CS 003A
Exam 1 Review

General Knowledge of

Functions:

- T/F If a function contains a statement that changes a value parameter, only the copy of the argument is changed, not the original.
  T
- How do you declare a function?
  return_type funtion_name (parameter types);
- Where does the prototype go?
  At the top of the file.
- What is the scope of a local identifier?
  Between the curly braces. {}
- T/F The scope of a value parameter is identical to the scope of a local variable declared in the outermost block of the function body.
  T
- T/F In C++, corresponding arguments from a calling function and parameters from a called function must have the same name.
  F
- When would you use a void function?
  When the function does not return data, ex. just displaying output to console
- How do you return multiple values from a function?
  You either use a heap allocated array, use a vector, or my favorite, return a struct
- T/F When passing by value data flow is one-way – into the function
  T
- What type of arguments can be sent from the calling function when passing by reference?
  If not a const reference, only variables, aka something with both a value and address (lvalue)
- What type of arguments can be sent from the calling function when passing by value?
  Any, both variables and just simple literal values.
- T/F All functions should have a return statement.
  F
- If a module is supposed to compute and return the average of five numbers, which is more appropriate a value returning function or a void function?
  Value returning function
- T/F Functions must have a return data type
  T
- What should the return data type for main be?
  int
- What is the advantage of using functions?
  you break up compute into sections. It is easier to reuse code, read the code, and modify the code
- Declare a function that is passed two integers representing a sum and a count and returns an average.

- Declare a function that returns the sum & avg of an integer array.

```
struct output
{
        int sum;
        double avg;
};
output SumAndAverage(int array[], int size);
```

- T/F You can only have 1 return statement in a function
  F
- What is an advantage of passing by value?
  The function can modify the passed in argument without outside changes; the passed in argument is assured to not change. You don't need to use a variable as argument
- What is a disadvantage of passing by value?
  Can not change the actual passed in variable. If passingbig structs/classes, a lot copying will occur: this is slow.
- When we pass by reference what is passed to the calling function.
  A reference to the variable, which is in essence an abstraction of the address of the variable that the language takes care of for us.
- What is a disadvantage of passing my reference?
  If you change the passed in argument inside the calledfunction, the changes will also appear outside.
- How do you pass by reference?

You add an & to the parameter type when defining and declaring the function. You pass arguments normally.

Header Files

- T/F You can put executable code in a header file.
  F (unless you count using a constructor for a an object defined in the header file as execution)
- T/F It doesn't matter what order the directives and declarations appear in a header file.
  F
- What order should the directives appear and why?
  The directives should appear at the top of the fileas to be able to encapsulate the rest of the header file. Remember, #include more or less just pastest in text inside of whatever of the file that is being told to include such as <iostream>
- What are these lines in a header file for?
  #ifndef MYHEADER_H_
  #define MYHEADER_H_
  #endif
  The are to ensure that the header file myheader.h is not declared in more than once.
- Does the extension for your header file matter?

No, not really. it just allows us humans to differentiate the type of files quickly. At the end of the day it's just bytes.

- How do you include a user-defined header file?
  #include "user_headerfile.h"

Arrays

- What is an array?
  A collection of the same types, all laid right nextto each other.
- Is this a valid statement: firstArray = secondArray;
  No.
- Is this a valid statement: if(firstArray == secondArray)
  Yes. but it is just comparing the address of the firstelements in each array
- T/F an array is a composite data type.
  F, they are all the same data type
- Why should we use constants for array size?
  Because arrays don't change in size
- How do we initialize an array?
  type array_name[number_elements] = {values,...,last_value};
  also
  array_name[array_index] = value;
- T/F An individual component of an array cannot be passed as an argument to a function. The entire array must be sent.
  F
- How many different data types can you have in one array?
  One
- T/F Given the declaration

  int someAr[20];
  int someAr2[20];
  cout << someAr[3]; outputs the 3rd element in the array. **F (the 4th element: starts counting at 0)**
  cin >> someAr[20]; will produce a compiler error **T**
  someAr = someAr2; will transfer all values from someAr2 to someAr **F**
- What is the advantage of passing by reference?
  The amount of data that is passed is reduce, but youare still able to access all the data by the passed in variable, and you can also modify the passed in variable
- Which of the following is true about an array?
  a) Arrays are always passed by reference. **T**
  b) The name of an array is the address in memory of the first element. **T**
  c) Array subscripts always begin at 0. **T**
- How would you declare an array of 20 c-strings that can hold up to 11 characters?
  char array[20][12]; //12 to hold null terminating character
- How would you compare an array of int.

