ORIGINAL COURSE IMPLEMENTATION DATE: September 2021
REVISED COURSE IMPLEMENTATION DATE: September 2026
COURSE TO BE REVIEWED (six years after UEC approval)**:** March 2032
Course outline form version: 29/08/2024

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:** 3 Course credit policy (105)
Course Full Title: Structured Programming for Engineers **Course Short Title:** Programming for Engineers	
Faculty: Faculty of Applied and Technical Studies	**Department/School:** 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:

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	45
Supervised laboratory hours (computer lab)	30
Total hours	**75**

Scheduled Laboratory Hours

Labs to be scheduled independent of lecture hours: **No**

Course Details

Special Topics course: **No**
(If yes, the course will be offered under different letter designations representing different topics.)

Directed Study course: **No**
(See policy 207 for more information.)

Grading System: **Letter grades**

Delivery Mode: **May be offered in multiple delivery modes**

Expected frequency: **Fall only**

Maximum enrolment (for information only)**: 24**

Prior Learning Assessment and Recognition (PLAR)

PLAR is available for this course.

Yes

Transfer Credit *(See **bctransferguide.ca**.)*

Transfer credit already exists: **Yes**

Submit outline for (re)articulation: **Yes**
(If yes, fill in transfer credit form.)

Department approval	**Date of meeting:**	September 2025
Faculty Council approval	**Date of meeting:**	October 9, 2025
Undergraduate Education Committee (UEC) approval	**Date of meeting:**	March 27, 2026

Learning Outcomes *(These should contribute to students' ability to meet program outcomes and thus Institutional Learning Outcomes.)*

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

1. Analyze the behaviour of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion.
2. 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. Augment short programs that use standard conditional and iterative control structures and functions.
4. Break problems up into sub-problems using functions, when writing programs.
5. Describe the concept of dynamic data structures and their uses.
6. Discuss the importance of consistent and readable documentation and program style standards in an engineering design context.
7. Create readable and maintainable software.
8. Explain the importance of Engineers and Geoscientists BC guidelines for Indigenization and Reconciliation and how they apply to the professional practice of engineering.
9. Explain Engineers and Geoscientists BC programs and initiatives for Equity, Diversity, and Inclusion.

Recommended Evaluation Methods and Weighting *(Evaluation should align to learning outcomes.)*

Final exam:	40%	Quizzes/tests/midterm:	25%	Lab work:	20%
Assignments:	15%		%		%

Details:

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

Typical Instructional Methods *(Guest lecturers, presentations, online instruction, field trips, etc.)*

Lecture and lab.

Texts and Resource Materials *(Include online resources and Indigenous knowledge sources. Open Educational Resources (OER) should be included whenever possible. If more space is required, use the Supplemental Texts and Resource Materials form.)*

	Type	Author or description	Title and publication/access details	Year
1.	Textbook	Savitch, W.	Problem Solving with C++	
2.	Textbook	Stephen Prata	C++ Primer Plus, Sixth Edition	2015
3.				
4.				
5.				

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

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