Kishwaukee College Syllabus
CIS 250 - 5001
C++ Programming II
3 Credit Hours, Spring 2018

Course Description

The second course in the C++ language. Abstract data types will be used in the design and implementation of solutions to large-scale problems. Topics include: classes, inheritance, polymorphism, and encapsulation; files and pointers; scope, blocks and dynamic memory; recursion; data structures including stacks, lists, queues, trees; graphs; text processing; and searching and sorting algorithms. Programming assignments will be completed outside of class. IAI CS 912. Three hours lecture/discussion a week. Prerequisite: CIS 150.

Meeting Time and Place

Lecture/Lab:	A-1374	
Time:	12:30 P.M. - 1:45 P.M.	Monday, Wednesday
Dates:	1/17/18 - 5/16/18	
Withdrawal date:	4/27/18	
MLK Birthday observed:	1/15/18	School closed
Spring break:	3/12/18 - 3/18/18	School closed
Faculty development:	3/29/18	School closed
Good Friday:	3/30/18	School closed
Midterm exam:	3/7/18	during class
Final exam:	5/14/18	Noon - 1:50 P.M.

Instructor Information

Instructor:	David G. Klick
Office:	A-1342
Email:	dklick@kish.edu
Phone:	815/825-9337
Website:	kermit.kish.edu/~dklick/
Backup website:	klickfamily.com/david/school/
Desire2Learn:	https://kish.desire2learn.com/
Division Secretary:	815/825-9380 (Brianna Hooker)
Office hours:	M 1:45 P.M. - 2:30 P.M., 5:00 P.M. - 6:00 P.M.
	T 1:45 P.M. - 2:30 P.M., 5:00 P.M. - 6:00 P.M.
	W 10:00 A.M. - 11:00 A.M.
	R 10:45 A.M. - 11:45 A.M.
	other times by appointment

Expected Learner Outcomes

Upon completion of this course, the student will be able to:

1. generate reports and process text
2. use sequential and random access files to save and retrieve data
3. create, use and destroy objects
4. control access to object member data and member functions
5. overload and override methods
6. overload operators
7. create new classes derived from existing classes
8. declare and use virtual functions
9. implement a number of data types including lists, stacks, queues, and trees
10. discuss graphs
11. discuss algorithm complexity
12. create and manage programs with multiple source files

Required Text and Materials

1. Malik, D. (2013). *C++ Programming: Program Design Including Data Structures, 6th edition.* Boston, Massachusetts: Course Technology. [ISBN-10: 1-1335-2632-2, ISBN-13: 978-1133526322]
   4th, 5th, and 7th editions acceptable also
2. Internet access

Breakdown of Course Requirements

10 programs @ 50 points each	500 points
1 midterm exam @ 100 points	100 points
1 final exam @ 100 points	100 points
Total	700 points

Final Grade Determination

A = 90 - 100%	630 points or more
B = 80 - 89.9%	560 - 629 points
C = 70 - 79.9%	490 - 559 points
D = 60 - 69.9%	420 - 489 points
F = below 60%	less than 420 points

Grade reports will not be mailed out. Please check KishSOS,
My Student Info, under Academic Profile, Grades, for grade reports.

Course Procedures

1. Students are expected to attend class sessions on time and prepared (Note: CIS 123 class sessions are optional attendance). Students should bring whatever they need to take notes to every class.
2. Food and beverages are not permitted in the classrooms or labs. See a more detailed policy at http://kermit.kish.edu/~dklick/foodDrinkPolicy.html
3. Cellphones, music players, etc. must be turned off in class.
4. Students are expected to spend **time outside of class** completing assignments.
5. A familiarity with computers and the Windows operating system is expected.
6. Depending on the assignment, both digital and hardcopy versions of assignments may be required for submission. The procedure for submitting digital copies of assignments will be explained in class. Make sure you always keep a copy of all of your assignments. The instructor is NOT responsible for network failures, server failures, or student mistakes.
7. The instructor answers many questions via email. Due to the high volume of requests, submissions, and questions received via email, the instructor must prioritize responses. Most questions will be answered (or at least acknowledged) within 48 hours. If you do not get a response when you expect one, please keep in mind that your email may have failed to reach the instructor, or may have automatically been rejected by an email client or server. Please try to contact the instructor again and possibly use the phone or an in-person visit if email is failing.

