**ECC CIS 121**

**Study Guide for FINAL EXAM**

The exam is closed book, closed notes; no discussion; calculators are not allowed.

You are encouraged to review all the assessment activities (e.g. quizzes, in-class exercises, homework).

A portion of the exam will be drawn from the programming assignments; you are encouraged to review the assignment description as well as your solution and/or the posted exemplars. You may be asked to ‘recreate’ portions of those assignments.

**Topics: review the assigned reading for specific details. However the major topics we have covered are:**

# Programming a Computer

* Create a project in Microsoft Visual Studio.
* Type, compile and run the program.
* Write a pseudocode description of an algorithm
* Identify the major components of a computer
* Identify the various phases in a software development project

# Basic Programming Concepts

* Declare variables and constants within a C++ program.
  + Variable naming rules
  + Data Types (int, double, char)
  + Constants
* Assignment statement
  + Arithmetic operators
    - +, -, \*, /, %
  + Integer vs. float division
  + Precedence & parentheses
* Read data fron the console input (cin)
  + Write data to the console output (cout)
  + Convert a series of math operations into a syntactically correct C++ math expression
  + Evaluate a C++ math expression to find its numerical value
  + Know the difference between integer and floating point division.
  + Place comments in your C++ program.
  + Read in a value from the console and place it in a variable
  + Display the contents of a variable to the console.

# Flow of Control – Selection Statement

* + The if/else statement
  + Boolean Expressions
  + Nested if/else
  + Be able to use the Boolean operators && and ||
  + Use a simple if statement in a C++ program
  + Use a simple if/else statement in a C++ program
  + Use nested if/else statements in a C++ program
  + Display a floating-point number to a specified precision.
  + Format outputted data using member functions:

setf() (ios::fixed, ios::scientific, ios::showpoint, ios::right, ios::left)

# Flow of Control II - Repetition Statements

* Use a switch statement in a C++ program
  + Use a while() statement in a C++ program
  + Use a do/while() statement in a C++ program
  + Write a program that sums/average all of the numbers
  + Debug code with a loop that is misbehaving
  + Design loops

# Procedure and Functions

* Call functions from the math library.
  + Know how to pass data to a function using parameters
* Know how to create a function prototype for a function
* Write a function that takes three integer parameters and returns the sum of the numbers
* Write a function that takes a single double precision floating point number and prints the number, its square, and its cube

# More About Functions

* + void functions
  + Recognize the scope of a variable
  + Create a global constant
  + Write a main() function that tests a function for proper operation.
  + Function overloading - functions with same name and different parameter lists
  + Understand how default parameters on functions affects function overloading
  + Random number generation: rand() (using library functions)

# Pass by Reference

* Reference variables
  + use & notation when declaring
  + an alias for another variable (i.e. nickname)
  + useful when variables are in different scopes (i.e. functions)
  + used in parameter passing and return values
* Pass by Value - local copies of parameters are made, and copies of returns are sent back
* Pass by Reference - Copies of parameters and returns are not made. Local parameters are references to the originals
* Use of const with reference parameters to prevent a function from changing the original (but avoid overhead of making a copy - faster execution)
* Write a function that takes two integers as parameters and returns both their sum and product
* Write a swap() function

# Streams and File I/O – Part I

* Create a stream:
  + ifstream ins;
  + ofstream outs;
* Connect a stream to a file on the computer disk: open()
  + Read a list of integers from a file on the computer disk and compute their sum
  + fail()
  + close()
  + Display a value in a field of a specified width.
  + Format outputted data using member functions: width()
  + Format outputted data using manipulators:

setw()

* Display a floating point number in fixed or scientific notation.

setiosflags() (ios::fixed, ios::scientific, ios::showpoint, ios::right, ios::left)

resetiosflags()

# Streams and File I/O- Part II

* Read the characters one at a time from one file and place them in another file
  + get(), put()
* Check the input end of file by using member functions eof()
* Write the program that takes input from data file and writes output to the screen and to the external file
* Write functions to convert between characters and numbers

# Arrays

* + Declare an array
  + Place data into an array
  + Reference data that is in an array
  + Pass data that is in an array into a function as a parameter
  + Pass an entire array into a function as a parameter
  + Pass an entire array into a function as a parameter
  + Search an array for a particular piece of data
  + Sort an array in a specific order
  + Multi-Dimensional Array
* Programming with arrays
* Write a function that will search and sort an array for a given value and return the index of that value
* Read data from a file and place it into a multidimensional arrays

### Array Properties

* indexed collection of data elements of same type
* consecutive storage locations
* default indexing is 0 through size-1 (where size is the number of elements in the array)

