THE ACADEMY FOR GIFTED CHILDREN – P.A.C.E.
CURRICULUM OUTLINE AND EVALUATION
GRADE 8 – COMPUTER SCIENCE III

OVERALL EXPECTATIONS:
By the end of this course, students will:

- solve problems and create computational representations of mathematical situations using coding concepts and skills;
- describe fundamental programming concepts and constructs;
- use proper code maintenance techniques and conventions when creating computer programs;
- demonstrate the ability to use software development tools to design and write a computer program;
- demonstrate the ability to use different data types in computer programs;
- demonstrate the ability to use control structures and simple algorithms in computer programs;
- demonstrate the ability to use subprograms within computer programs;
- design software solutions to meet a variety of challenges;
- design algorithms according to specifications;
- apply a program development life-cycle model to a program development project;
- identify and describe impacts of coding and of emerging technologies, such as artificial intelligence systems, on everyday life, including skilled trades;
- analyze contributions to science and technology from various communities.

UNITS OF STUDY

Unit 1: Introduction to Computer Science

By the end of the Unit, students will:

- use correct terminology to describe programming concepts;
- describe the types of data that computers can process and store (e.g., numbers, text);
- write keyboard input and screen output statements that conform to program specifications;
- use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation;
- demonstrate an understanding of an integrated software development environment and its main components (e.g., source code editor, compiler, debugger);
- demonstrate the ability to correct syntax, logic, and run-time errors in computer programs;
- describe fundamental concepts in programming, that are similar across languages;
- describe the steps of editing, compiling, linking, and editing for creating a program in C;
- describe the benefits and uses of documentation when writing programs;
- describe the features and benefits of modular programming.

Unit 2: Variables and Math

By the end of the Unit, students will:

- use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs;
- demonstrate an understanding of how a computer uses various systems (e.g., binary, hexadecimal, ASCII, Unicode) to internally represent data and store information;
- write programs that incorporate user input, processing, and screen output;
- determine the expressions and instructions to use in advanced programming statements, taking into account the order of operations (e.g., precedence of arithmetic operators, assignment operators, and relational operators);
- demonstrate the ability to use integer division and resultant remainders in computer programs;
- demonstrate an understanding of type conversion (e.g., string-to-integer, character-to-integer, integer-to-character, floating point-to-integer, casting in an inheritance hierarchy);
- demonstrate an understanding of the limitations of finite data representations (e.g., integer bounds, precision of floating-point real numbers, rounding errors) when designing algorithms;
- design algorithms that require precision and accuracy when rounding numbers;
- distinguish between signed and unsigned integer types and use them when storing variables in memory;
- select the appropriate numeric data types based on their limitations;
- choose the correct type of operator for different uses;
- describe the characteristics and uses of the character data type, and utilize properties of character input and output.

Unit 3: Decision Structures

By the end of the Unit, students will:

- read and alter existing code involving the analysis of data in order to inform and communicate decisions, and describe how changes to the code affect the outcomes and the efficiency of the code;
- demonstrate the ability to use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in computer programs;
- write a program that includes a decision structure for two or more choices;
- demonstrate the ability to validate a program using test cases;
- describe the function of a variety of logical operators;
- explain the function and purpose of the switch statement, and when it can be used in a program;
- select and use the appropriate functions for testing characters;
- identify situations where one would need to test for characters.

Unit 4: Repetition Structures

By the end of the Unit, students will:

- identify situations in which decision and looping structures are required;
- use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program;
- write algorithms with nested structures;
- design simple algorithms according to specifications;
- describe the function and purpose of the goto statement;
- utilize understanding of the limitations of the goto statement when coding;
- identify appropriate cases for use of increment and decrement operators;
- identify appropriate uses for the 'for' loop;
- describe how the increment and decrement operators may be used in 'for' loops;
- determine when to use the prefix and postfix forms of the increment operator;
- apply both the prefix and postfix forms of the increment operator to their 'for' loop code;
- apply nested loops to a variety of problem-solving situations;
- use the do-while statements while coding.

Unit 5: Functions

By the end of the Unit, students will:

- use sequence, selection, and repetition control structures to create programming solutions;
- demonstrate the ability to use existing sub-programs (e.g., random number generator, substring, absolute value) within computer programs;
- describe what variable scope is;
- write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs;
- work independently to design and write functioning computer programs;
- use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program;
- use the features of a software development environment to debug programs and create functioning computer programs;
- work independently, using the Help function, to resolve syntax issues while programming;
- demonstrate the ability generate pseudo-random integers using functions;
- describe functions and uses of the continue statement;
- explain what math library functions are;
- create a function definition for different applications;
- use function prototypes in modular programs.

Unit 5: The Program Development Life Cycle (PDLC) – ongoing throughout the year

By the end of this Unit, students will:

- apply the process of mathematical modelling to represent, analyse, make predictions, and provide insight into real-life situations;
- use a visual problem-solving model (e.g., IPO [Input, Process, Output] chart) to plan the content of a program;
- represent the structure and components of a program using industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode);

- describe the phases (i.e., problem definition, analysis, design, writing code, testing, implementation, maintenance), milestones (e.g., date of completion of program specification), and products (e.g., specification, flow chart, program, documentation, bug reports) of a program development life cycle;
- outline and apply strategies to encourage ethical computing practices at home, at school, and at work.

