An Introduction to Programming though C++

Abhiram G. Ranade

Lecture 5.2

Ch. 10: Recursive functions

How to think about recursion

- A recursive program seems very complex.
- As it executes, multiple calls might be in progress at the same time.
- How do we reason about all these?
- Isnt it very hard just to visualize the execution?

Key point: We do not attempt to visualize all this.

High level ideas

Designing a recursive algorithm:

- Attempt to solve the given problem instance by constructing and solving smaller instances of the same type.
  - GCD: (m,n) □ GCD(n,m%n), Drawing: (L,...) □ (L-1,...)
- Solve the simplest instances directly.
  - GCD: m%n = 0, Drawing: L=0

Understanding recursive algorithms:

- Is there a “problem size” that reduces in the recursive calls?
  - GCD: Argument 2. Drawing: Argument 1.
- Will we eventually get to problems that are solved without recursion?

Some terminology

Top level recursive call:
The call made from the main program, or the first call made to the recursive function.

- gcd: the call **gcd(m,n)** Drawing: the call **tree(L,rx,ry,H,W)**

Level 1 recursive calls
Calls made directly while executing the top level call.

- gcd: **gcd(n,m%n)**
- Drawing:

tree(L-1, rx-W/4, ry-H/L, H-H/L, W/2)
tree(L-1, rx+W/4, ry-H/L, H-H/L, W/2)

Base cases:
Input values for which the top level call returns without recursing.

- gcd: m,n such that m%n == 0 Drawing: L = 0

More Terminlogy

Preconditions: Valid values for inputs

GCD: m,n > 0.

Drawing: ***L≥0, rx,ry, H≥0, W≥0***

Problem Size: Something indicative of the amount of work needed to find the solution.

Needs to be chosen creatively

GCD: n

Drawing: L

Understanding recursive functions 1: Base cases

Are there any base cases?

- Base cases must exist, otherwise program will not terminate.

Does the function produce correct results for the base cases?

- gcd: base cases: m,n such that m%n == 0.
  - Answer in such cases: n, which is correct.
- Drawing: base case: L = 0
  - Answer in this case: Nothing drawn.
  - Correct because tree is empty for L=0.

Understanding recursive functions 2: Valid level 1 recursive calls

Are the arguments in the level 1 calls valid, i.e. do they satisfy the preconditions of the functions?

- gcd: level 1 call is gcd(n, m%n)
  - We require that arguments must be positive integers.
  - n > 0 because n is an argument to the top level call gcd(m,n), and we assume that it satisfies preconditions
  - Level 1 call is made only if m%n > 0. So second argument is also positive.
- Drawing: Level 1 calls are:
  - ***tree(L-1, rx-W/4, ry-H/L, H-H/L, W/2)***
  - ***tree(L-1, rx+W/4, ry-H/L, H-H/L, W/2)***
  - Level 1 calls are made only if L>0, so L-1 ≥ 0.
  - We can also see that H – H/L, W/2 are ≥ 0.

Understanding recursive functions 3: Does the problem size reduce? Can it reduce indefinitely?

- gcd: level 1 call is gcd(n, m%n)
  - Second argument reduces. It must stay above 0.
- Drawing: Level 1 calls are:
  - ***tree(L-1, rx-W/4, ry-H/L, H-H/L, W/2)***
  - ***tree(L-1, rx+W/4, ry-H/L, H-H/L, W/2)***
  - First argument reduces. It cannot become negative.

Underst. Rec Functions. 4: Will the top level calls return the correct result if the level 1 calls do?

GCD:

- Assume level 1 call, GCD(n,m%n) returns correct result.
- Top level call returns what level 1 call returns.
- Code examination: GCD of m,n = GCD of n, m%n. So top level is correct.

Drawing:

- Suppose Level 1 calls (L-1,...), (L-1,...) draw the subtrees correctly
- Then top level call draws the branches in the right positions.
- Overall drawing will be correct

Summary

To check if a recursive function is correct we should check

1. There are base cases and correct results are obtained for the base cases.
2. The level 1 recursive calls satisfy the preconditions.
3. The problem size reduces but cannot reduce indefinitely.
4. If the level 1 calls work correctly, the top level call will work correctly.

- We do not need to argue that the level 1 calls work correctly.
- We dont even need to think about calls made by level 1 calls.
- 1,2,3 ensure that the computation will terminate eventually.
- 4 ensures that the correct result will be returned.

Exercise

Consider the function below and its specification.

```
int f(int n){
  if(n == 0) return 1;
  return f(n-1) + f(n-2);
}
```

Precondition: n should be a non-negative integer

Postcondition: f(n) should equal f(n-1) + f(n-2), with f(0) = 0

State whether this function is correct.

This is a trick question: you should also consider whether the post condition is specified fully. You should be able to do this by asking the questions discussed earlier.

Concluding remarks

- Recursion allows many programs to be expressed very compactly.
- The idea that the solution of a large problem can be obtained from the solution of a similar problem of the same type, is very powerful.
  - Euclid probably used this idea to discover his GCD algorithm.
  - Recursion is very natural for objects having recursive definition, e.g. trees
- To understand if/why a recursive function works, we need to make a few simple checks.
- Some programs can be written recursively or iteratively (gcd, factorial), but some others are best written recursively (drawing trees).

🎻🎻