Review for Final – CS 102

Vocabulary

Know vocabulary from each chapter

- Take a look at the previous chapter study guides

Also, know this vocabulary from Chapter 8

Binary file, text file, bit, byte, file, primary storage, secondary storage, volatile storage, reading a file, writing to a file

General Ideas

Know things to do to increase readability

- Commenting, Indenting, separating chunks of repeated code into functions

Be able to use the C++ data types

- string, char, int, (long), double, Boolean (bool), float

Be able to use and understand the math operators, including ++, --, and the shortcut operators

- ++ increments by 1
- -- decrements by 1
- Others include things like +=, -=, /=, *=, %=

Know basic string ideas (Declaring, initializing, concatenation, the length function)

- Declare a string using variable name, and then the letters to go inside a string: string name = "Kayley";
- Initialize a string the same way, could be something like string name; if you do not have anything in it yet
- Concatenate using + in between the different strings, like name+" "+lastname
- Use the `length()` function to tell you how long a string is
- Use the `substr()` function to give you a part of a string; the first input being which character you want to start at and the second input being the length you want

Why is duplicating code with copy-and-paste bad? What are some ways to get around that?

- You make the code a lot longer than it actually needs to be
- You can put the repeat code in functions to not only organize your code but increase readability as well

What is a constant variable? How do you declare a constant variable?

- A constant variable is a variable that will stay the same throughout a program.
- You declare a constant variable by using `const` in front of your data type declaration: `const double PI = 3.14159;`

  What are the naming conventions for a constant variable?
  - You use capital letters
  - One benefit of constants are that you can save a little bit of runtime

Be able to code a switch statement

- It is the usual way to test if a variable has one value chosen from a list
- ```cpp
  switch(employee_code)