Make-up Policy

1. Assignments are to be turned in on time. Assignments which are not turned in on time will not be accepted unless individual arrangements are made **in advance** with the instructor. In unusual cases where late assignments are accepted, the cost of being late is ten percent of the total possible points for every portion of a day late, up to a maximum of three days late. For example, an assignment received twenty-five hours past its due date will lose twenty percent of its total possible point value (because it is two days late). Assignments which are received more than three days (seventy-two hours) late will not be accepted and are not worth any points. Exceptions may be made to this rule if the student contacts the instructor before the due date and makes special arrangements **in advance** with the instructor. All late acceptance decisions of this nature are left solely to the discretion of the instructor. This rule does not apply once answers to an assignment have been distributed or posted. Assignments submitted after answers have been released are worth zero points even if the answers are posted one minute past the due date.
2. Answers to assignments may be posted online, handed out in class, or sent via email by the instructor. Once an answer to an assignment has been released, no further submissions for the assignment will be allowed. This rule supersedes all other rules about when late assignments may be accepted. In general, the instructor will try to wait at least forty-eight hours before posting or distributing solutions, but there is no guarantee, so get your assignments in on time.

3. Tests are to be taken at the day and time scheduled. Failure to take a test at the scheduled time may result in a grade of 0 for that test. In the case of an excusable absence or a genuine emergency, the instructor must be contacted as soon as possible, preferably before the scheduled test, to reschedule the makeup of that test in the Learning Skills Center on the day the student returns to campus.

Attendance Policy

Class attendance is strongly encouraged. You are responsible for whatever was covered in class, whether you are there or not. If you must miss a class, it is your responsibility to contact the instructor and make arrangements for notes, handouts, or announcements that were missed. Although attendance is not counted toward the final grade, there may be coursework which is done during class time which may count toward the final grade and may not be able to be taken outside of class time.

Kishwaukee College Policies and Resources

It is the responsibility of the student to be aware of Kishwaukee College Policies & Resources found on this link: kish.edu/kcsyllabuspolicies

Tentative Weekly Schedule

Please note that this schedule and the topics covered are likely to change. Changes will be announced in class. If you are not able to attend class, it is your responsibility to find out what was covered. A more detailed schedule is provided on the course website. Assignment descriptions and due dates will also be posted on the course web site.

Week	Date	Topics	Reading
1	1/15, 1/17	I/O and formatting **School closed on 1/15 for MLK observance** review syllabus compiling and running programs on the remote server assignment submission using cin, get, and ignore using peek and putback detecting input stream failure and using the clear function formatting output: setprecision, fixed, showpoint, setw, setfill, left, right additional formatting manipulators: dec, hex, oct, showbase, boolalpha, etc. I/O using string objects debugging using cout statements text file input and output	Syllabus Chapter 3
2	1/22, 1/24	User-defined functions function prototypes void vs. value returning functions formal parameter lists parameters vs. arguments default values for parameters	Chapter 6

		function overloading passing by value, passing by reference, and reference variables "returning" more than one value from a function variable/identifier scope (global, local) variable lifetime (static, automatic) how/where variables are stored in memory: stack vs. heap testing functions a brief introduction to recursion	
3	1/29, 1/31	User-defined simple types, namespaces, string objects declaring enumerations declaring variables with an enumeration data type using enumerations (operators, I/O, passing to/from functions) the importance of using enumerations creating and using namespaces (example: kishio I/O library) declaring and using C++ string objects	Chapter 7
4	2/5, 2/7	Arrays and C-style strings declaring one-dimension arrays accessing a member of a one-dimension array initializing a one-dimension array during declaration passing one-dimension arrays to functions (passed by reference) using a loop to iterate through the elements of a one-dimension array common errors trying to access non-existent array elements using an array name as a pointer to the first element using the const keyword to prevent changes to a passed array declaring and using C-style strings (arrays of type char) comparing C-style strings performing I/O with C-style strings declaring and using parallel arrays declaring two-dimension arrays accessing a member of a two-dimension array initializing a two-dimension array during declaration passing two-dimension arrays to functions (passed by reference) using nested loops to iterate through the elements of a two-dimension array arrays of objects (such as C++ string objects) extending array concepts beyond two dimensions	Chapter 8
5	2/12, 2/14	Structs and classes defining a struct or class declaring variables with a struct or class data type accessing members of a struct or class specifying public and private access the differences between a struct and a class passing structs and classes to and from functions creating an array of a struct or class type	Chapters 9, 10

		using assignment with a struct or class built-in operations struct/class scope accessor and mutator functions constructors default constructor destructors static class members (including initialization of static variables) the importance of information hiding UML diagrams of classes	
6	2/19, 2/21	Inheritance and composition overriding member functions constructors of derived and base classes destructors in a derived class header files and header guards protected class members public vs. private vs. protected composition	Chapter 11
7	2/26, 2/28	Exception handling throwing an exception using try/catch blocks rethrowing an exception creating your own exception class exception handling techniques using assertions exceptions vs. assertions error handling techniques (terminate, fix and continue, log and continue)	Chapter 14
8	3/5, 3/7	Pointers, classes, virtual functions, lists, **midterm exam** declaring and initializing pointer variables the address-of operator (&) the dereferencing operator (*) dynamic memory; using new and delete operations on pointer variables creating and using dynamic arrays shallow vs. deep copies functions that objects with dynamic memory should implement/override functions that need special care when using pointers (constructor, copy constructor, assignment operator, destructor) inheritance, pointers, and virtual functions abstract classes and pure virtual functions demonstrate polymorphism array-based lists unordered lists ordered lists **midterm exam**	Chapter 12