### Declaring Arrays

* format:   typeName variableName[size];
  + Example:  int list[10]
* The type can be any basic type or any user-defined type
* the size must be known by the compiler, so it must be a positive integer literal or constant.
* 2-dimensional arrays
  + Example:  double table[5][10]

### Initializing Arrays

* Can initialize arrays in the same line as declaration
* Format:    type name[size] = { list of elements };
  + Example:  int list[5] = {1, 3, 5, 9, 10};
* The list of elements goes in { } and is separated by commas
* may leave size box empty when **initializing** on the declaration line - compiler sets size.
* Can also initialize with for loops (good with regular patterns)

### Using Arrays

* valid indices are 0 through size-1.
* may use any of these index numbers to access a single array element:
* may use any positive integer r-value to index arrays (i.e. variables, expressions, etc)
* it is the programmer's job to check for out-of-bounds index!
* Copying Arrays
  + Assignment between array names does not copy one array to another
  + If you want to copy one array to another, do it element by element (easy with a loop)

### Arrays as function parameters

* Know how to pass an array into a function
* Usually a good idea to pass in a size as well
* Function always has access to the array contents -- only the address is sent in
  + There's no pass-by-value vs. pass-by-reference with arrays
* Use const on the array parameter when the function shouldn't change the array

### Array Usage and Algorithms

* Understand how to handle arrays that are declared to a certain size, but are not always "full" to their capacity
* Understand common array algorithms and patterns, including (but not limited to):
  + iterating through an array elements with a loop
  + Printing array contents
  + adding or counting array elements
  + finding largest/smallest element of an array
  + initializing array contents, with either formulas, user entry, or initializer list
  + Using parallel arrays
  + Swapping or moving around array elements

### Searching Arrays

* + **Linear Search**
    - The most obvious and intuitive search mechanism
    - Looks through array elements one by one until the item is located
    - takes longer than some specialized search techniques

### Sorting Arrays

* + **Bubble Sort**
    - Compares side-by-side elements. If out of order, swap them
    - Uses nested loops
    - Each run through inner loop "bubbles" one element to the end, its final position
    - Outer loop must run size-1 times
  + **Selection Sort**
    - Also uses nested loops
    - Run through inner loop "selects" the largest (or smallest) element of the remaining items (i.e. the ones that are not yet in position), and swaps it with the end element
    - Outer loop must run size-1 times

# C-Strings

* Create a character array for storing a string
  + - strings: null-terminated character arrays
    - can initialize on the declaration with a string literal
    - Example:  char name[7] = "Marvin";
  + size must leave room for null-character '\0'
  + Move a string from one array to another
  + Compare two strings
  + Read a string from the console
  + C-String to Numbers
    - atoi

## <cstring> library functions:

* strlen (string length)
* strcpy (string copy)
* strcmp (string compare)
* strcat (string concatenation)

### Using c-strings

* A c-string can be used like a normal array (of characters)
* cout and cin objects also work with c-strings (for output and input of words)
* >> operator for input stops at white space (space, tab, newline, etc.)
  + only good for one word at a time
* get and getline for reading strings from input
  + get, getline read up to specified delimiter -- can read entire sentences

# String Class

* String class and the member functions
* Read an entire line from the console into a string
  + Convert a string to a numeric value
  + Write functions to deal with string class

## string objects

* Built with the string class library
* variable length, flexible
* Supports more intuitive operator notations, like assignment, comparisons, + for concatenation, etc
* Understand the difference between c-strings and string objects
* Know the commonly used operators, as well as the usage of basic member functions discussed in class
* Understand the useage of the .size() , .length(), .clear(), .at()
* Understand the substr() function

### <cctype> library

* A useful C library of character handling functions
  + - toupper()
    - tolower(),
    - isspace()
    - isdigit()
    - isalpha()
* Understand the boolean functions whose names start with is, for determining if a character fits in a certain given category

# Week 15 Object Oriented Programing Part I

* Abstract Data Types
* Object-Oriented Programming
* Introduction to Classes
* Creating and Using Objects
* Defining Member Functions
* Declare a class and use it to create an object
* Implement a class’s behaviors as member function
* Constructors
  + Default constructor: Rectangle ( );
  + Constructors taking arguments: Rectangle (double wid, double len);
* Use a constructor to initialize an object’s data

Rectangle box;

Rectangle box (2.5, 5.2);

* Call an object’s member function to make them perform their tasks
  + box. getArea( )

# WHAT TO STUDY

* Your textbook. See the assigned reading in D2L.
* Programming assignments
* Homework assignments
* In-class exercise problems
* Quizzes
* Lecture handouts
* Your class notes

**Sample Questions:**

Also see the suggestions with the assigned readings.