ASSESSMENT AND EVALUATION

ASSESSMENT FOR LEARNING

	Term I	Term II	Term III
Knowledge & Understanding	Output Coding Exercise #1 (Student Product) Variables Design a Program #1 (Obs.) Term I Check-in (Discussion)	Decisions Coding Exercise #3 (Student Product) Decisions Design a Program #4 (Obs.) Term II Check-in (Discussion)	Functions Design a Program #6 (Obs.) Term III Check-in (Discussion)
Application	Output Coding Exercise #1 (Student Product) Variables Design a Program #1 (Obs.) Term I Check-in (Discussion)	Decisions Coding Exercise #3 (Student Product) Decisions Design a Program #4 (Obs.) Term II Check-in (Discussion)	Functions Design a Program #6 (Obs.) Term III Check-in (Discussion)
Thinking/Inquiry	Output Coding Exercise #1 (Student Product) Variables Design a Program #1 (Obs.) Term I Check-in (Discussion)	Decisions Coding Exercise #3 (Student Product) Decisions Design a Program #4 (Obs.) Term II Check-in (Discussion)	Functions Design a Program #6 (Obs.) Term III Check-in (Discussion)
Communication	Output Coding Exercise #1 (Student Product) Variables Design a Program #1 (Obs.) Term I Check-in (Discussion)	Decisions Coding Exercise #3 (Student Product) Decisions Design a Program #4 (Obs.) Term II Check-in (Discussion)	Functions Design a Program #6 (Obs.) Term III Check-in (Discussion)

ASSESSMENT AS LEARNING

	Term I	Term II	Term III
Knowledge & Understanding	In-Class Examples (Obs.) Exit Cards (Obs.)	Input Design a Program #3 (Obs.) In-Class Examples (Obs.) Exit Cards (Obs.)	In-Class Examples (Obs.) Exit Cards (Obs.)
Application	In-Class Examples (Obs.) Exit Cards (Obs.)	Input Design a Program #3 (Obs.) In-Class Examples (Obs.) Exit Cards (Obs.)	In-Class Examples (Obs.) Exit Cards (Obs.)
Thinking/Inquiry	In-Class Examples (Obs.) Exit Cards (Obs.)	Input Design a Program #3 (Obs.) In-Class Examples (Obs.) Exit Cards (Obs.)	In-Class Examples (Obs.) Exit Cards (Obs.)
Communication	In-Class Examples (Obs.) Exit Cards (Obs.)	Input Design a Program #3 (Obs.) In-Class Examples (Obs.) Exit Cards (Obs.)	In-Class Examples (Obs.) Exit Cards (Obs.)

ASSESSMENT OF LEARNING (EVALUATION)

	Term I	Term II	Term III
Knowledge & Understanding	25% • Variables Coding Exercise #2 (Obs.) • PDLC Test (Student Product)	25% • Variables Design a Program #2 (Student Product) • Flowcharts #1 (Discussion)	25% • Flowchart #2 (Discussion)

	• IPO Assignment (Student Product) • Intro to C Coding Assessment (Discussion) • Unit 1 Test (Student Product)	• Loops Coding Exercise #4 (Obs.) • Unit 2 Test (Student Product)	• Loops Design a Program #5 (Student Product) • Math Functions Coding Exercise #5 (Obs.) • Final Project (Student Product)
Application	25% • Variables Coding Exercise #2 (Student Product) • PDLC Test (Student Product) • IPO Assignment (Student Product) • Intro to C Coding Assessment (Discussion) • Unit 1 Test (Student Product)	25% • Variables Design a Program #2 (Student Product) • Flowcharts #1 (Discussion) • Loops Coding Exercise #4 (Obs.) • Unit 2 Test (Student Product)	25% • Flowchart #2 (Discussion) • Loops Design a Program #5 (Student Product) • Math Functions Coding Exercise #5 (Obs.) • Final Project (Student Product)
Thinking/Inquiry	25% • Variables Coding Exercise #2 (Student Product) • PDLC Test (Student Product) • IPO Assignment (Student Product) • Intro to C Coding Assessment (Discussion) • Unit 1 Test (Student Product)	25% • Variables Design a Program #2 (Student Product) • Flowcharts #1 (Discussion) • Loops Coding Exercise #4 (Obs.) • Unit 2 Test (Student Product)	25% • Flowchart #2 (Discussion) • Loops Design a Program #5 (Student Product) • Math Functions Coding Exercise #5 (Obs.) • Final Project (Student Product)
Communication	25% • Variables Coding Exercise #2 (Student Product) • PDLC Test (Student Product) • IPO Assignment (Student Product) • Intro to C Coding Assessment (Discussion) • Unit 1 Test (Student Product)	25% • Variables Design a Program #2 (Student Product) • Flowcharts #1 (Discussion) • Loops Coding Exercise #4 (Obs.) • Unit 2 Test (Student Product)	25% • Flowchart #2 (Discussion) • Loops Design a Program #5 (Student Product) • Math Functions Coding Exercise #5 (Obs.) • Final Project (Student Product)
Total	**100%**	**100%**	**100%**

Final Score = (Term I + Term II + Term III) ÷ 3

RESOURCES

- CodeBlocks (free download)
- Digital Textbooks (free download)
- Ontario Mathematics Guidelines 2020
- Ontario Science and Technology Guidelines 2022
- Locally Designed Resources by P.A.C.E.