with a loop and if statements

- What will this statement do: int item[5]={2,12,1};
  initialize the first three elements in the item array.
- What will this statement do: int item[5]={0};
  initialize the whole item array to all 0s
- What will this statement do: int item[5]={2,12,1,2,9,5};
  initialize the 5 elements in the item array, and then change the value of of what the address at item+5 has; aka *(item+5) = 5. You shouldn't change it, but that's what it does. NOTE, usually the compiler will give you a warning/error.
- T/F The compiler will give you an out of bounds error when using arrays if your index is too big or too small.
  F, they will usually just let you compile
- What is stored in the array variable? (e.g. myArray)
  The address of the first element in the array
- What is the base address?
  The address of the first element in the array thatevery other element in the array is relative to
- Know how to use loops with arrays
  I do!
- T/F Arrays can be returned as a return value in a function.
  T
- T/F Arrays must be passed by reference using the &.
  F
- T/F When you pass an array you don't have to include the size in the parameter list for the first dimension.
  T
- How should you pass an array when you don't intend to modify it in the function being called.
  With the const qualifier in the function
- T/F C-Strings are special arrays.
  T, they are arrays of characters
- T/F char name[16]= "Pete"; ⇔ char name[]= "Pete";
  F
- How would you declare parallel arrays that could contain a movie title, genre, and running time (in minutes)? Assume a const AR_SIZE
  std::string movie_title[AR_SIZE];
  std::string genre[AR_SIZE];
  int running_time[AR_SIZE];

- How would your read these values in from a file (assume a list with \n after each entry) – use the fstream variable inFile?

```
for (int i = 0; i < AR_SIZE; i++)
{
        inFile >> movie_title[i];
        inFile >> genre[i];
```

```
        inFile >> running_time[i];
}
```

- How would you declare a multi-dimensional array that would hold 10 scores for 5 people?
  int scores[5][10];
- How would you read these values into the array from a file (assume the inFile variable is already assigned – assume 10 scores per row)?

```
for (int i = 0; i < 5; i++)
{
        for (int j = 0; j < 10; j++)
        {
                inFile >> scores[i][j];
        }
}
```

- how to output multi-dimensional arrays and how to initialize them.

  if n dimensions, use n for loops, with n variables, to index each index.

  array_type
  array_name[array_size_dimension_1][array_size_dimension_2]...[array_size_dimension_n]

  =

  {{...{ /*n curly braces*/ values}...}};

Structs

- What is the advantage of using structs?
  Increased readability, and just less code to write, and everything is tied in a neat data package.
- What is a member?
  A variable associated with the struct
- Are aggregate operations allowed on structs?
  Yes, they are more or less like classes, just that by default their members are public.
- Can you pass structs by value or reference?
  Either or
- Can structs be a return type?
  Yes.
- Define a struct called DvdRec, that contains the title, genre, and running time.

```
struct DvdRec
{
        std::string title;
        std::string genre;
        int running_time;
};
```

- Declare an array 100 elements of that struct called movies.
  DvdRec movies[100];
- How would you output the title of the 10th element in your array?
  std::cout << movies[9].title << "\n";
- Be able to write a function that can read into an array of structs or output an array of structs.

  Yes? I am able.

  void funcName(struct_type[]);

  struct_type* funcName(void);

Pointers

- What is a pointer?
  It is an address value in memory.
- Given the following contents in memory. What would these statements output?
  cout << intPtr; **//The address of whatever integer intPtr is pointing to**
  cout << *intPtr; **//The value of whatever integer intPtr is pointing to**
  cout << &intPtr; **//The address of the intPtr variable.**
- Know how to access members of a struct using pointers.

  structPointer->member;

Classes

- What is the difference between Unstructured and Structured programming?
  Unstructured programs are not split up into sub tasks; it is the opposite for structured programing.
- What is the difference between Structured and OOP?
  Types are also part of execution, each object can do work. We further organize our code into conceptual class, and the objects derived from them.
- OOP – focus is on data and methods are used to access data
  T
- What is information hiding?
  Not showing all the attributes of an object. Keepsome of it hidden from the public interface.
- What is the difference between a mutator and an accessor?
  Mutators change the state of the object. Accessorsget and return the state of the object without modifying the object.
- What is a constructor?
  It is a special type of pseudo method that usuallyis made to initialize an object to be ready to complete whatever task it was set out to do.
- What is a destructor?

