Recursion

Lecture 28
Sections 10.1 - 10.2

Robb T. Koether

Hampden-Sydney College

Fri, Mar 27, 2009
Outline

1. Recursion
2. Advantages and Disadvantages
3. Higher-Order Recursion
4. Assignment
A recursive function is a function that calls on itself.
A recursive function must contain a decision structure that divides the code into two cases.
- The recursive (general) case.
- The nonrecursive (base) case.
The code must guarantee that eventually the nonrecursive case is called.
Advantages of Recursion

- The code may be much easier to write.
- Some situations are naturally recursive.
Naturally recursive data structures:
- Linked lists.
- Binary trees.

Naturally recursive problems:
- Traversing linked lists.
- Traversing binary trees.
- Evaluating expressions.
- Differentiating functions.
- The Triangle puzzle.
Recursive Problems

Example (The Triangle Puzzle)

```cpp
bool solveTrianglePuzzle()
{
    if (one peg remains)
        return true;
    while (there is an untried move)
    {
        Make the next move;
        if (solveTrianglePuzzle())
            return true;
    }
    return false;
}
```
Disadvantages of Recursion

- Recursive functions are generally slower than nonrecursive functions.
- Excessive recursion may overflow the run-time stack.
- One must be very careful when writing recursive functions; they can be tricky.
Disadvantages of Recursion

- There is **shallow recursion** and there is **deep recursion**.
- Shallow recursion will not overflow the stack, but it may take an excessively long time to execute.
- Deep recursion is generally much faster, but it may overflow the stack.
Higher-Order Recursion

- Sometimes each function call generates two or more recursive calls at that level.
- This has the potential to consume an enormous amount of time.
Example (1st-Order Recursion)

```c
int factorial(int n)
{
    if (n == 0 || n == 1)
        return 1;
    else
        return n*factorial(n - 1);
}
```
Recursive Problems

Example (2nd-Order Recursion)

```c
int fibon(int n)
{
    if (n == 0 || n == 1)
        return 1;
    else
        return fibon(n - 1) + fibon(n - 2);
}
```
Recursive Problems

Example (Greatest Common Divisor)

```c
int gcd(int a, int b)
{
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}
```
Recursive Problems

Example (Binomial Coefficients)

```c
int binom(int n, int r)
{
    if (r == 0 || r == n)
        return 1;
    else
        return binom(n - 1, r) + binom(n - 1, r - 1);
}
```
Examples of Recursive Functions

- Download and run `TowersOfHanoi.cpp`
Assignment

Homework

- Read Section 10.1, pages 521 - 533.
- Read Section 10.2, pages 538 - 547.