

CSE 142 Computer Programming I

Recursion

© 2000 UW CSE

W-1

Overview

Review

Function calls in C

Concepts

Recursive definitions and functions

Base and recursive cases

Reading

Read textbook sec. 10.1-10.3 & 10.7

Optional: sec. 10.6 (Towers of Hanoi, a classic example)

Skip sec. 10.4-10.5

W-2

Factorial Function

Factorial is an example of a mathematical function that is defined *recursively*, i.e., it is partly defined in terms of itself.

$$n! = \begin{cases} 1 & n \leq 1 \\ n * (n-1)! & \text{otherwise} \end{cases}$$

Factorial Revisited

We've already seen an implementation of factorial using a loop

```
int factorial ( int n ) {  
    int product, i;  
    product = 1;  
    for ( i = n; i > 1; i = i - 1 ) {  
        product = product * i;  
    }  
    return product;  
}
```

1! is 1
2! is 1 * 2
3! is 1 * 2 * 3
4! is 1 * 2 * 3 * 4^{W-4}
5! is 1 * 2 * 3 * 4 * 5
..

Factorial, Recursively

But we can use the recursive definition directly to get a different version

/ Compute n factorial – the product of the first n integers, 1 * 2 * 3 * 4 ... * n */*

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

W-5

Trace

factorial(4) =

4 * factorial(3) =

4 * 3 * factorial(2) =

4 * 3 * 2 * factorial(1) =

4 * 3 * 2 * 1 =

4 * 3 * 2 =

4 * 6 = **24**

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

What is Recursion?

Definition: A function is **recursive** if it calls itself

```
int foo(int x) {  
    ...  
    y = foo(...);  
    ...  
}
```

How can this possibly work???

W-7

Function Calls

Answer: there's nothing new here!

Remember the steps for executing a function call in C:

- Allocate space for called function's parameters and local variables
- Initialize parameters
- Begin function execution

Recursive function calls work exactly the same way
New set of parameters and local variables for each (recursive) call

Trace

main k 24

```
int factorial(int n){  
    int result;  
    if (n <= 1)  
        result = 1;  
    else  
        result = n *  
            factorial(n - 1);  
    return result;  
}  
  
int main(void) {  
    ...  
    k = factorial(4);  
    ...  
}
```

W-9

Recursive & Base Cases

A recursive definition has two parts

One or more **recursive cases** where the function calls itself

One or more **base cases** that return a result without a recursive call

There **must** be at least one base case

Every recursive case **must** make progress towards a base case

Forgetting one of these rules is a frequent cause of errors with recursion

W-10

Recursive & Base Cases

Base case

Recursive case

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

Does This Run Forever?

Check:

Includes a base case?

Yes

Recursive calls make progress? Hmm...

Answer: Not known!!!

In tests, it always gets to the base case eventually, but nobody has been able to **prove** that this must be so!

```
int f (int x) {  
    if (x == 1)  
        return 1;  
    else if (x % 2 == 0)  
        return 1 + f(x/2);  
    else  
        return 1 + f(3*x + 1);  
}
```

W-12

3N + 1 function

$$\begin{aligned} f(5) &= 1 + f(16) = 2 + f(8) = 3 + f(4) \\ &= 4 + f(2) = 5 + f(1) = 6 \end{aligned}$$

$$\begin{aligned} f(7) &= 1 + f(22) = 2 + f(11) = 3 + f(34) \\ &= 4 + f(17) = 5 + f(52) = 6 + f(26) \\ &= 7 + f(13) = 8 + f(40) = 9 + f(20) \\ &= 10 + f(10) = 11 + f(5) = 12 + f(16) \\ &= 13 + f(8) = 14 + f(4) = 15 + f(2) \\ &= 16 + f(1) = 17 \end{aligned}$$

W-13

Summary

Introduced Recursion

Functions that call themselves

Base and recursive cases

Examples

Factorial

3N+1 function

An important concept with many uses

W-14