EXAM 1 - Overview

FORMAT

- Bring a scantron
- Some T/F
- Some Mult Choice
- Some Problem Solving
- NOT open notes/book

Topics Covered

1. Basic Input/Output
2. Arithmetic in C++
3. Selection/Repetition
4. Functions
5. User defined header file, using files and enumerator datatype
6. Strings
7. Arrays
8. Searching and sorting
9. Structs
10. Testing

Some tips – avoiding test anxiety

- Get a good nights rest
  - I know this is tough, but you don't think as well without sleep
- Don't skip a meal before an exam
  - Your brain needs protein → try not to eat a high carb meal
- Don't Cram! Pace your studying
  - Try not to put it off until the last minute
  - If you pace yourself → you will be prepared
- Study with classmates so you can compare notes
  - don't discuss the exam just before coming in
  - their anxiety may impact you
- Take deep breaths → relax yourself
  - Thing positive thoughts → remind yourself that you are prepared
- Don't get bogged down on a question
  - answer the questions you know quickly → go back to the others
- Ask Questions
  - Calm yourself before you come in…
- Avoid being late

Basics

- T/F A C++ identifier may not begin with a digit or an underscore.
  True
- What is a literal constant?
  An actual value
- What type of loop should be used for a program segment that should sum a list of positive integers (it is unknown how many inputs the user will provide_?
  A while loop
- What should the LCV be?
  the input number
- What should the sentinel value be?
  a negative input
- When should you initialize the LCV, for each loop?
  While loop - before the loop
  do while - at the top of the loop
  for loop - in the loop statement
- When should you update the LCV, for each loop?
  While loop - at the end of the loop
  do while - at the top of the loop
  for loop - in the loop statement

Output

- What is this symbol <<?
  insertion operator
- What can be on the left side of the insertion operator?
  Left → files, cout
- What can be on the right side of the insertion operator?
  right → expressions, variables, constants, literals
- Which manipulator(s) would you use output in columns ?
  setw()
- What are the escape sequences?
  \n, \t, \a, \\, \', \"
- Which output manipulators would you use to format floating point numbers?
  fixed
  setprecision(n)
  showpoint
- Know how they work together

Input

- What is this symbol >>?
  extraction operator
- What can appear on the left side of the extraction operator?
  cin, files
- What can appear on the right side of the extraction operator?
  variables
- When do you have to use .ignore() when reading in c-strings or strings?
  -in between the extraction operator and a getline or a .get
  -after a .get
- What function do you use to get a line of characters and place it in a string
  getline(fileName or cin, stringName)
- T/F cin always reads directly in from the keyboard
  False

7

Arithmetic

- What is the difference between type coercion and type casting?
  type coercion is when the type is converted implicitly through the use of mixed-mode arithmetic
  example: intNum = (5 * 3) / 2.0;
  type casting is explicitly modifying the type
  example: flNum = (5 * 3) / float (2);
- How would the following statement be evaluated?
  in1 = (fn3 = (in2 = 5) * (4 / 8.0)) * 2;
  in1 = 5, in2 = 5, fn3 = 2.5
- Rewrite the following using combined operators:
  The remainder of n1 divided by (n2 * 12) with the value stored in n1
  n1 %= n2* 12
- Also understand Boolean expressions and the conditional operator.

8

Arithmetic

- What is the order of precedence?

Order of Precedence
()
++, --
! (unary)
* / %
+ -
< <= > >=
== !=
&&
||
= += -= *= /= %=

- How would the following statement s be evaluated?

```
a = 4;
b = 3;
c = --a + 10 * b++;
```

a = 3, b = 4, c = 33;

9

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.
  True
- How do you declare a function?
  prototype
- Where does the prototype go?
  above the int main()
- What is the scope of a local identifier?
  the end of the block of code in which it was declared
- 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.
  True
- T/F In C++, corresponding arguments from a calling function and parameters from a called function must have the same name.
  False
- When would you use a void function?
  when nothing is being returned from the function
  or if you need to return multiple values from the function

10

Functions

- How do you return multiple values from a function?
  by passing by reference
- What is a side effect?
  when a variable is passed by reference and is inadvertently changed by the function being called
- How can we avoid side effects?
  1. pass by constant reference
  2. pass by value
- Parameter passage by reference is used if a parameter's data flow is
  a) one-way, into the function
  b) one-way, out of the function
  c) two-way, into and out of the function
  b & c
- T/F When passing by value data flow is one-way - into the function
  True
- What type of arguments can be sent from the calling function when passing by reference?
  variables

11

Functions

- What type of arguments can be sent from the calling function when passing by value?
  variables, constants, expressions, literals
- T/F All functions should have a return statement.
  False - not void functions
- 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?
  A value returning function
- T/F Functions must have a return data type
  True
- What should the return data type for main be?
  int
- What is the advantage of using functions?
  supports reusability
  makes code more readable
  makes code easier to test
  It keeps code more organized