	3/12 - 3/18	**School closed for Spring break**	
9	3/19, 3/21	Overloading and templates the reasons for operator overloading restrictions on operator overloading overloading binary operators overloading unary operators overloading binary operators member vs. non-member syntax for overloading functions friend functions overloading the stream insertion operator (<<) overloading the stream extraction operator (>>) specifying post-increment and post-decrement operator overloads overloading the assignment operator overloading the array index operator ([]) function templates class templates	Chapter 13
10	3/26, 3/28	Recursion definition of recursion direct and indirect recursion avoiding infinite recursion recursion vs. iteration when to use (or not use) recursion **School closed on 3/29 for faculty development** **School closed on 3/30 for Good Friday**	Chapter 15
11	4/2, 4/4	Linked lists header and implementation files (revisited) structure of a linked list and its nodes basic implementation of a linked list implementing a copy constructor, assignment operator, and destructor operations on a linked list (insertion, deletion, access elements, display, etc.) basic introduction to algorithm complexity analysis: linked list operations templating a linked list linked list iterators linked list variation: doubly linked list linked list variation: unordered list base class linked list variation: ordered list derived class	Chapter 16
12	4/9, 4/11	Stacks, queues structure of a stack (LIFO) basic implementation of a stack implementing a copy constructor, assignment operator, and destructor operations on a stack (push, pop, peek/top, isEmpty/empty) templating a stack implementing a stack using a linked list	Chapter 17

		implementing a stack using an array stack applications structure of a queue (FIFO) basic implementation of a queue implementing a copy constructor, assignment operator, and destructor operations on a queue (add, remove, isEmpty/empty) templating a queue implementing a queue using a linked list implementing a queue using an array queue applications queue variation: ring buffer queue variation: double ended queue (deque)	
13	4/16, 4/18	Searching and sorting sequential search binary search restrictions on binary search (data must be in sorted order) algorithm complexity analysis of linear and binary search basic sorting algorithm implementation: insertion sort basic sorting algorithm implementation: selection sort basic sorting algorithm implementation: bubble sort advanced sorting algorithm walk-through: quick sort advanced sorting algorithm walk-through: merge sort introduction to binary tree structure and properties binary tree variation: the heap data structure minheaps vs. maxheaps implementing a heap using an array advanced sorting algorithm walk-through: heap sort advanced sorting algorithm walk-through: bogosort/Robsort algorithm complexity analysis of sorting algorithms sorting arrays vs. linked lists	Chapter 18
14	4/23, 4/25	Binary trees properties of a binary tree (revisted) properties of a binary search tree (BST) implementation of a binary search tree operations on a binary search tree: insert, delete, search, traverse, isEmpty avoiding degenerate binary search trees when inserting sorted data: balanced BSTs BST traversal: inorder, preorder, postorder finding a BST's minimum and maximum values finding the successor or predecessor of a node in a BST various ways of handling duplicate values in a BST using recursion vs. iteration when traversing a BST algorithm complexity analysis for BST operations BST applications	Chapter 19
15	4/30, 5/2	Graphs graph terminology: vertex, edge, neighbor, weighted, directed, acyclic, connected, etc.	Chapter 20

		graph data structures: vertex list, edge list, adjacency list, adjacency matrix adding a vertex adding an edge breadth-first traversal depth-first traversal determining if a path exists determining if a graph is connected finding a minimum spanning tree finding a shortest path graph applications	
16	5/7, 5/9	Binary files and random access files writing binary data reading binary data writing to a random access file reading from a random access file advantages and disadvantages of binary files advantages and disadvantages of random access files	Appendix E
Finals	5/14	**Final exam: Noon - 1:50 P.M., Rm. A-1374**	

Addendum

Suggested assignment topics:

1. file I/O, output formatting, implementing an algorithm
2. functions, error checking, exceptions
3. two dimensional array processing
4. array processing, classes
5. inheritance, addition OOP techniques
6. classes, operator overloading
7. linked lists, templates
8. classes, stacks, linked lists
9. modify and use a binary search tree
10. modify and use a graph class