**C++ basics**

1. Write the loop condition to continue a while loop as long as x is negative. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What is the value of x after the following statements?

int x;

x = 15 %4;

1. What is the value of x after the following statement?

float x;

x = 3.0 / 4.0 + 3 + 2 / 5

1. Given the following code fragment and the input value of 2.0, what output is generated?

float tax;

float total;

cout << "enter the cost of the item\n";

cin >> total;

if ( total >= 3.0)

{

tax = 0.10;

cout << total + (total \* tax) << endl;

}

else

{

cout << total << endl;

}

1. Given the following code fragment, and an input value of 5, what is the output?

int x;

if( x< 3)

{

cout << "small\n";

}

else

{

if( x < 4)

{

cout << "medium\n";

}

else

{

if( x < 6)

{

cout << "large\n";

}

else

{

cout << "giant\n";

}

}

}

1. What is the output of the following code fragment?

int x=0;

while( x < 5)

cout << x << endl;

x ++;

cout << x << endl;

1. Translate the following code fragment into an equivalent program segment using the **switch** statement.

if (x == 1) {

cout << "x is 1";

}

else if (x == 2) {

cout << "x is 2";

}

else {

cout << "value of x unknown";

}

1. What is the output of the given program fragment? Translate the following while statement into an equivalent program segment using the **for** statement.

int i=8;

while (i >10)

{

cout << i<< endl ;

i-- ;

}

1. Predict the output of the following program.

#include <iostream>

using namespace std;

int fun(int);

int main()

{

int k = 1;

cout << "The value returned is " << fun(++k) << endl;

return 0;

}

int fun(int x)

{

cout << "The value within the function is " << x << endl;

return (++x);

}

**Flow of control**

* + - 1. Which boolean operation is described by the following table?

|  |  |  |
| --- | --- | --- |
| 1. A | 1. B | 1. Operation |
| 1. True | 1. True | 1. True |
| 1. True | 1. False | 1. False |
| 1. False | 1. True | 1. False |
| 1. False | 1. False | 1. False |

* + - 1. Which of the following are equivalent to (!(x<15 && y>=3))?
  1. (x>15 && y<=3)
  2. (x>=15 && y < 3)
  3. (x>=15 || y < 3)
  4. (x>15 || y < 3)

1. What is the output of the following code fragment if x is 15?

if(x < 20)

if(x <10)

cout << "less than 10 ";

else

cout << "large\n";

1. What is the output of the following code fragment?

int i=5;

switch(i)

{

case 0: i=15;break;

case 1: i=25;break;

case 2: i=35;break;

case 3: i=40;

default: i=0;

}

cout << i <<endl;

1. What is wrong with the following switch statement?

int ans;

cout <<"Type y for yes on n for no\n";

cin >> ans;

switch (ans)

{

case 'y':

case 'Y': cout << "You said yes\n"; break;

case 'n':

case 'N': cout << "You said no\n"; break;

default: cout <<"invalid answer\n";

}

1. What is the output of the following code fragment?

int x=0;

{

int x=13;

cout << x <<",";

}

cout << x << endl;

1. What is the value of x after the following code executes?

int x=10;

if( ++x >10)

{

x =13;

}

1. How many times is "Hi" printed to the screen

for(int i=0;i<14;i++);

cout <<"Hi\n";

1. Given the following code, what is the final value of i?

int i,j;

for(i=0;i<4;i++)

{

for(j=0;j<3;j++)

{

if(i==2)

break;

}

}

1. What is wrong with the following for loop?

for( int i=0 ; i<10 ; i--)

{

cout << "Hello\n";

}

**Functions**

1. What is the correct way to call the following function? Assume that you have two variables named intArgument (int) and floatArgument(float).

void doThings(float x, int y);

1. If you need a function named getData to get both the number of items and the cost per item from a user, write a good function declaration to use.
2. What is the output of the following function and function call?

//function prototype

void calculateCost(int count, float& subTotal, float& taxCost);

//code fragment

float tax = 0.0,

subTotal = 0.0;

calculateCost(15, subTotal,tax);

cout << "The cost for 15 items is " << subtotal

<< ", and the tax for " << subTotal << " is " << tax << endl;

//end of fragment

void calculateCost(int count, float& subTotal, float& taxCost)

{

if ( count < 10)

{

subTotal = count \* 0.50;

}

else

{

subTotal = count \* 0.20;

}

taxCost = 0.1 \* subTotal;

}

1. Given the following function definitions and program fragments, what is the output?

void f1(int& z, int &q)

{

int temp;

temp=q;

q=z;

z=temp;

}

void f2( int& a, int& b)

{

if( a<b)

f1(a,b);

else

a=b;

}

in main():

int x=3, y=4;

f2(y,x);

cout << x <<" " << y << endl;

1. What is the output of the following code fragment?

double size, volume=16.0;

size = sqrt(sqrt(volume)) / 3;

cout.setf(ios::fixed)

cout.setf(ios::showpoint);

cout.precision(2);

cout << size;

