**

A Note on Accessibility

If you have a disability that may impede your ability to successfully complete this course, you should contact the **Disabled Students Program and Services** (**DSPS**) at 432-5807 or 432-5604 TDD, **no later than** the first week of the course. Once you have completed the **DSPS application** process, **notify me** to make special arrangements. **Please note that without going through the DSPS application process no special arrangements can be made.**

Due to the nature of the material presented in this class, it is not possible to convert the graphs associated with the slides into plain text. Therefore, if you have a visual impairment, please inform me immediately. The **DSPS** center can provide most of the material in **Grade II Braille**, and I will provide the rest of the material, along with labs and exercises, in a format that is suitable for you.

Course Description

Second course in **ANSI/ISO Standard C++** programming language. Topics include sorting and searching, data structures, operator overloading, memory management, exception handling, name scope management, polymorphism, templates, STL containers, STL algorithm and iterators, and functional programming.

Prerequisites. Before enrolling in this class, you need to **complete CS A150** (C++ Programming Language 1) with a grade of C or better.

When taking this course, **you are EXPECTED to already know how to create C++ applications that include the following:**

- Control structures
- Functions that pass parameters by value and by reference
- Classes
- OOP and separate compilation
- Dynamic variables and arrays
- Pointers
- Recursion

Transfer Credit. CSU, UC.

Course Objectives. The objectives for this course are as follows:

1. Compare performance of various sort and search techniques.
2. Create class templates and function templates that show the extensibility of the language.
3. Create programs that use the container classes in the Standard Template Library (STL).
4. Apply memory management techniques to solve problems.
5. Differentiate between the assignment operator and the copy constructor.
6. Produce programs that use the concept of overloaded functions and operators with respect to classes.
7. Implement and manipulate singly- and doubly-linked lists.
8. Produce programs that show the understanding of polymorphism as it applies to C++.
9. Apply the concept of abstract classes to enhance their understanding of OOP concepts.
10. Produce programs that use the exception handling features of the language.
11. Produce programs that show an understanding and appreciation for iterators and algorithms in STL.
12. Produce programs that show an introductory understanding of functional programming.

***** LECTURE CONTENTS *****
Please note that the following topics
might **NOT** be presented necessarily in this order.

TOPICS	DETAILS
Separate compilation	1. Principles of Encapsulation 2. Header files 3. Implementation files 4. Reusable components 5. Using #ifndef
Linked data representations	1. Standard lists and iterators 2. Implement singly- and doubly-linked lists 3. Insert and remove nodes in singly- and doubly-linked lists 4. Stacks and queues 5. STL common containers and algorithms

Polymorphism	1. Polymorphic variables 2. Virtual and non-virtual overriding 3. Pure virtual member functions 4. Run-time typing information 5. Multiple inheritance
Name scope management	1. Name scope and lifetime 2. Forward reference 3. Protected scope 4. Friends 5. Nested classes 6. Name spaces
Operator overloading	1. Overloading simple arithmetic operators 2. Overloading comparison operators 3. Overloading input and output 4. Overloading increment and decrement operators 5. Overloading the assignment operator
Memory management	1. Different categories of memory 2. Constructors, destructors and copy constructors 3. Reference counting
Templates	1. Template functions 2. Template classes 3. Typedef 4. Non-type template arguments 5. Non-type template arguments
STL: Containers	1. Fundamental sequential containers – vector, list, deque 2. Adapted containers – stack, queue, priority queue 3. Ordered container – set 4. Associative container – map
STL: Iterators and Algorithms	1. Iterator as a general idea for pointers 2. Categories of iterators 3. Function objects, generators, predicates 4. Generic algorithm 5. Stream iterators
Algorithm efficiency	1. Big-O notation 2. Estimate the efficiency of algorithms 3. Compare the performance of algorithms 4. Compare sequential search and binary search

Functional programming	1. Installing interpreter 2. Prefix notation 3. Defining variables and procedures 4. Cond and if 5. Logical operators 6. Constructing procedures using Lambda

Student Learning Outcomes. Upon successful completion of this course, the student should be able to:

- Use C++ to solve problems that incorporate the use of class and function templates, linked lists, and overloaded operators.
- Use a functional programming language to solve problems by implementing lambda expressions.

Course Material

Textbook. There is no specific textbook for this course. **Slides**, **coding examples**, **assignments** and **practice exercises** make up the material you need to prepare for the course. If you would like to study from a textbook as well, then there are a few suggestions for you listed below (note that the book's edition does not really matter).

