Chapter 15 - Recursion

Spring 2022

Objectives (1 of 2)

In this chapter, you will:

- Learn about recursive definitions
- Explore the base case and the general case of a recursive definition
- Discover what a recursive algorithm is
- Learn about recursive functions

Objectives (2 of 2)

- Become familiar with direct and indirect recursion
- Explore how to use recursive functions to implement recursive algorithms
- Become aware of recursion vs. iteration

Recursive Definitions (1 of 5)

- **Recursion**: solving a problem by reducing it to smaller versions of itself
  - Provides a powerful way to solve certain problems which would be complicated otherwise

Recursive Definitions (2 of 5)

- **Recursive definition**: a definition in which something is defined in terms of a smaller version of itself
- **Base case**: the case for which the solution is obtained directly
  - Every recursive definition must have one (or more) base case(s)
  - The **general case** must eventually reduce to a base case
  - The **base case** stops the recursion

Recursive Definitions (3 of 5)

- Example: factorial of a nonnegative integer

```
0! = 1  (15-1)
n! = n x (n– 1)!   if   n > 0   (15-2)
```

- Equation 15-1 is called the base case
- Equation 15-2 is called the general case

Recursive Definitions (4 of 5)

- A **recursive algorithm** finds a solution to a given problem by reducing the problem to smaller versions of itself
  - Must have one (or more) base cases
  - General solution must eventually reduce to a base case
- A **recursive function** is a function that calls itself
- Recursive algorithms are implemented using recursive functions

Recursive Definitions (5 of 5)

- Think of a recursive function as having infinitely many copies of itself
  - Every call has its own code and its own set of parameters and local variables
  - After completing a particular recursive call:
    - Control goes back to the calling environment, the previous call
    - Execution begins from the point immediately following the recursive call

Direct and Indirect Recursion

- **Directly recursive**: a function that calls itself
- **Indirectly recursive**: a function that calls another function and eventually results in the original function call
- **Tail recursive function**: a recursive function in which the last statement executed is the recursive call

Infinite Recursion (1 of 2)

- **Infinite recursion**: every recursive call results in another recursive call
  - In theory, infinite recursion executes forever
- Because computer memory is finite:
  - Function executes until the system runs out of memory
  - Results in an abnormal program termination

Infinite Recursion (2 of 2)

- To design a recursive function:
  - Understand the problem requirements
  - Determine limiting conditions
  - Identify the base cases and provide a direct solution to each base case
  - Identify the general cases and provide a solution to each general case in terms of smaller versions of itself

Recursion or Iteration? (1 of 4)

- An **iterative control structure** uses a loop to repeat a set of statements
- There are usually two ways to solve a particular problem
  - Iteration (looping)
  - Recursion

- When choosing the method of solving, we must consider:
  - The nature of the problem
  - Efficiency

Recursion or Iteration? (2 of 4)

- Whenever a function is called:
  - Memory space for its formal parameters and (automatic) local variables is allocated
- When the function terminates:
  - That memory space is then deallocated
- Every (recursive) call has its own set of parameters and (automatic) local variables

Recursion or Iteration? (3 of 4)

- Overhead associated with executing a (recursive) function in terms of:
  - Memory space
  - Computer time
- A recursive function executes more slowly than its iterative counterpart
- Today's computers are fast
  - Overhead of a recursion function is not noticeable

Recursion or Iteration? (4 of 4)

- Sometimes an iterative solution is more obvious and easier to understand
- If the definition of a problem is inherently recursive, consider a recursive solution

Quick Review (1 of 3)

- Recursion: process of solving a problem by reducing it to smaller versions of itself
- Recursive definition: defines a problem in terms of smaller versions of itself
- Has one or more base cases
- Recursive algorithm: solves a problem by reducing it to smaller versions of itself
- Has one or more base cases

Quick Review (2 of 3)

- The solution to the problem in a base case is obtained directly
- A recursive function is a function that calls itself
- Must have one or more base cases
- Recursive algorithms are implemented using recursive functions
- The general solution breaks the problem into smaller versions of itself

Quick Review (3 of 3)

- The general case must eventually be reduced to a base case
- The base case stops the recursion
- A function is called directly recursive if it calls itself

- A function that calls another function and eventually results in the original function call is said to be indirectly recursive
- A recursive function in which the last statement executed is the recursive call is called a tail recursive function

Questions?