1. What is the output of the following function call? How to change the code to output the correct value?

//function body

int factorial(int n)

{

int product=0;

while(n > 0)

{

product = product \* n;

n—;

}

return product;

}

//function call

cout << factorial(4);

1. What is the output of the following function call? How to change the code to output the correct value?

//function body

int factorial(int n)

{

int product=0;

while(n > 0)

{

product = product \* n;

n—;

}

return product;

}

//function call

cout << factorial(4);

**I/O Stream**

1. Write the codes to correctly assign all the values in array1 to the array2? (Assume both arrays are of the same type and have SIZE elements)

1. To open a file with a user supplied name, you would need to store the name in a variable. If the file name was to have no more than 20 characters in it, how to declare the file name variable?
   1. char filename;
   2. char filename(20);
   3. char filename[20];
   4. char filename[21];
2. What does the command outFile.precision(2) do?

1. We have a file that has a name in it, but the name is written one character per line. We need to write this name to the screen. What is wrong with the following code?

ifstream fileIn;

fileIn.open("file.txt");

char ch;

fileIn.get(ch)

while(!fileIn.eof())

{

cout.put(ch);

fileIn.get(ch);

}

1. Which function returns true if the character argument is a letter?
   1. isdigit
   2. islower
   3. isalpha
   4. isspace

**Arrays**

1. Write the declaration for a function named funct1 that expects an array of floats, the number of elements in the array and does not return any value.

1. What is wrong with the following code fragment?

const int SIZE =5;

float scores[SIZE];

for(int i=0; i<=SIZE;i++)

{

cout << "Enter a score\n";

cin >> scores[i];

}

1. Declare an array named myArray that can hold up to 3 rows of 5 columns of doubles?

1. Write a statement which will read values from the keyboard into the array myArray? (Assume the size of the array is SIZE).

**Strings**

1. How do you call the function to read a whole line of input(up to 80 characters) from the keyboard into a c-string named str?

1. How do you concatenate two string values (str1, str2)?

1. Declare a c-string and initialize it to the value of "phonebook",

1. If the name of a file to open is in the string variable name fileName, which of the following will correctly open the file for output?
   1. out\_file.open(fileName);
   2. out\_file.open("fileName");
   3. fileName.open(outfile);
   4. out\_file.open(fileName.c\_str());

**Objected Oriented Programming**

* + - 1. Explain what public: and private: do in a class definition. In particular, explain why we do not just make everything public:?
      2. What is a constructor? What is a default constructor?
      3. Given the following class and object declaration, how would you print out the age and cost of a bottle of wine?

class Wine

{

public:

Wine();

int getAge();

float getCost();

private:

int age;

float cost;

}

Wine bottle;

* 1. cout << bottle;
  2. cout << Wine.age, Wine.cost;
  3. cout << bottle.getAge() << bottle.getCost();
  4. cout << bottle.getAge << bottle.getCost;
  5. cout << bottle.age << bottle.cost;

1. In a class, all members are \_\_\_\_\_\_\_\_\_\_\_\_ by default
   1. public
   2. private
   3. global
   4. all of the above

1. Data members or member functions of a class that are declared to be private may
   1. only be accessed by the main program
   2. only be accessed by members of the class
   3. not be accessed by the class
   4. are considered to be global variables
2. Given the class definition, how could you use the constructor to assign values to an object of this class?
3. Which of the following is the correct function definition header for the getAge function which is a member of the Person class?
   1. int getAge();
   2. int getAge()
   3. int Person:getAge()
   4. int Person::getAge()

1. Declare a class called BankAccount to maintain your financial records.
   1. What information should be public, what should be private?
   2. What data should be stored inside the object representing your bank account?
   3. How would you declare an object so that the object automatically called the default constructor?

**Programming Questions:**

Please review the exercise in labs and assignments.

**Sample questions：**

1. Write a function that takes in two integer parameters and returns true if the first parameter is a multiple of the second, and false otherwise. (Hint: The % operator can help you determine factors and multiples)
2. Write a program that has the user enter the radius of a circle, then compute and print the diameter, the circumference, and the area. Use the value 3.14159 for pi.
3. Write a function that takes in two integer parameters, x and y, and computes and returns x raised to the y power. Do this without using any <cmath> library functions
4. Write a function that takes in a positive integer parameter N, and then returns the sum of all the positive numbers that are less than or equal to N. (i.e. add up all numbers from 1 through N -- you'll need a loop).
5. Write a program that asks the user for a file name. Assume the file contains a series of numbers, each written on a separate line. The program should read the contents of the file into an array and then display the following data:
   1. The lowest number in the array
   2. The highest number in the array
   3. The total of the numbers in the array
   4. The average of the numbers in the array