**CS 170 Computer Science 1: C++**

**Spring Semester 2022**

**Assignment 2**

Due date: Friday, April 19th by 5 pm

Course weighting 20%

File name: AS1\_98S1.DOC

Last revised: Thursday, 7 April 2022 at 12:13 AM

This assignment requires you to solve three programming problems, and to implement your solution in C++. You will be assessed by your final delivery. This is an individual assignment. No collaboration is permitted.

Problems:

1. Functions with variable number of arguments  
   Write a function write with variable number of arguments that takes a string first argument followed by any number of arguments of type double and prints on the screen a string formatted by the rules described below. The first argument may contain ***formats*** in curly braces of the form **{*index*[:*specifier*]}**, where the square brackets show optional parts (that is **:*specifier*** may be missing), and ***index*** is the sequence number of an argument of type double (starting from sequence number 0).

Rules for formatting: In the printed string the curly brackets and their content will be replaced by the argument with the given index, formatted according to the given format specifier. If the format specifier is missing, the argument will be printed with its default format. For example:

write("The number {0} is greater than {1}.", 5.0, -3.0);

will print

The number 5 is greater than -3.

write("There are no format specifiers here.");

will print

There are no format specifiers here.

The format specifiers and their meanings are listed in the following table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Specifier** | **Meaning** | **Format** | **Output for 1.62** | **Output for 2.0** |
| none | default | {0} | 1.62 | 2 |
| c | currency | {0:c} | $1.62 | $2.00 |
| e | scientific | {0:e} | 1.620000e+000 | 2.000000e+000 |
| f | fixed point | {0:f} | 1.620000 | 2.000000 |
| i | round to int | {0:i} | 2 | 2 |

NOTE: an overview of the C++ ios flags is available at http://www.cplusplus.com/reference/ios/ios\_base/fmtflags/

Limitations: You may limit the maximum number of arguments your function can process to a certain value, for example 10.

Suggested extensions:

* add an optional alignment specification in the format , e.g., make the format of the form **{*index*[,*alignment*][:*specifier*]}**, where the brackets **[** and **]** enclose an optional part and ***alignment*** is an integer specifying the width of the field in which the corresponding argument will be printed. If alignment is positive, align to the left, if it is negative, align to the right.
* Accept an optional integer after the specifier letter, specifying the required precision in the output. For example, {0:f2} will print the number 1.6234 as 1.62, but {0:f5} will print it as 1.62340.

[45 points]

1. Function pointers  
   Compare the execution the speed of sorting methods. Write a function measure that uses the built-in function clock\_t clock() to measure the time of sorting an 1-dimensional array.

* To create the 1-dimensional array, your main program should initialize a 2-dimensional array with random numbers between 1 and 100 and pass it to a function named makeOneDimArray, which will copy all elements of the 2-dimensional array row by row to a dynamically allocated 1-dimensional array.
* Then main will call measure, passing as arguments the dynamically allocated 1-dimensional array and a function to sort the array. Function measure will return the time measurement in milliseconds. Run function measure with at least three different sorting functions implementing different sorting algorithms.
* Discuss your approach, the results of the measurements and their comparison in the test report.

Function clock returns the number of clock ticks since the beginning of your program. To measure the elapsed time between two points in the program, you can call it twice, for example:

clock\_t start, end;

start = clock();

. . . . .

end = clock();

The difference between end and start is the number of ticks between the two calls. The constant CLOCKS\_PER\_SEC contains the number of clock ticks per second.

[20 points]

1. Arrays  
   Magic squares. A n x n matrix that is filled with the numbers 1, 2, 3, … n2 is a magic square if the sum of the elements in each row, in each column, and in the two diagonals is the same value. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | 3 | 2 | 13 |
| 5 | 10 | 11 | 8 |
| 9 | 6 | 7 | 12 |
| 4 | 15 | 14 | 1 |

Implement the following algorithm to construct magic n x n squares; it works only if n is odd. Place 1 in the middle of the bottom row. After k has been placed in the (i,j) square, place k+1 into the square to the right and down, wrapping around the borders. However, if you reach a square that has already been filled, then you must move one square up instead. Write and test a function with argument n, which prints a magic square of order n if n is odd.

[35 points]

**Development Requirements**

1. **Constraints**. Coding must use C++ streams for input and output and generate a DOS executable.
2. **Dependencies**. You are encouraged to use global constants, but your program must not declare any global variables, whether of simple data types, structures, arrays, or file streams.
3. Do not use **goto** statements. Do not use automatic typing.
4. **Functions**. Pay special attention to breaking down your solutions to multiple functions. Your functions should be *cohesive* – each function should do one thing, but do it well. There should be *low coupling* between functions – changing the body of one function should not require changing others. All communications between functions should be achieved through parameters and return values, and NOT through global variables.
5. **Standards.** Your programs must meet the programming standards for this course as described in Assignment 1.

Delivery

All your source code files (.cpp and .h) and any data files (if applicable) must be placed in the directory X:\Dropoff\CS\ganchevg\CS170\Assignment2 in a subdirectory YourName. You should name the source files containing your main programs Problem.cpp, Problem2.cpp, and Problem3.cpp.   
NOTE: If for any of the problems these files are missing, (or incorrectly named or in the wrong directory) or the program will not compile and link correctly, I will not be able to grade your work for this problem. Please double-check this. Please do not submit whole Visual Studio projects or solutions!!

You should also submit in a subfolder Documentation of your folder:

1. Grading sheet (supplied) with sections 1, 2 and 3 completed to show what you have done
2. For each problem

* A structure chart showing the modular structure of your program
* Optionally, structure diagrams or pseudocode showing the algorithms design

1. A test report showing:
2. Check-points, with a clearly indicated result (Y or N)
3. Test data and results in a table with three columns:
   * test input (typed)
   * expected results (typed)
   * the actual results of your testing (typed)
4. A brief analysis of any known errors, which the program still produces or a statement that there are no known errors.
5. Sample copies of any printed reports produced by your program (optional, if applicable).

**Grading Schedule**

The assignment will be graded on an A to F scale of grades.

Work which barely meets the minimum requirements and either has problems with usability or readability or does not meet the programming standards will be graded D- to C. Work which shows a useable solution with all the minimum requirements and meets the programming standards will be graded C+ to B+. Work which in addition demonstrates initiative in design and implementation as evidenced by superior user interaction, additional functionality, robustness and reliability will be graded A- to A.

Grading Notes:

1. To get credit for a program feature, it must be coded, tested and documented correctly according to the given standards and be working in all respects.

## A feature that is either not shown on the test plan as tested or does not work correctly will be given no credit. Penalty: points will be deducted for a faulty feature that is shown on the test plan as working.

1. The grading criteria include functionality, non-functional requirements, documentation and development requirements as indicated on the grading sheet. Pay special attention to function design.

Assignment 2 Grading Sheet

Name and email:

*(your full name and email address)*

1. Minimum requirements satisfied *(check the boxes of problems solved)*

* Problem 1
* Problem 2
* Problem 3

2. Additional functionality *(list extensions completed)*

3. Documentation submitted *(check boxes of items attached)*

* Structure charts
* Structure diagrams or pseudocode (optional)
* Test reports
* Source code
* Printed reports (optional)

4. Non-functional requirements grading criteria

- screen layout

- user interaction

- data validation

5. Development requirements grading criteria

- program organization

- dependencies

- data structures

- function/method design

- program layout

- internal documentation

Grade \_\_\_\_