Introduction to Computer Programming

Basic Course Information

Title:
Introduction to Computer Programming

Length of course:
Full Year

Subject Area:
College-Preparatory Elective ("g")

Grade Levels:
9^{th}, 10^{th}, 11^{th}, 12^{th}

Course learning environment:
Classroom Based

Course Description

Course Overview:
There are 3 main goals of Introduction to Computer Programming:

1. To provide students with the basic concepts/principles of most programming languages.
2. To prepare students to take AP Computer Science A.
3. To encourage and provide guidance for students interested in pursuing a degree or career that uses programming.

This course is designed for students with little or no prior knowledge of programming. The intent is that the course could be taught in any programming language and could change from year to year based on the language used on the AP Computer Science A exam, or as industry standard changes. There will be a focus on good programming practices, both in formatting as well as applying algorithms and problem-solving skills.

Prerequisites:
Algebra 1

Co-requisites:
None

Course Content:
Student will demonstrate knowledge in 4 ways:

1. Written handouts will be completed with each unit to practice the content covered. These will include performing computer arithmetic, identifying program errors, and determining the purpose and output of a program.
2. Programming assignments written on the computer will be completed with each unit to demonstrate application of content and problem-solving skills. Many programs will incorporate math topics learned in Algebra 1.

3. Quizzes will be administered weekly to check for understanding. Quizzes will be hand-written and include short answer questions.
4. Tests will be administered at the end of each unit to demonstrate mastery. Tests will include multiple choice questions, short answer questions, and programming questions.

Unit 1: How a computer works

Students will start the year learning the different parts of the computer and how they communicate. Students will learn the purpose of the processor and memory, and learn several examples of both input and output devices. The binary number system (machine language) will also be covered. Students will learn to convert between decimal numbers and binary numbers and how to add binary numbers. Machine language will be discussed again in Unit 3 when students learn to compile and run a program. Hexadecimal numbers and how they are related to binary numbers will also be covered. Hexadecimal numbers will be seen again in Unit 2 when students learn to create a webpage.

Assignments for this unit will include:

- Complete a worksheet reviewing the parts of a computers, how memory is measured, and listing examples of input and output devices.
- Complete worksheets to practice converting between decimal, binary, and hexadecimal numbers.
- Create a "secret message" and encode it using binary numbers. Trade with a classmate and decode each other's messages.

Unit 2: Focus on the details: HTML

Because program syntax is never ambiguous, it is important that students learn that their code must follow syntax rules exactly in order to work correctly. To demonstrate and practice this idea, students will learn some very simple HTML concepts. Students will learn to create a webpage using only a text editor. Tags that will be learned include formatting the size, color, and font of text, inserting images, changing the background color, bulleted and numbered lists, tables, and linking to other pages. Students are encouraged to use the internet to look up other tags they are interested in learning.

Assignments for this unit will include:

- Complete a worksheet reviewing the HTML tags learned in class.
- Create a short webpage to practice using HTML tags.
- Create a website about a member of your family, including pages with biographical information, a family tree, and information about that person's place of birth.

Unit 3: Input/Output and numeric data types

This unit will begin with learning how to use an integrated development environment (IDE) and how an IDE works. Students will learn how to output text to the screen, including escape characters. Numeric variables will be introduced, both integer and floating-point types. Students will learn how to get user input and assign those values to variables.

Programming assignments for this unit will include:

- Design and output a "house" made of symbols including escape characters.
- Get user input for the 3 sides of a triangle and output its perimeter.
- Get user input for a person's age and then output that age in days, hours, and minutes.

Unit 4: Mathematical Operations with mixed data types
This unit will focus on how the computer does arithmetic. Students will practice the order of operations and will learn to do arithmetic with a mix of integer and floating-point types. Integer division and the idea of "truncating" will be introduced. Students will learn the function and purpose of the modulus operator (%) and how to use its result. The modulus operator will be revisited in later units when determining if a number is even or odd and when checking for divisibility.