```
{
 case 3:
 {
  cout << "Programmer";
  break;
 }
 case 2:
 {
  cout << "Department Manager";
  break;
 }
 case 1:
 {
  cout << "CEO";
  break;
 }
 else
 {
  cout << "Invalid code";
 }
}
```

The if statement

Know how to compare floating point numbers

- You cannot compare if the float variables are equal outright because of round off errors, they could be really close (basically they are equal)
- The test to see if they are the same is : `if (fabs (double1 – double2) < epsilon))`

The last choice in a multi-way if should almost always be an else, not an else if

Know what nested ifs are

- Not hard to understand, just if, if else, else statements inside a bigger if, if else, or else statement

Know what short-circuit if logic is

- Boiled down, short-circuit `if` logic means that in some cases you do not need to test all the conditions; this is highly case-by-case dependent

Know how to check if a data value is in a range

(For example, is x between 10 and 20?)

- Inside the parenthesis you would write a statement like `(x > 10 && x < 20)` then do something with it

Be able to code an if using &&, ||, !

- && means and
- || means or
- ! means not, can be used in conjunction as !=, means does not equal to

Loops

The for loop is primarily for counting. It can count up or down, and by any number

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

The other loops are while loops and do…while loops

Know which loops are pre-test and which are post-test and what these words mean

- Pre-test means that you check the condition before entering the loop; examples include a for loop and a while loop
- Post-test means that you check the condition after entering the loop; one example is the do (while) loop

Know what the word iteration means

- Iteration is a fancy way of saying one time through the loop

Typical applications of loops. Know the code for each of these.

Input validation

- You can check using a loop (do…while, while)
- Check using an if statement and the `fail()` function: `if (cin.fail ()){}`

Calculating a sum and average, especially for the elements in an array

- For sum, the basic logic is to do a for loop to iterate through an array, and then have it add on to the total variable declared earlier
- For average, you use the same logic as sum, except you need to declare another variable to check for how many times you iterate through the loop, and after exiting the loop, divide the total by the number of iterations
- https://github.com/kayleyseow/cpp-problems/blob/master/04.%20Loops/Chapter4Notes2.md#applications-of-loops-1-calculating-a-sum-and-average

Finding the maximum and minimum

- Gist of it is that you set a variable equal to the first number in an array, and then you have a loop starting from the second element of an array to test if it is larger or smaller
- Same logic applies to non-array, you just need to test which one is bigger or smaller and test it
- https://github.com/kayleyseow/cpp-problems/blob/master/04.%20Loops/Chapter4Notes2.md#applications-of-loops-4-finding-the-maximum-and-minimum

Counting spaces in a string

- Use a for loop to iterate through the length of a string, when you compare it and the char at the string is equal to a space, you can increment the counter variable declared earlier
- https://github.com/kayleyseow/cpp-problems/blob/master/04.%20Loops/Chapter4Notes2.md#applications-of-loops-2-counting-spaces-in-a-string

Finding the first match

- Similar to counting spaces, whereas this time you are finding the match to something else and you use a while loop to iterate
- https://github.com/kayleyseow/cpp-problems/blob/master/04.%20Loops/Chapter4Notes2.md#applications-of-loops-3-finding-the-first-match

Function ideas

Know the benefits of using functions

- Simplify the coding process
- You do not need to rewrite code
- Testing is easier
- Easier for a team to design a large project

Functions should always be short: A function should do only one task

Code after a return statement is not executed; it is ignored. It is called unreachable code.

A function of type void should not have a return

A function of any other type must have a return that matches the function type

What is the difference between pass-by-value and pass-by-reference?

- "The terms "pass by value" and "pass by reference" are used to describe how variables are passed on. To make it short: pass by value means the actual value is passed on. Pass by reference means a number (called an address) is passed on which defines where the value is stored." https://blog.penjee.com/passing-by-value-vs-by-reference-java-graphical/#:~:text=The%20terms%20%E2%80%9Cpass%20by%20value,where%20the%20value%20is%20stored.

The default calling scheme in C++ is call by value; the exception to this is arrays

To change this, you need to put an ampersand before a variable's name

- For example, you would do `void change(int& i)`; basically you just put an & before the parameter

Arrays

An array has a definite size that cannot be changed while a program is running

Know how to declare an array given some information about it

- Declare an empty array with size: string words [3];
- Declare an array with values: string words[] = {"Kayley", "Hoffman", "C++ is fun"};
- Declare using a mix of both: string words [3] = {"Kayley", "Hoffman", "C++ is fun"};

Know what the binary search is and be able to explain it with pseudocode

- TL;DR – you keep on halving the things which you are looking for until you eventually find the number
- Need to keep track of which side the number is on
- Use a while loop

How do you pass an array to a function?

- You just pass it in like usual, into the parameter, but remember to add in brackets to indicate that it is an array
- Add in the size of the array as a parameter as well

When an array is passed to a function, it is passed by reference

Know how to use an array. Some things to know:

Be able to print and/or calculate with an element of an array

- Use a number at array[x], for example, and treat it like an integer
- Can just calculate using that

Be able to initialize an array using a for loop
- Iterate the for loop for as many elements as there are in the array
- Add in whatever you want into each element
- Remember to start at 0

Be able to calculate the sum, average of the elements in an array
Be able to find the maximum, minimum element in an array

Be able to copy one array to another array
- You have to do this element by element
- Make another array of the same size
- Use a for loop to copy things over into the other array

Vectors

You can add an element to the end of a vector using push_back
You can delete an element from the end of a vector using pop_back
Know how to use the size member function
- It is basically just `size()`

You can copy one vector to another like you copy two ints
- You have to do this element by element
- Make another vector of the same size
- Use a for loop to copy things over into the other vector

Declare a vector by using `vector <datatype> variablename;`
Call `vector`s much like you would call an array, like `variablename[element];`

Sorting

Sorting data takes a long time.
Be able to explain the selection sort using pseudocode
- Find the smallest element in the array
- Swap the elements
- Minimum value put to the front, and then from the next position on, swapped, etc.

Know how to swap two data items
- Make a third variable as a temporary one
- Copy the first variable into the third variable
- Copy the second variable into the first variable
- Copy the original first/third variable into the second variable
- Remember to use `&` (reference) while doing a function

Pointer Ideas

Know how to declare a pointer variable
- int* int_ptr;

Know how to assign a pointer variable to point to a variable using &
- int* int_ptr = #
- If it is already declared, int_ptr = #

Be able to use an assigned pointer value
- Remember, a `*` in front of a pointer name means that it is dereferencing and directly changing the value of the data at the reference value that the pointer is pointing to

Pointers not in use should be set to NULL
Know specifically what the new and delete commands do

- Both are used for dynamic memory allocation
- `new` allocated new memory in the heap and returns a pointer
- `delete` recycles the memory in the heap, and you can not use the location after it has been deleted

Pointer Arithmetic and the Array/Pointer Duality Law

Be able to use the array/pointer duality law to access locations in an array

- `a[i] = *(a+i)`

The Operating System (OS)

The operating system is software that controls the computer's resources
Be able to name some resources and talk about how the OS controls them

- Keyboard, the monitor, memory, the storage, loads programs into memory

Linked lists

Know what a linked list is

- Linked using pointers
- A pointer to the start of the list, then the node will point to the next node

Know how to add and delete from the beginning of a linked list
Be able to write code that traverses a linked list

Text files

Know how to declare and open a text file for input or output

- Input:
  - ifstream in_file;
  - in_file.open(filename.c_str());
  - in_file >> name >> value;
- Output:
  - ofstream out_file;
  - out_file.open("c:\\output.txt");
  - out_file << name << " " << value << endl;

Be able to read from (using >> or getline()) or write to the file

- Read from:
  - Using >>
    - string a_line;
    - data_file >> a_line;
    - Read several variables using data_file >> line_1 >> line_2;
  - Using getline()
    - string input_line;
      getline(data_file, input_line);
    - getline (cin, string_variable);
  - Remember to check for failure
- Write to:

- ofstream out_file;
- out_file.open ("\\CS102\\programs\\datafile");
- out_file << data_to_write << endl;
- out_file.close();

Random Access Files

A random access file is one type of binary (non-text) file

Know how to declare and open a random access input stream

- fstream random_file;
  random_file.open (filename.c_str(), ios::in | ios::out | ios:: binary);

Know that usually, the data is read into/written from a struct

Know that you can position the file pointer using seekg(). Also, know how to do this

- If you want to read record n, first use:
  data_file.seekg ((n-1)*sizeof(a_record));

The command prompt

To access arguments from the command prompt, main() should be

