September 6, 2017

Syllabus, class admin. details:

See class website, syllabus.

Introduction to Course Material:

- Demonstration/review of types

  X = 00110000

  What's the value of Y = X + 1 ? (+ x 1)

  ANSWER: depends on the type of X and how "+" operates on that type

  i.e. what do the bits of X represent?

  If X is an 8 bit unsigned integer: X = 48 (decimal), Y= 49 (decimal)

  If X is a character encoded in ASCII: X = 0, Y=01 (string "+" operation interpreted as append)

  (many other right answers here, too)

  - A **type** tells us how to interpret the bits stored in memory, defines what operators are possible, and how those operators affect those bits mean

- Most languages have several built-in types, e.g. Python: numerics (e.g. int, float), strings; C: int, float, char

- Object-oriented programming languages (e.g. Java) are built around the idea of defining your own types
  - Example "FaceChat" account type (see ppt slides)

- Concepts:
  - Our new type is called a "class" – class structure defines

    (a) what data are associated with this type (i.e. the bits)

(b) the possible operations on this type

(c) how those operations affect the data

- Instances of this type are called “objects”. They share the same structure and possible operations, but the value of their data variables can be different
- Operations are called methods— allow well-defined interaction with objects of this class
- Constructor—special “operation” that actually “builds” an object from the class template

- Key themes throughout the course:
  - <u>object-oriented design/programming</u>
    - Model programs after structure in the world around us: a set of objects that interact with each other to accomplish a task.
    - Define types that describe these objects and clear interfaces for their interaction
  - <u>abstract data types</u>
    - our lives are held together organization (“To do” lists, lines at the store check-out, dictionaries, maps)
    - Much of computer science centers on how we manage, organize, search, learn from large collections of data (e.g. social networking example—managing user profiles, data, network of friends)
    - abstract data type (ADT)— specification that describes a collection of data and operations on that collection (abstract—implementation independent—only what operations are possible, not how they’re carried out, “black box”) --- examples:
      - List: A collection that numbers its items
      - Stack: Orders items chronologically, Last In, First out
      - Queue: Orders items chronologically, First in, First out
      - Others: dictionary, tree, graph
    - data structure—an implementation of an ADT

        - important point: which ADT and which implementation is driven by the application--- you will learn to make these connections in this course

- algorithm design and analysis

    - What is an algorithm?: precise set of instructions for completing a task

    - Considerations during algorithm design:

        - Is it correct?

        - How efficient is it? How do we measure efficiency? (time, space complexity, how do they scale with data set size?)

    - Implementation of data structures (e.g. a sorted list) depends on proper and efficient algorithms-- we will learn about some common sorting/search algorithms, how to analyze efficiency

Next time:

More introduction to object-oriented design: defining classes, methods, object-oriented design

Basics of Java programming

Java programming environment (IntelliJ)