Programming assignments for this unit will include:

- Use the modulus operator to calculate a customer's change in terms of number of quarters, dimes, nickels, and pennies.
- Convert temperature from Fahrenheit to Celsius.
- Calculate the total cost of a stock, given the number of shares and the price per share as a mixed number.

Unit 5: Types of Errors and Type casting
Students will learn about the 3 types of errors that can occur in programming: syntax errors, execution errors, and intent errors. There will be a focus on when in the programming process these errors appear as well as discussing common examples of each type of error, such as improper punctuation or off-by-one errors. Given a piece of code, students will be able identify the type of error (if any) and provide multiple ways of fixing that error. Type casting (converting from one primitive data type to another) will also be covered and will be incorporated into programs that have multiple numeric data types.

Programming assignments for this unit will include:

- Calculate the averaging of floating-point numbers, but express the average as an integer.
- Calculate your cell phone bill when you go over your data limit.
- Use type casting to round a floating-point number to its nearest integer.
- Given a 3 digit integer, determine the hundreds digit, tens digit, and ones digit.

Unit 6: Built-in Math functions and Formatting Floating-Point numbers
Students will learn how to use some basic math functions such square root, absolute value, power, and random. These functions will be used to perform and execute more complicated calculations. The concepts of return type and arguments/parameters will also be introduced, and later revisited in Units 10 and 11. Students will also learn how to format their output so that only a certain number of decimal places are displayed.

Programming assignments for this unit will include:

- Calculate the bill at a fast-food restaurant and express the total to 2 decimal places.
- Given the radius, calculate the volume of a ball.
- Given 2 points, calculate the distance between them.
- Generate a random dice roll.
- Given the purchase price, down payment, interest rate, and length of loan, calculate the monthly mortgage payment.

Unit 7: Conditional Statements and Input Validation
Students will learn how to write conditional statements so that a piece of code is only executed when a particular condition is met. The structures that are covered include if, if else, else if chain, and switch statements. Students will learn how to use relational operators to create boolean expressions. The boolean operators ! (not), && (and), and || (or) will be introduced, and students will learn the conditions under which each of the operators evaluates to true or false. Students will learn several ways to write execution equivalent conditional statements, and understand the pros and cons of each way. Conditional statements will also be used to validate user input and display an appropriate error message.

Programming assignments for this unit will include:

- Determine if the sum and product of 2 integers is even or odd.
- Simulate flipping a coin and determine if the outcome is heads or tails.
- Given the age of a customer, determine the price of their movie ticket.
- Given the weight and dimensions of a package, determine if it is suitable for shipping.
- Given the 3 sides of a triangle, determine if it is equilateral, isosceles, or scalene, if it is a right triangle, or if the sides don't form a triangle at all.

Unit 8: For Loops, counting and summing
Students will learn that when they have code that needs to be repeated a given number of times, they need to use a count-controlled loop such as a for loop. Students will learn the common syntax of a for loop as well as how to increment and decrement a counter variable by one or any other necessary value. Ways of using for loops and if statements together will be discussed. Students will learn the common algorithms of counting (determining the number of times a particular event occurs) and summing (finding the total of a particular type of value). Nesting for loops will also be covered, including what happens during execution and when it is appropriate to use them.

Programming assignments for this unit will include:

- Print a given number of address labels.
- Flip a coin 500 times and determine the number of heads and tails that occurred.
- Enter a given number of test scores and calculate the total and average of those scores.
- Given the numerator and denominator, reduce a fraction.
- Output the first 50 numbers in the Fibonacci sequence.

Unit 9: While and Do-While Loops
Students will learn to use an event-controlled loop when they don't know how many times the loop should be executed. The while loop (a pre-test loop) and the do-while loop (a post-test loop) will be covered and students will learn when it is appropriate to use each one. Students will learn how to structure each loop with an appropriate sentinel value when getting user input. Ways of using while and do-while loops together with if statements will be discussed, including validating user input. Students will learn how these loops can be used with division and modulus to access the individual digits of an integer.

