

Recursion

Lecture 2

Sections 14.1 - 14.5, 14.7

Robb T. Koether

Hampden-Sydney College

Fri, Jan 20, 2017

- 1 Recursion
- 2 Advantages and Disadvantages
- 3 Examples
- 4 Assignment

Outline

- 1 Recursion
- 2 Advantages and Disadvantages
- 3 Examples
- 4 Assignment

Recursion

Definition (Recursive Function)

A **recursive function** is a function that calls on itself.

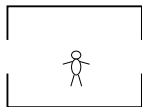
- A recursive function must contain a decision structure that divides the code into two cases.
 - The **recursive (general) case**.
 - The **nonrecursive (base) case**.
- The code must guarantee that eventually the nonrecursive case is called.

1st-Order Recursion

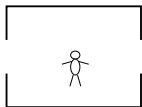
Example (The Factorial Function)

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

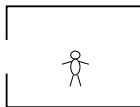
An Analogy



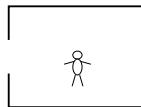
Level 1



Level 2



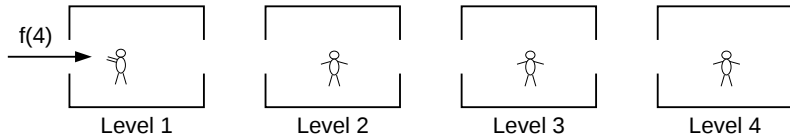
Level 3



Level 4

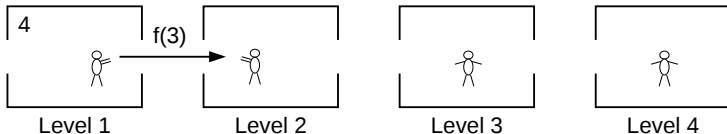
caption

An Analogy



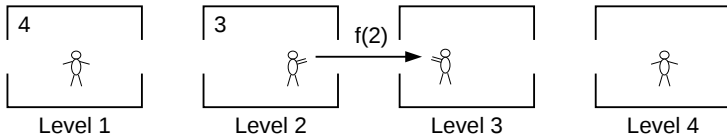
caption

An Analogy



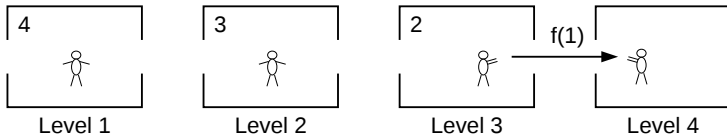
caption

An Analogy



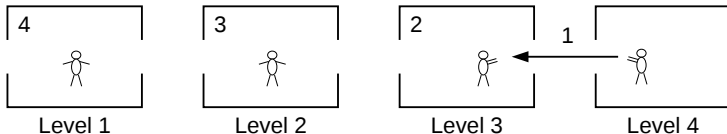
caption

An Analogy



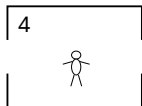
caption

An Analogy

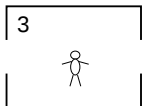


caption

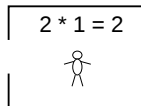
An Analogy



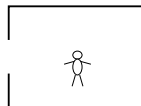
Level 1



Level 2



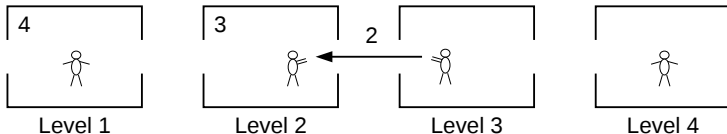
Level 3



Level 4

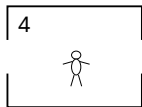
caption

An Analogy

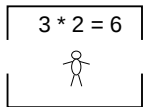


caption

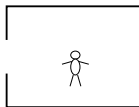
An Analogy



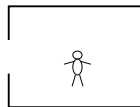
Level 1



Level 2



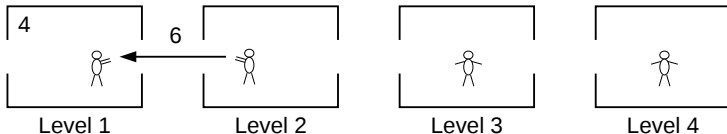
Level 3



Level 4

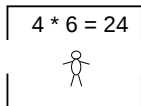
caption

An Analogy

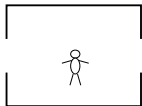


caption

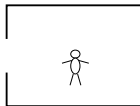
An Analogy



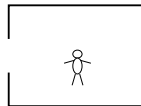
Level 1



Level 2



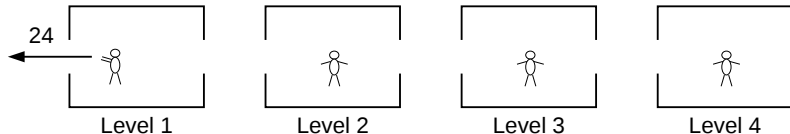
Level 3



Level 4

caption

An Analogy



caption

2nd-Order Recursion

Example (The Fibonacci Sequence)

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

Greatest Common Divisor

Example (Greatest Common Divisor)

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

Binomial Coefficients

Example (Binomial Coefficients)

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

Outline

- 1 Recursion
- 2 Advantages and Disadvantages**
- 3 Examples
- 4 Assignment

Advantages of Recursion

- Advantages

- The code may be much easier to write.
- Some situations are naturally recursive.

Advantages of Recursion

- Naturally recursive data structures:
 - Linked lists.
 - Binary trees.
- Naturally recursive problems:
 - Traversing linked lists.
 - Traversing binary trees.
 - Evaluating expressions.
 - Differentiating functions.

Disadvantages of Recursion

- Disadvantages
 - 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.
- Sometimes each function call generates two or more recursive calls at that level.
- This has the potential to consume an enormous amount of time.

Outline

- 1 Recursion
- 2 Advantages and Disadvantages
- 3 Examples**
- 4 Assignment

Examples of Recursive Functions

- `GCD.cpp`.
- `BinomialCoefficients.cpp`.
- `TowersOfHanoi.cpp`.

Outline

- 1 Recursion
- 2 Advantages and Disadvantages
- 3 Examples
- 4 Assignment**

Assignment

Assignment

- Read Sections 14.1 - 14.5, 14.7.