CS 13011 Computer Science IA: Procedural Programming
CS 13012 Computer Science IB: Object-Oriented Programming
Spring 2018

Instructor
Mikhail Nesterenko

e-mail: mikhail AT cs.kent.edu
office: MSB 356
phone: (330) 672-9109
web: http://www.cs.kent.edu/~mikhail/
office hours: TR 1:00-3:00pm

Meetings

<table>
<thead>
<tr>
<th>Sections</th>
<th>Lectures</th>
<th>Labs</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>M 9:55-11:50am, MSB 139</td>
</tr>
<tr>
<td>002</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>M 2:15-4:10pm, MSB 139</td>
</tr>
<tr>
<td>003</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>F 2:15-4:10pm, MSB 139</td>
</tr>
<tr>
<td>004</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>M 9:55-11:50am, MSB 162</td>
</tr>
<tr>
<td>005</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>M 2:15-4:10pm, MSB 162</td>
</tr>
<tr>
<td>006</td>
<td>TR 11:00am-12:15pm, LCM 101</td>
<td>F 2:15-4:10pm, MSB 162</td>
</tr>
</tbody>
</table>

There are no course prerequisites.

Course Overview
The goal of the course is to familiarize the students with programming in a high-level object-oriented language (C++) while studying the main constructs of C++. The students will learn to translate algorithms into correct programs as well as to debug, document and maintain the code.

The C++ constructs covered include: conditional and loop statements, functions, arrays, pointers, object classes, dynamic memory allocation.

Course Organization
The course contains two parts: A and B. To be allowed to enroll in part B, students need to complete part A with the grade of C or above.

Textbook

Class Web Page, Mailing List, Contacting the Instructor
The web page for the class is http://www.cs.kent.edu/~mikhail/classes/csi/ I have a link to this page from my homepage. The web page contains links to the following course materials:
- course syllabus;
- class schedule;
- lecture notes and programming examples used in class;
- mailing list info and archives
- link to the lab website;

There is a mailing list set up for the students taking this course. I am going to send announcements and other class related information to this list. It is very important to be on this list to get the latest
news and updates about the class. You will sign up to the mailing list as a part of your first lab. The subscription instructions are on the course’s webpage. You have to check your mail at least once a day while the school is in session. The simplest way to contact me is via e-mail. I prefer to use the department email address (shown above). If you need to talk to me in person – see me during my office hours or make an appointment via e-mail. When initiating contact by email, make sure the email contains salutation, closing, your name and the course name you are enrolled in.

Lectures
Students are expected to attend each lecture. I do not take attendance, yet attendance and active participation during a lecture is instrumental for course success.

Quizzes
There will be approximately 4 quizzes held during the class: two during part A, and two during part B. Each quiz is on the material of the previous lectures. Each question is worth 1 point. A quiz is held during the first 10 minutes of the class. Late students will not be given extra time to complete the quiz.

Exams
There are two exams. First held in the final week of CSIA. The second is held during the semester’s finals. All exams are closed book, closed notes, and must be individual work. It is expected that you take each exam at the scheduled time, unless you make prior arrangements with me, or have a documented illness, in which case I expect you to contact me as soon as possible. You will be tested on the material we covered in class. The textbook or the slides alone may not be sufficient for adequate preparation for the exams.

Labs
The lectures are complemented by lab sessions. The sections of this course differ in the time of their lab sessions. The lab session is conducted by a lab instructor. Lab attendance and participation is required. Lab sessions are an integral part of the course and lab assignments constitute a significant part of the course grade. The lab policies are stated on the lab website and are to be followed for the success in the lab.

Academic Integrity
Academic dishonesty in any form will be reported to the Office of Student Conduct and penalized up to assigning grade F for the course.

Cooperation on Programming Projects (Labs). You are allowed to discuss projects and solutions with your peers outside the lab. However, you should code the projects individually. This means that you should not look at other students’ programs either on the screen or in printouts. You should not copy other students’ solutions. Joint programming, even in pseudo-code, is not allowed. Cooperation during labs is not allowed. If you have a question during a lab, ask your lab instructor. Do not ask your classmate: you are distracting him/her, you may be getting an incorrect answer and you may be inadvertently involving him/her in joint work.