Programming assignments for this unit will include:

- Enter in many letter grades until a "Q" is entered to quit, then determine the total number of passes and fails.
- Use a loop to determine the value of a Certificate of Deposit after a given number of years.
- Make a guessing game where the user thinks of a number, then inputs if the number is too high, too low, or correct. Calculate and output the number of guesses it took to be correct.
- Given an integer, output the reverse of that integer.
- Given an integer, output its prime factorization.

Unit 10: The String class
Students will be introduced to the 2 categories of data types (primitive and object) and how each works differently in memory. Students will learn that String is an object type and will learn how to use the methods length, equals, compareTo, charAt, substring, and indexOf, including the return type and parameter type of each (revisited concept from Unit 6). The Character methods isUpperCase, isLowerCase, toUpperCase, and toLowerCase will also be introduced. Students will use these methods along with if statements and loops to check String equality, determine alphabetical order, visit each character in a String, and insert, remove, or manipulate parts of a String.

Programming assignments for this unit will include:

- Given 3 words, output them in alphabetical order.
- Determine the number of "e"s in a given sentence.
- Given a sentence, convert all the uppercase letters to lowercase letters, and all the lowercase letters to uppercase letters.
- Given a word, determine if it is a palindrome.

Unit 11: Create your own functions/methods
Students will learn how to create their own static methods, how they can be called within a program or in another program, and why user-defined methods are an important part of a well-designed program. Students will learn the difference between void and value-returning methods and when it is appropriate to use each one. Students will be able to determine the appropriate return type and parameters for the methods they create. Students will understand what it means for a method to be "pass by value". Local variables will be created and students will learn the concept of variable scope. Students will also learn how multiple methods can be used together to complete a large task.

Programming assignments for this unit will include:

- Write and test a method that determines the largest of 3 given integers.
- Write and test a method that counts the number of vowels in a given word.
- Write and test a method that inserts a space between each letter in a word.
- Write and test a method that determines if a number is prime.
- Write and test a method that reduces the square root of a number.

Unit 12: Introduction to arrays (optional, time permitting)
Students will learn how a single array can be used to store many values, instead of creating many variables. They will also learn how an array is created and accessed in memory. Students will be able to understand and use related vocabulary such as the terms element and index. Students will be able to perform various array algorithms including summing all the elements, finding the minimum and maximum elements, searching for a particular element, accessing the elements in reverse order, and comparing elements in parallel arrays.

Programming assignments for this unit will include:

- Given an array of integers, output all the even elements, all the odd elements, and all the negative elements.
- Given an array of integers, determine the position of a particular value.
- Given an array of words, determine the number of words that contain the letter "c".
- Given 2 arrays, one that stores the answers to a quiz and one that is a student's answer sheet, "score" that student's quiz.

Unit 13: Introduction to Object-Oriented Programming
Students are introduced to the idea of object oriented programming and how it is different from procedural programming (what they have been doing so far). Students will learn what an object class represents, the role of its 3 components (instance variables, constructors, and methods), and how that relates to encapsulation and information hiding. Students will learn how an object class and a tester class work together to create a well-designed and reusable program. Students will be able to identify when constructors and methods are being called based on the keyword "new" and dot notation. This unit will lead directly into the first unit in AP Computer Science A.

Programming assignments for this unit will include:

- Create a class that represents a Dice object, then create tester programs that do the following:
  - Roll a pair of Dice and output their sum.
  - Roll 3 Dice and determine the number of rolls were needed until all 3 had the same value.
  - Create 2 12-sided Dice and determine the number of rolls it took to get a particular sum.
  - Create a 2-sided Dice (to simulate flipping a coin) and roll it 10,000 times. Determine the length of the longest sequence of heads that was generated.
- Create a class that represents a Point object, then create tester programs that do the following:
  - Given 2 Point objects, find the distance between them.
  - Given 2 Point objects, find the slope between them.
  - Given 3 Point objects, determine which is the closest to the origin.

Course Materials

Textbook:
Teacher-created book "Java Jive" that has a chapter for each unit. This book is updated periodically with new information and assignments.

Websites:
codecademy.com
codingbat.com
codehs.com