COURSE TITLE

Computer Programming C++

LENGTH

One Semester
Grades 10-12

DEPARTMENT

Computer Department
Barbara O'Donnell, Supervisor

SCHOOL

Rutherford High School

DATE

Spring 2017

COMPUTER PROGRAMMING C++

I. Introduction/Overview/Philosophy

C++ is one of the most widely used programming languages for developing portable application software. This course introduces students to problem-solving and program development using object-oriented design, structured programming techniques, and the C++ programming language. Specific topics include data types, control statements, functions, argument passing, arrays, and strings. Students will design, construct, and test programs with primarily scientific, mathematical, and business applications.

II. Objectives

Course Outline:

I. Preview Computer Science
- A. Understand that computer science is interdisciplinary
- B. Discuss the historical development of computers
- C. Know what constitutes computer hardware/software
- D. Observe differences between types of computer languages
- E. Discuss the evolution of C++
- F. Computer and Information Ethics
- G. Compiling a simple program
- H. The compilation process
- I. Algorithms

II. Numbers and Objects
- A. Vocabulary of C++
  1. Reserved words
  2. Library identifiers
  3. Basic program components
- B. Number types
  1. Numeric range and precision
  2. Variable definition
  3. Output statement
- C. Input
  1. Format
  2. Prompt
  3. Buffered / Failed
- D. Assignment
  1. Expression
  2. Right into left
  3. Roundoff errors
  4. Types / Casts
  5. Combining assignment and arithmetic
  6. Incrementing / Decrementing

E. Constants
    1. Definition / Format
    2. Proper use
    3. Enumerated types
F. Arithmetic
    1. Symbols / Meanings
    2. Order of operations
    3. Modulus
    4. Integer division
    5. Unbalanced parentheses
    6. Mathematical functions
G. Strings
    1. String variables
    2. String functions
    3. Concatenation
    4. Characters and strings
    5. Formatted output

III. Control Flow
A. The If statement
    1. Purpose
    2. Block statement
    3. Brace layout
B. Relational operators
    1. Notation / Description
    2. = vs ==
    3. Compile errors
    4. Comparison of floating point numbers
    5. ASCII and Comparison of strings
C. Multiple alternatives
    1. Two-way branch / branches
    2. Order of Conditions
    3. The switch statement
    4. Dangling else
D. Nested branches
    1. Two-level decision process
    2. Multi-level decision process
E. Boolean operations
    1. && / || / !
    2. Left to right evaluations / Short-circuiting
    3. Multiple relational operators
    4. Combining && and || Conditions
    5. DeMorgan's Law
F. The "while" loop
    1. Format
    2. Infinite loops
    3. Off-by-one error

G. The “for” loop
    1. Format
    2. Symmetric and asymmetric bounds
    3. Count iterations
    4. Incorrect use of !=
H. The “do” loop
    1. Purpose
    2. Initialization of values
I. Nested loops
    1. Patterns
    2. Processing Inputs

IV. Subprograms: functions for problem solving
A. Purpose
    1. Rationale for program design
    2. Use of functions
B. User-defined functions
    1. Built-in vs user-defined
    2. Function declarations
C. Implementing functions
    1. Name and type
    2. Call to function
    3. Reuse
    4. Side effects
D. Function comments
    1. Relevance
    2. Pre-condition / Post-condition
    3. Identify parameters
    4. Return values
E. Return values
    1. Termination of function
    2. Missing return value
F. Parameters
    1. Formal / Actual
    2. Call by value and call by reference
    3. Meaningful Names
    4. Type Mismatch
E. Scope of identifiers
    1. Local / global variables

V. Arrays
A. Basic concept and notation for arrays
    1. Definition and usage
B. Input/output of arrays
C. Arrays in functions
    1. Array parameters
D. Searching an array
E. Selection sort

Student Outcomes:

Upon completion of the course, students will demonstrate the ability to:

1. describe basic computer organization.
2. identify various types of computer programming languages.
3. demonstrate problem-solving skills necessary for algorithm development.
4. incorporate modular and object-oriented design into problem-solving tasks.
5. develop C++ programs using structured and object-oriented programming techniques.
6. construct programs using the C++ programming language syntax.
7. test C++ programs to verify correctness.
8. formulate, manipulate, and evaluate arithmetic expressions.
9. identify the C++ language data types.
10. declare C++ variables.
11. manipulate C++ variables.
12. construct C++ output statements.
13. incorporate modular and object-oriented design through the utilization of functions and methods.
14. recognize selection structures.
15. design, construct, and implement selection structures in C++ programs.
16. recognize looping structures.
17. design, construct, and implement looping structures in C++ programs.
19. declare, construct, and process one-and two-dimensional arrays.
20. declare, initialize, and manipulate pointer variables.
21. recognize the use of pointers for call-by-reference functions.
22. utilize string processing techniques and functions.
23. describe object-oriented program abstractions and design.
24. analyze and implement C++ classes.
25. implement user-designed functions.
26. examine computer and information ethics
27. explore history of the computer.

New Jersey Student Learning Standards

TECHNOLOGY

Standard 8.1: Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaboratively and to create and communicate knowledge.