12

Functions

- What are the three steps in the function process
  1. Declaring a function
  2. Defining the function
  3. Using the function
- Where do each of the steps above occur
  1. a function is declared using a prototype above int main() or the calling function or in a header file which is included above the calling function
  2. A function is defined either below the int main() or in a separate source file
  3. The function can be called within the body of any function
- Declare a function that is passed two integers representing a sum and a count and returns an average.
  float avgInts(int sum, int count);
- Declare a function that returns the sum & avg of an integer array.
  void CalcSumAndAvg(const int myIntAr[], const int AR_SIZE, int& sum, float& avg);
- T/F You can only have 1 return statement in a function
  False
- What is an advantage of passing by value?
  no accidental modification of arguments by the function being called (ie no side effects) & you can pass expressions, literals, and constants

13

Functions

- What is a disadvantage of passing by value?
  Large variables take up a lot of overhead
  (the value of the variable has to be copied and initialized)
  (with large variables time & space penalties become a problem)
- When we pass by reference what is passed to the calling function.
  the address of the variables passed
- What is a disadvantage of passing my reference?
  1. side effects
  2. Hard to tell if the parameter being passed is to be used as input output or both
  3. can only pass in variables
- How do you pass by reference?
  Use the & after the data type

14

ARRAYS

- What is an array?
  a collection of data of the same type (an aggregate data type)
- Is this a valid statement: firstArray = secondArray;
  no
- Is this a valid statement: if(firstArray == secondArray)
  no
- T/F an array is a composite data type.
  True
- Why should we use constants for array size?
  1. improves readability
  2. improves versatility
  3. improves maintainability
- How do we initialize an array?
  either in the declaration using ={}
  or using a for loop

15

ARRAYS

- T/F An individual component of an array cannot be passed as an argument to a function. The entire array must be sent.
  False
- How many different data types can you have in one array?
  none - they all must be the same
- T/F Given the declaration
  int someAr[20];
  int someAr2[20];
  cout << someAr[3]; outputs the 3rd element in the array.
  false
  cin >> someAr[20]; will produce a compiler error
  false
  someAr = someAr2; will transfer all values from someAr2 to someAr
  false - why?

16

ARRAYS

- What is the advantage of passing by reference?
  You can return more than one value.
  Less overhead, so it is faster and takes less memory.
- Which of the following is true about an array?
  a) Arrays are always passed by reference.
  b) The name of an array is the address in memory of the first element.
  c) Array subscripts always begin at 0.
  All of the above
- How would you declare an array of 20 c-strings that can hold up to 11 characters?
  char chAr[20][12];
- How would you compare an array of int.
  using a for loop → compared each index.

17

ARRAYS

- What will this statement do: int item[5]={2,12,1};
  declare an array called item of 5 elements and assign the values 2, 12, 1, 0, 0 to the elements in the array
- What will this statement do: int item[5]={0};
  declare an array called item of 5 elements and assign initialize all the values of the array to 0
- What will this statement do:
  int item[5]={2,12,1,2,9,5};
  give a compiler 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.
  FALSE
- What is stored in the array variable? (e.g. myArray)
  The base address
- What is the base address?
  the address of the first element of the array

18

ARRAYS

```
float gpa[5];  // an array holding 5 grade point averages - INP.& OUT.
// search an array
index = 0;
while(index  <  MAX_ITEMS  &&  !found
   {
        if (item[index] == searchItem)
        {
            found=true;
        }
        else
        {
             index++;
        }
   }
// output the contents of the array
    cout << "\n\nStudent Grade Point Averages\n
    for(int j = 0; j < 5; j++)
    {
       cout << "\nGPA for student " << j+1 <<": " << gpa[j];
    }
```

Know how to use loops with arrays

When do you use a for loop →

When do you use a while loop?

Know how to use a loop to initialize an array too!

ARRAYS

- T/F Arrays can be returned as a return value in a function.
  False
- T/F Arrays must be passed by reference using the &.
  False
- T/F When you pass an array you don't have to include the size in the parameter list for the first dimension.
  True
- How should you pass an array when you don't intend to modify it in the function being called.
  as a constant
- T/F C-Strings are special arrays.
  True
- T/F char name[16]= "Pete"; ⇔ char name[]= "Pete";
  False
- Make sure you understand parallel arrays!

ARRAYS

Which loop and conditions to use for?

#1 - Sum an array

For Loop, 0 to AR_SIZE - 1

#2 - Read from an input file into an array

While Loop, InFile && index < AR_SIZE

#3 - Search for multiple instances of an int

For Loop, 0 to AR_SIZE - 1

#4 - Search for one instance of an int

While Loop, !found && index < AR_SIZE

#5 - Output array contents

For Loop, 0 to AR_SIZE - 1

#6 - Read from a user into an array

While Loop, input != EXIT && index < AR_SIZE

#7 - Initialize an array to the value -1

For Loop, 0 to AR_SIZE - 1

Arrays

- How would you declare parallel arrays that could contain a movie title, genre, and running time (in minutes)? Assume a const AR_SIZE

```
string title[AR_SIZE];
string genre[AR_SIZE];
int 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?

```
index = 0;
while (inFile && index < AR_SIZE)
{
      getline(inFile, title[index]);
      getline(inFile, genre[index]);
      inFile >> time[index];
      inFile.ignore(100, '\n');
      index++;
}
```

- Know how to use files.

Arrays

- 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)?