You should be careful not to give others access to your code. This means that you should not keep your program in a publicly accessible folder, you should not leave your computer unattended, and you should not forget to pick up your printouts.

Grades
Course parts A and B are graded separately. You need to get at least C grade in part A to be allowed to enroll in part B. The grades are calculated as follows.
• quizzes (approx. 2 in each part)  10 points each  
• exam (1 in each part)  100 points  
• lab assignments (approx. 6 in each part)  10 points each

There are no extra credit assignments. The sum of the possible scores on all assignments is considered 100% and your final course grade will be determined as follows – 100-93% A, 92-90 A–, 89-87 B+, 86-83 B, 82-80 B–, 79-77 C+, 70-76 C (note that there is no C- grade), 69-67 D+, 66-60 D, 59-0 F. There will be no curve at the end of the course. Your score will not be rounded up: if you get 66.99% you will get a D not a D+. Thus, you should always be able to determine how well you are doing in the course.

Students with Disabilities

University Policy 3342-3-01.3 requires that students with disabilities be provided reasonable accommodations to ensure their equal access to course content. If you have a documented disability and require accommodations, please contact me at the beginning of the semester to make arrangements for necessary classroom adjustments. Please note, you must first verify your eligibility for these through Student Accessibility Services (contact 330-672-3391 or visit www.kent.edu/sas for more information on registration procedures).

Miscellaneous

Try not to be late for class. Make sure you silence your cellphone. The use of laptops while the class is in session is allowed only with my permission. If you take notes or otherwise cannot avoid using your laptop, obtain my permission within the first two weeks of classes.

There is a possibility of testing out of this course. If you are proficient in C++, notify me within the first week. I will interview you. If you demonstrate sufficient knowledge, you will be allowed to skip this and enroll in a more advanced course.
SDF/Fundamental Programming Concepts

[10 Core-Tier1 hours]
This knowledge unit builds the foundation for core concepts in the Programming Languages Knowledge Area, most notably in the paradigm-specific units: Object-Oriented Programming, Functional Programming, and Event-Driven & Reactive Programming.

Topics:
• Basic syntax and semantics of a higher-level language
• Variables and primitive data types (e.g., numbers, characters, Booleans)
• Expressions and assignments
• Simple I/O including file I/O
• Conditional and iterative control structures
• Functions and parameter passing
• The concept of recursion

Learning Outcomes:
1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion. [Assessment]
2. Identify and describe uses of primitive data types. [Familiarity]
3. Write programs that use primitive data types. [Usage]
4. Modify and expand short programs that use standard conditional and iterative control structures and functions. [Usage]
5. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, the definition of functions, and parameter passing. [Usage]
6. Write a program that uses file I/O to provide persistence across multiple executions. [Usage]
7. Choose appropriate conditional and iteration constructs for a given programming task. [Assessment]
8. Describe the concept of recursion and give examples of its use. [Familiarity]
9. Identify the base case and the general case of a recursively-defined problem. [Assessment]

SDF/Fundamental Data Structures

[12 Core-Tier1 hours]
This unit builds the foundation for core concepts in the Algorithms and Complexity Knowledge Area, most notably in the Fundamental Data Structures and Algorithms and Basic Computability and Complexity knowledge units.

Topics:
• Arrays
• Records/structs (heterogeneous aggregates)
• Strings and string processing

• References and aliasing

• Linked lists

• Strategies for choosing the appropriate data structure

Learning Outcomes:
1. Discuss the appropriate use of built-in data structures. [Familiarity]
4. Compare alternative implementations of data structures with respect to performance. [Assessment]
5. Describe how references allow for objects to be accessed in multiple ways. [Familiarity]
6. Compare and contrast the costs and benefits of dynamic and static data structure implementations. [Assessment]
7. Choose the appropriate data structure for modeling a given problem. [Assessment]