Standard 8.2: Technology Education, Engineering, and Design: All students will develop an understanding of the nature and impact of technology, engineering, technological design, and the designed world, as they relate to the individual, global society, and the environment.

> Strand E. Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.

21ST CENTURY LIFE AND CAREERS

Standard 9.2: Career Awareness, Exploration, and Preparation

Standard 9.3 – Career & Technical Education (CTE)
Pathway: Programming & Software Development (IT-PRG)

III. Proficiency Levels

This course is open to grades 10-12. A passing grade in Computer Programming 1 is required to take this course.

IV. Methods of Assessment

Student Assessment

The teacher will provide a variety of assessments during the course of the year. Among these are homework, laboratory exercises, teacher made tests and quizzes, and long-term projects.

Curriculum/Teacher Assessment

The teacher will provide the subject area supervisor with suggestions for changes on an ongoing basis.

V. Grouping

The prerequisite for Computer Programming C++ is successful completion of Computer Programming 1.

VI. Articulation/Scope & Sequence/Time Frame

Course length is one semester and is offered to students in grades 10-12.

VII. Resources

Adams, Joel, Leestma, Sanford, and Nyhoff, Larry C++ An Introduction to Computing Upper Saddle River, NJ: Prentice Hall, 1995.

Astrachan, Owen L. A Computer Science Tapestry New York: McGraw Hill Inc., 1996.

Bergin, Joseph, Stehlik, Mark, Roberts, Jim, and Pattis, Richard C++ A Gentle Introduction to the Art of Object-Oriented Programming New York: John Wiley and Sons, Inc., 1997.

Dale, Nell A Laboratory Course in C++ Sudbury, MA. Jones and Bartlett Publishers Inc., 1997.

Horstmann, Cay, Budd, Timothy, Big C++ 2nd Edition: Wiley & Sons Inc., 2009.

Lambert, Kenneth and Nance, Douglas Understanding Programing and Problem Solving with C++. Boston: PWS Publishing Company, 1997.

VIII. Methodologies

Much of the class time is spent in lab work and on programming problems to be completed. When group instruction is necessary, topics are taught using the computer projection system, in conjunction with student classwork.

IX. Suggested Activities

- Laboratory programming problems
- Class presentations
- Cooperative programming projects

X. Interdisciplinary Connections

Connections are made to mathematics by using a variety of arithmetic formulas, as well as higher mathematical concepts. Connections are also made to the disciplines of art and English, by means of incorporation of theses ideas into programming projects.

XI. Differentiating Instruction for Students with Special Needs: Students with Disabilities, English Language Learners, and Gifted & Talented Students

Differentiating instruction is a flexible process that includes the planning and design of instruction, how that instruction is delivered, and how student progress is measured. Teachers recognize that students can learn in multiple ways as they celebrate students' prior knowledge. By providing appropriately challenging learning, teachers can maximize success for all students.

Examples of Strategies and Practices that Support:

Students with Disabilities

- Use of visual and multi-sensory formats
- Use of assisted technology
- Use of prompts
- Modification of content and student products
- Testing accommodations
- Authentic assessments

Gifted & Talented Students

- Adjusting the pace of lessons
- Curriculum compacting
- Inquiry-based instruction
- Independent study

- Higher-order thinking skills
- Interest-based content
- Student-driven
- Real-world problems and scenarios

English Language Learners

- Pre-teaching of vocabulary and concepts
- Visual learning, including graphic organizers
- Use of cognates to increase comprehension
- Teacher modeling
- Pairing students with beginning English language skills with students who have more advanced English language skills
- Scaffolding
    - •word walls
    - •sentence frames
    - •think-pair-share
    - •cooperative learning groups

XII. Professional Development

The teacher will continue to improve expertise through participation in a variety of professional development opportunities.

XIII. Curriculum Map

Class	September	October	November	December	January
Computer Programming C++	• Sec(1.4-1.8) • Input/Output/Header files • Lab#1 • Test(1.4-1.8) • Sec (2.1-2.5) • Write Code in C++ • Data Types and Output/Format • Number Types • Input • Assignment Statements • Constants • Arithmetic • Lab#2 • Test(Sec2.1-2.5)	• Sec(2.6)(Strings) • Variables • Substrings • Concatenation • Formatted Output • Lab#3 • Test(Strings) • Sec(3.1-3.5) • The If Statement • Relational Operators • Multiple Alternatives • Nested Branches • Boolean Operations • Sample Programs (from CodingBat)	• Lab#4 • Test(3.1-3.5) • Sec(3.6-3.8) • The While Loop • The For Loop • The Do Loop • Samples from CodingBat • Lab#4 • Nested Loops • Test (3.6-3.8) • Sec (4.1-4.5) • Functions as Black Boxes • Implementing Functions • Function Comments • Return Values • Parameters • Reference Parameters	• Lab#5 • Quiz(4.1-4.5) • Sec(4.6-4.10) • Variable Scope and Global Variable Refinement • Side Effects • Procedures • Test (Chap4) • Random Numbers • Simulated Gaming Programs • Lab#6 • Test(Random Numbers)	• Sec(6.4) • Parameters and Return Values • Removing and Inserting Array • Elements • Reverse Orders • Searching / Sorting Orders • Lab#7 • History of Computers • Computer and Information Ethics