It is a special type of pseudo method that is usually used to free up resources associated with a specific object.

Command Line

- How to compile programs on the command line.
  compiler_name cpp_program (any arguments such as optimization, std, warnings, output name)
- How to compile without calling the linker.
  Using arguments, for gcc I believe you can use -E to just get the preprocessed file and -S to get the assembly code
- How to compile on the command line and set the executable file to something other than the default executable file (a.out).
  with gcc, use -o file_name
- You should know some simple bash commands, such as ls and cd.
  I do.

1. Write the definition of a void function that takes as input a decimal number and as output 3 times the value of the decimal number. Format your output to two decimal places.

```
void func(double num)
{
        std::cout << std::fixed;
        std::cout << std::setprecision(2);
        std::cout << num*3.0 << "\n";
}
```

2. Write the definition of a `void` function that takes as input two parameters of type `int`, say `sum` and `testScore`. The function updates the value of `sum` by adding the value of `testScore`. The new value of `sum` is reflected in the calling environment.

```
void func(int& sum, int testScore)
{
        sum += testScore;
}
```

13. What is the output of the following program?

```
#include<iostream>

using namespace std;

void find(int a, int& b, int& c);

int main()
{
   int one, two, three;
   one = 5;
   two = 10;
   three = 15;
   find(one, two, three);
```

```
   cout << one << ", " << two << ", " << three << endl;
   find(two, one, three);
   cout << one << ", " << two << ", " << three << endl;
   find(three, two, one);
   cout << one << ", " << two << ", " << three << endl;
   find(two, three, one);
   cout << one << ", " << two << ", " << three << endl;
   return 0;
}

void find(int a, int& b, int& c)
{
   int temp;
   c = a + b;
   temp = a;
   a = b;
   b = 2 * temp;
}
```

5, 10, 15
20, 10, 15
25, 30, 15
45, 30, 60

14. Identify error(s), if any, in the following array declarations.
    a. `intlist[10];` //**No type on the left**
    b. `const int size = 100;`
    c. `double list[SIZE];` //**SIZE is not defined**
    d. `int numList[0..9];` //**Incorrect syntax 0..9**
    e. `string names[20];`
    f. `scores[50] double;` //**type declaration should be on left, not right**
15. What is the output of the following code?

```
int list[] ={6, 8, 2, 14, 13};
for(int i = 0; i < 4; i++)
   list[i]= list[i]- list[i + 1];
for(int i = 0; i < 5; i++)
   cout << i << " " << list[i]<< endl;
```

0 -2
1 6
2 -12
3 1
4 13

16. Suppose that you have the following function definition.

```
void sum(int x, int y, int& z)
```

```
{
   z = x + y;
}
```

Consider the following declarations:

```
int list1[10], list2[10], list3[10];
int a, b, c;
```

Which of the following function calls is valid?

```
a. sum(a, b, c);     //Valid
b. sum(list1[0], list2[0], a); //Valid
c. sum(list1, list2, c);
d. for(inti = 1; i <= 10; i++)
sum(list1[i], list2[i], list[3]); //Almost, list is not a defined
variable, should have probably been list3[i]
```

17. Define a two-dimensional array named `temp` of three rows and four columns of type `int` such that the first row is initialized to 6, 8, 12, 9; the second row is initialized to 17, 5, 10, 6; and the third row is initialized to 14, 13, 16, 20.

```
int temp[3][4] =
{
    6, 8, 12, 9,
    17, 5, 10, 6,
    14, 13, 16, 20
};
```

18. Given the declaration:

```
char str1[15];
char str2[15]= "Good day";
```

mark the following statements as valid or invalid. If a statement is invalid, explain why.

```
a. str1 = str2;      //Usually an invalid array assignment
b. if(str1 == str2) //Valid comparison, but it is not comparing the
                     //contents of the arrays, but the addresses
   cout << " Both strings are of the same length." << endl;
c. if(strlen(str1) >= strlen(str2))
   str1 = str2; //Again usually not valid in modern compilers
d. if(strcmp(str1, str2) < 0) //Yes. Valid
   cout << "str1 is less than str2." << endl;
```