- **Absolute C++** (Savitch)
- **C++ How to Program** (Deitel)
- **Big C++** (Horstmann)

Keep in mind that some authors use different code syntax, which might not be the same style used in the examples I provide. You will be responsible for material given and presented in class, **NOT** for other material learnt from any of the suggested books.

Slides and Coding Examples. The **slides** provide details of the topics presented in class. Most topics will have **coding examples** in the form of **Visual Studio projects** and/or **.cpp/.h files**.

Software. We will be using **Microsoft Visual Studio 2019** for **PCs** to complete all programming exercises and exams. Do **NOT** download the version for Mac (see comment below). The software is installed on most of the computers in the Computing Center. If you would like to own the software, download a **free** version from the **Microsoft site**. Since the link keeps on changing, the best way to get there is to do a Google search, "**microsoft visual studio community c++ 2019**" and choose the community (free) version.

*** **MAC USERS:** If you have a Mac, **do NOT download** Visual Studio for Mac, because it is **different** from the one used for the exams and it does not have a good debugger (you will be tested on debugging). Install a **Virtual Machine** on your Mac and then download Visual Studio Community 2019 **for PCs**. Search on the Web for instructions on how to do this (you are a programmer and you must learn how to look for solutions on your own).

Grading

Grades will be available on **Canvas**. Always check your grades carefully and notify me immediately if there are any inconsistencies.

*** GRADING ***		
Assignments	5%	**A >= 90%** **B >= 80%** **C >= 70%** **D >= 50%**
Exam 1	20%	
Exam 2	20%	
Exam 3	35%	
Projects	20%	

Assignments. Assignments will be given weekly. You may work on assignments with other students--a **maximum of five (5) students in a group**, including yourself.

- **Late assignments:** If you submit your assignment past the due date, you will lose 5% for every hour, or portion of it.
- If you submit an empty file or the incorrect file, your work will not be graded. How do you know if you submitted the right file? Simply do the following:
    1. Once you submitted your file on Canvas, download it.
    2. Open it in a text editor (Notepad).
    3. Verify that it is the file you intended to submit.

Projects. There will be **two (2)** programming projects that you can complete by yourself or as a group with other students in the class—a **maximum of five (5) students in a group**, including yourself, is allowed. Projects will be assigned **after week 12**. You are **required to do most of the coding in class; therefore, attendance is mandatory** (see the homepage on Canvas for dates). **Points will be detracted** for each absence.

Exams. There will be a total of **three (3)** exams. The format of the exams will include multiple-choice and short answer questions, as well as programming using Visual Studio. All exams will be administered in class.

*** IMPORTANT ***
There is **NO make-up** for any classwork missed for being absent. This includes **all exams** and **in-class assignments**.

How to Study for this Course

The following are general guideline that can help you succeed in any course.

- **Complete the assigned reading** by the due date, to have a general idea about the topic. It will be easier to follow the lecture and understand details that you might have missed while reading.
- **During lecture,** pay attention and write notes. I will point out important details and items that are **not** on the slides, and might be included in the exams.
- **When possible (and allowed!) complete the assigned work with other students**. Instead of working on your own, team up with other students—you always learn more when collaborating with others.
- **Study all the material on the slides.** Keep in mind that the slides are a summary of the material presented on textbooks—**do NOT make a summary of the summary!**
- **Examine carefully the code examples.** It is important that you pay attention to the **syntax**, **readability** and **efficiency** of the code, because you will be tested on all of these.

HOW TO IMPROVE YOUR CODING SKILLS

- **Practice** your coding skills as much as you can. When you learn **new syntax**, you should try it on **Visual Studio**, even if it is not assigned as homework.
- Most of the **learning comes from making errors** and trying to correct them. If you write code and you make no errors, then you have learned **very little**. Change your code to crash the program or modify the output, and inspect the incorrect code to learn more!

Saving Your Work

As a programmer, you should already know how important it is to **back up your work** regularly. **You should always have at least two (2) copies of your work saved in up two (2) different places**. Accidentally deleting or losing your work is **NOT** an excuse for not turning in your assignments.

These are a few options on **how to save extra copies of your work**:

- Upload it to **Google Drive** via your student account.
- **Send an email to yourself** with your work as an attachment, **BUT** do **not** include the **.exe file**, because your email server will recognize it as an application and possibly discard the attachment.
- Bring a flash drive.