```
int row;
int col;
row = 0;
while (inFile && row < 5)
{

      for (col = 0; col < 10 ; col++)
      {
          inFile >> scores[row][col];

      }
     row ++;
}
```

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

Strings

- How would you compare C-Strings.
  strcmp
- T/F Strings have null terminators.
  False
- T/F String size is dynamically allocated.
  True
- T/F You can't use getline with a string only with C-strings.
  False
- If we declare 3 strings. This is valid str3 = str1+str2.
  yes
- What will it do?
  concatenates the strings
  str1 = "abc";
  str2 = "def";
  str3 will = "abcdef";

FILES

- What is the 5-step process for using an I/O file?
  1. include the file fstream
  2. Declare the file stream variables
  3. Associate the file stream variables with the I/O sources
  4. Use the file stream variables with >>, or << or other I/O functions
  5. Close the files
- Which loop structure is best for reading in input from a file?
  While → while(inFile)
- When reading in from a file into an array what are the two conditions that should be checked?
  the end of file and the size of the array
  while(inFile && index < AR_SIZE)

HeaderFiles

- T/F You can put executable code in a header file.
  False
- T/F It doesn't matter what order the directives and declarations appear in a header file.
  False
- What order should the directives appear and why?
  Pre-processor directives
  typedefs /enum types
  constants
  function prototypes
- What are these lines in a header file for?
  #ifndef MYHEADER_H_
  #define MYHEADER_H_
  #endif
  they prevent you from replicating definitions
- Does the extension for your header file matter?
  Yes! - it should be .h
- How do you include a user-defined header file?
  #include "myheader.h"

26

Enums/typedefs

- What are the 3 basic categories of datatypes?
  simple
  structured
  address
- What are the 3 basic simple datatypes
  integral → char, short, int, long, & bool
  floating → float, double
  enum → user defined
- What category do arrays fall under?
  structured (which also includes structs, unions, and classes)
- How would you define an enumerated type to represent the seasons?
  enum Seasons {SPRING, SUMMER, WINTER, FALL};
- Why would you use enumerated types?
  they make your code self-documenting → increase readability
  simplifies coding

27

Enums/typedefs

- Since enums are essentially evaluated into integers, can you perform arithmetic on them?
  only if you typecast or assign the result into an integer
- How would your output an enum?
  switch statement (know how to do this)
- What does a typedef do?
  creates an additional name for an already existing data type
- Can you do this?
  typedef float FloatArrayType[100];
  Yes
- How would you use it?
  FloatArrayType myArray;

28

Searches&Sorts

- What are the advantages of a sequential search?
  easy to implement
  array doesn't have to be sorted
- What is the disadvantage of a sequential search?
  it can be inefficient
- What is the advantage of a binary search?
  more efficient
- When would you use a sequential search over a binary search?
  when you don't have to search often AND the list is unsorted
  → binary searches have the overhead of sorting the list first
- Know how to identify the searching and sorting algorithms.
- Know how to implement sequential search

29

Searches&Sorts

- Which search is this?

```
int intAr[AR_SIZE] = { 5, 12, 6, 4, 2};
index=0;
found = false;
ikey = 6;

while (!found  && index < AR_SIZE)
{
      if (intAr[index] == ikey)
      {
        found = true;
      }
      else
      {
        index ++;
      }
}
```

sequential search

30

Searches&Sorts

- Order the sorts discussed in class in order of efficiency
  Bubble Sort
  Selection Sort
  Insertion Sort
- Why is the most efficient sort more efficient?
  because the insertion sort performs only one swap for each pass through the array
- Know how to identify these sorts

31

Searches&Sorts

- Which sort is this?

```
const int AR_SIZE = 5;
int intAr[AR_SIZE] = { 7, 12, 6,14, 22};

for (int i = 1; i <AR_SIZE; ++i)
{
      temp = intAr[i];
      j = i - 1;
      while ( j >= 0 && intAr[j] > temp)
      {
           intAr[ j + 1 ] = intAr[j];
           j = j - 1;
      }
      intAr[ j + 1 ] = temp;
 }
```

Insertion sort

32

Searches&Sorts

Sorting Algorithms

	Buddle Sort	Selection Sort	Insertion Sort
Outer Loop	For Loop	For Loop	For Loop
Outer Loop Condition	Index 0 to AR_SIZE - 1	Index AR_SIZE - 1 to 0	Index 0 to AR_SIZE - 1
Inter Loop	For Loop	For Loop	While Loop
Main Action	Swap adjacent numbers in their correct order	Select the largest number for each loop	Insert each number in their correct location

33

Structs

- What is the advantage of using structs?
  easier to organize related data items.
- What is a member?
  a field within the structure
- Are aggregate operations allowed on structs?
  only assignments
- Can you pass structs by value or reference?
  yes
- Can structs be a return type?
  yes

34

Structs

- Define a struct called DvdRec, that contains the title, genre, and running time.

```
struct DvdRec
{
    string title;
    string genre;
    int 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?

```
cout << movies[9].title;
```

- Be able to write a function that can read into an array of structs or output an array of structs.

35