19. Write a C++ program to accept five integer values from keyword. The five values will be stored in an array using a pointer. Then print the elements of the array on the screen.

```
int *values = new int[5];
for (int i = 0; i < 5; i++)
        std::cin >> values[i];
for (int i = 0; i < 5; i++)
        std::cout << values[i] << "\n";
delete [] values;
```

20. Write a program that asks the user to enter integers as inputs to be stored in the variables 'a' and 'b' respectively. There are also two integer pointers named ptrA and ptrB. Assign the values of 'a' and 'b' to ptrA and ptrB respectively, and display them.

```
int a, b;
std::cin >> a >> b;
int *ptrA = &a, *ptrB = &b;
std::cout << a << b << ptrA << prtB;
```

21. Consider the following statements:

```
struct nameType{
   string first;
   string last;
};

struct dateType{
   int month;
   int day;
   int year;
};

struct personalInfoType{
   nameType name;
   int pID;
   dateType dob;
};

personalInfoType person;
personalInfoType classList[ 100] ;
nameType student;
```

Mark the following statements as valid or invalid. If a statement is invalid, explain why.

a. `person.name.first = "William";` **//valid**

b. `cout << person.name << endl;` **//Not valid, trying to output nameType struct**

c. `classList[ 1] = person;` **//Valid**

d. `classList[20].pID = 000011100;` **//Valid**

e. `person = classList[ 20] ;` **//Valid**

f. `student = person.name;` **//Valid**

g. `cin >> student;` **//Not valid, trying to input to nameType**

h.
```
for (int j = 0; j < 100; j++)
   classList[j].pID = 00000000;
```
//Valid

i. `classList.dob.day = 1;` **//Valid**

j. `student = name;` **//Invalid, there is no name variable declared.**

22. Write a set of functions that take in a vector of ints and calculates the average, max, and min.

```
using namespace std;
double Avg(vector<int> v)
{
        long sum =0;
        for (int i = 0; i < v.size(); i++)
                sum += v.at(i);
        return (double)sum/v.size();
}
```

```
int Max(vector<int> v)
{
	int max = v.at(0);
	for (int i = 0; i < v.size(); i++)
	{
		if (max < v.at(i))
			max = v.at(i);
	}
	return max;
}

int Min(vector<int> v)
{
	int min = v.at(0);
	for (int i = 0; i < v.size(); i++)
	{
		if (min > v.at(i))
			min = v.at(i);
	}
	return min;
}
```

23. Declare an empty `char` vector named `vowels`. Then insert the letters `a,e,i,o,u` vowels into the vector.

```
vector<char> vowels;
vowels = {'a', 'e', 'i', 'o', 'u'};
```

24. Write a function `vectorAverage` that returns the average of the elements of an integer vector. The prototype is written below.

```
double vectorAverage (vector<int> v);

double vectorAverage (vector<int> v)
{
	long sum = 0;
	for (int i = 0; i < v.size(); i++)
		sum += v.at(i);
	return (double)sum/v.size();
}
```

25. Write a function `insertElement` that inserts a given integer x into an integer vector before a given position pos. The prototype is written below.

```
void insertElement (vector<int> &v, int pos, int x);
void insertElement (vector<int> &v, int pos, int x)
{
	v.insert(v.begin()+(pos-1), x);
}
```

`//Didn't have time to finish, did all the bullet points as well`

26. Write a function `replaceVowels` that takes a string as input and replaces every lower-case vowel (aeiou) with an one of the four symbols below:

```
* & # @
```

The symbol used should be selected at random. A sample run is shown below.

```
cout << replaceVowels ("Frodo, look out for the orcs!!!");
Output: Fr&d*, l#*k *@t f&r th# &rcs!!!
```

27. Write a class `Car` that keeps track of the name of the car, the fuel efficiency in miles per gallon (mpg), the number of miles driven, and the number of gallons left in the gas tank. The constructor should take the car name and the fuel efficiency as input. Write the class declaration first and then define the member functions for the class. You'll have to figure out what functions to add based on the sample usage below.

```
Car my_car ( "Ferrari Testarossa", 20 ); //My Ferrari gets 20 miles
to the gallon.
my_car.add_gas (10); //Add 10 gallons of gas to the tank.
my_car.drive (100); //Drive 100 miles.
cout << my_car.get_name( )<< " has driven " << my_car.get_miles( )
     << " miles and has " << my_car.get_gas( )
     << " gallons left in the tank.";
Output: Ferrari Testarossa has driven 100 miles and has 5 gallons
left in the tank.
```