Programming Standards

Name Headers. You are required to write a name header on **ALL your coding files**. The name header is **placed at the beginning of each file** you are submitting. Failure to do so will cost you **1 point**.

The format required for the name header is shown below.

```
1   /*                      // (Make sure you use TABS to indent.)
2       Dijkstra, Edsger    // Last-name, First-name
3       Turing, Alan        // If you are working as a group, write all names.
4                           // (Leave one line.)
5       October 29, 2057    // Month MUST be written in letters.
6                           // (Leave one line.)
7       CS A250             // Course number
8       Lab 1               // Lab/Project/Exam number
9   */
10  // (Leave one line.)
11  // Your code here...
```

IMPORTANT: This is about **paying attention to detail**, which means that you are **expected** to pay attention to **spaces**, **commas** and **upper-** and **lower-case letters**.

Naming Standards. We will work as a **team**; therefore, we will all be using the same **standards** for **identifiers**.

STANDARDS FOR NAMING
Use **camelCase** for: • Variables (var, myVariable, aVariable) • Objects (obj, myObject, anObj) • Functions (func, myFunction, aFunction)
Use all **CAPITAL_LETTERS** separated by an **underscore** (_) for: • **GLOBAL** constants (CONSTANT, MY_CONSTANT) **NOTE:** This does **not** include **const parameters**.
Use **PascalCase** (this is also called **UpperCamelCase**) for: • Classes ➔ MyClass, BaseClass, DerivedClass… • File ➔ Main.cpp, MyClass.h, MyClass.cpp…

Academic Honesty

Although I encourage you to discuss your work with other students and look at their code as well, you are **NOT** allowed to copy code from someone else's implementation **AND/OR** to exchange files. At the end of the semester, **all files** from **all CS A250 sections** will be **tested** with **MOSS** (**Measure of Software Similarity**), which is a software used to detect plagiarism. If two or more students have the same files, **they will automatically get an F in the class and will be reported to the Dean of Students**.

RULES FOR ALL EXAMS

- **MUST** bring an **OCC ID**
- No cell phones (keep them in your backpack **OR** leave them with me)
- No flash drives (or any type of external memory)
- No notes, no books
- No calculators
- No smartwatches
- No hats
- No ear phones
- No talking to other students
- Seating will be assigned
- Bring a pen/pencil (I will provide paper)

You are not allowed to take breaks while you are taking the exam.
Leaving the classroom for any reason will end your exam.

Attendance

Attendance is mandatory. Class starts at the time shown on the schedule, but **lecture starts 10 minutes after**. This is to give you time to log in to a computer and be ready for lecture. Except for the first day, during those 10 minutes you are required to sign the **attendance sheet**, specifying at which computer station you are sitting (the computer number is on the upper right side of the computer screen).

ATTENDANCE RULES

- **Attendance** at **OCC** is **mandatory**.
- If **you miss any class meetings** during the **first two weeks** you **will** be **dropped without notice**.
- If you **miss more than three (3) class meetings**, you **might be dropped without notice**; do **NOT** assume, however, that you will be automatically dropped if you fail to come to class. It is **your responsibility** to withdraw from class and to check the deadlines on the OCC Web site.
- If you are more than 10 minutes late, **you will be considered absent**.
- If you **fall asleep** during class hours, you **will be asked to leave the room**.
- Note that **no student can be dropped AFTER** the deadline for the last day to drop with a W.

Communication

The best way to communicate with me is through **email** and **during office hours**. **I read and reply to emails early in the morning**; if you email me in the afternoon or evening, you will get a response the day after.

Email. You may send me emails from your personal account **as long as you write in the subject "CS A250"**. Replies will be sent to the sending address.

Announcements. I will email you any last-minute announcements through **Canvas**, so do **check your email daily**.

Other Course Information

Incomplete grade. I do **not** give incomplete grades to anyone. If you are not able to take the final exam, you will not pass the class.

Disruptive Behavior. I expect all of you to be polite, respectful, and helpful to your fellow students, and I expect you **to leave your sitting station neat and clean**. If, in my judgment, your behavior negatively impacts the rest of the class, you may be subject to disciplinary action.

Reservation of Rights. I reserve the right to change this syllabus without limitation and without prior notice. If I do modify any item or policy, I will notify you by sending you an email to your OCC email account.