ORIGINAL COURSE IMPLEMENTATION DATE: September 2021
REVISED COURSE IMPLEMENTATION DATE: September 2022
COURSE TO BE REVIEWED (six years after UEC approval)**:** February 2027
Course outline form version: 05/18/2018

OFFICIAL UNDERGRADUATE COURSE OUTLINE FORM

Note: The University reserves the right to amend course outlines as needed without notice.

Course Code and Number: ENGR 153	**Number of Credits:** 4 Course credit policy (105)

Course Full Title: Structured Programming for Engineers
Course Short Title: Programming for Engineers
(Transcripts only display 30 characters. Departments may recommend a short title if one is needed. If left blank, one will be assigned.)

Faculty: Faculty of Applied and Technical Studies	**Department (or program if no department):** Physics

Calendar Description:

Students will learn programming design, data types, functions, and data structures, with a focus on engineering applications.

Note: Students with credit for COMP 152 cannot take this course for further credit.

Prerequisites (or NONE):	B or better in one of Pre-Calculus 12, MATH 093, or MATH 096.
Corequisites (if applicable, or NONE):	None
Pre/corequisites (if applicable, or NONE):	None

Antirequisite Courses *(Cannot be taken for additional credit.)*
Former course code/number:
Cross-listed with:
Dual-listed with:
Equivalent course(s): **COMP 152**
(If offered in the previous five years, antirequisite course(s) will be included in the calendar description as a note that students with credit for the antirequisite course(s) cannot take this course for further credit.)

Typical Structure of Instructional Hours

Lecture/seminar hours	45
Tutorials/workshops	
Supervised laboratory hours	30
Experiential (field experience, practicum, internship, etc.)	
Supervised online activities	
Other contact hours:	
Total hours	**75**

Labs to be scheduled independent of lecture hours: ☒ No ☐ Yes

Special Topics *(Double-click on boxes to select.)*
This course is offered with different topics:
☒ No ☐ Yes *(If yes, topic will be recorded when offered.)*

Independent Study
If offered as an Independent Study course, this course may be repeated for further credit: *(If yes, topic will be recorded.)*
☐ No ☐ Yes, repeat(s) ☐ Yes, no limit

Transfer Credit
Transfer credit already exists: *(See **bctransferguide.ca**.)*
☒ No ☐ Yes
Submit outline for (re)articulation:
☐ No ☒ Yes *(If yes, fill in transfer credit form.)*

Grading System
☒ Letter Grades ☐ Credit/No Credit

Maximum enrolment (for information only): 24

Expected Frequency of Course Offerings:
Fall *(Every semester, Fall only, annually, etc.)*

Department / Program Head or Director:	**Date approved:**	August 2021
Faculty Council approval	**Date approved:**	October 14, 2021
Undergraduate Education Committee (UEC) approval	**Date of meeting:**	February 26, 2022

Learning Outcomes:

Upon successful completion of this course, students will be able to:

- Analyze the behaviour of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion.
- 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, parameter passing, constants, and enumerated types.
- Modify and expand short programs that use standard conditional and iterative control structures and functions.
- Break problems up into sub-problems using functions, when writing programs.
- Describe the concept of dynamic data structures and their uses.
- Discuss the importance of consistent and readable documentation and program style standards in an engineering design context.
- Create readable and maintainable software.

Prior Learning Assessment and Recognition (PLAR)

☒ Yes ☐ No, PLAR cannot be awarded for this course because

Typical Instructional Methods *(Guest lecturers, presentations, online instruction, field trips, etc.; may vary at department's discretion.)*

Lecture and lab.

NOTE: The following sections may vary by instructor. Please see course syllabus available from the instructor.

Typical Text(s) and Resource Materials *(If more space is required, download Supplemental Texts and Resource Materials form.)*

	Author (surname, initials)	Title (article, book, journal, etc.)	Current ed.	Publisher	Year
1.	Savitch, W.	Problem Solving with C++	☒	Pearson	
2.			☐		
3.			☐		
4.			☐		
5.			☐		

Required Additional Supplies and Materials *(Software, hardware, tools, specialized clothing, etc.)*

Typical Evaluation Methods and Weighting

Final exam:	40%	Assignments:	15%	Field experience:	%	Portfolio:	%
Midterm exam:	%	Project:	%	Practicum:	%	Other:	%
Quizzes/tests:	25%	Lab work:	20%	Shop work:	%	Total:	100%

Details (if necessary):

Typical Course Content and Topics

These are the provincially mandated course outcomes for this course. The programming language must be C or C++ and include:

1. Program comprehension
   - Analyze and explain the behaviour of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion.
2. Program design and implementation
   - 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, parameter passing, constants, and enumerated types.
3. Primitive data types
   - Identify and describe the appropriate use of primitive data types
   - Write programs that use primitive data types Conditional and Iterative Constructs
   - Choose appropriate conditional and iteration constructs for a given programming task
   - Modify and expand short programs that use standard conditional and iterative control structures and functions.
4. Functions
   - Describe the purpose of function definitions
   - Describe the importance of modularization when solving problems
   - Break problems up into sub-problems using functions, when writing programs
5. Advanced data structures
   - Write programs that use each of the following data structures: arrays, structs, strings.
   - Write programs that use pointers for dynamic memory allocation and release
   - Describe the concept of dynamic data structures and their uses

- Recognize the risks of pointers.

6. Code quality
   - Apply consistent documentation and program style standards
   - Describe the importance of consistent documentation and program style standards
   - Create readable and maintainable software using conventions like documentation and program style standards

These will be implemented through the following topic areas:

1. Basics and definitions; computing as an engineering design problem
2. Data types and representations
3. Operations and library functions
4. Decision making options
5. Looping options
6. Functions and passing variables
7. Arrays
8. Pointers
9. String manipulation
10. Introduction to microprocessors