```
int main (int argc, char* argv [])
```

- You need two parameter variables, the first one being an integer and the second one being an array of string literals of type char*
- Apparently used to specify options and file names
  - Argc tells how many arguments were used
  - Argv holds the arguments

Know how to redirect input and output (< for input, > for output, and | to connect programs)

- Note that cin and cout work with the I/O redirection
- < is for redirecting input; for example, you can type in more < dirlist to display the dirlist file one screenful at a time
- > is for redirecting output; for example, you can type in dir > dirlist and use the dirlist like any other file
- | lets you use the output of one program as the input to another; for example, you can use `dir|more` when you do not want to save a file
  - Something basically like a < firstfile > secondfile except maybe like a firstile|secondfile?

True or False?

Clients generally care about what a function does but not how the function does it.
True

Overloaded functions are selected by number, types, and order of types of parameters.
True

Variables declared const cannot be modified after they are initialized.
True

A whole array cannot be passed to a function—each element must be sent to the function separately.
False

An array is a dynamically resizable data structure.

False
A vector is a dynamically resizable data structure.
True
A sequential access file allows records to be accessed in any order.
False
A sequential access file allows records to have variables sizes.
True
A random access file allows records to be accessed in any order.
True
A random access file allows records to have variable sizes.
False
Records in a sequential file are usually updated in place.
False
A for loop is a post-test loop.
False

The final exam will have some multiple-choice questions, some true/false questions, and some short answer essay questions. Like the other tests, some questions will involve understanding code and some will require writing code.