An Introduction to Programming
through C++

with

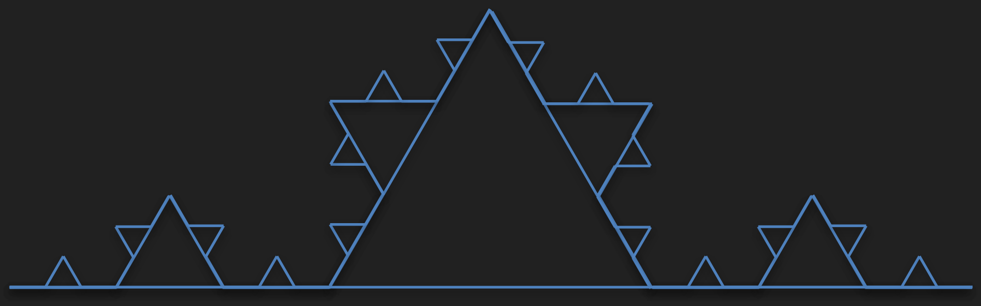
Manoj Prabhakaran

Lecture 15

Recursion

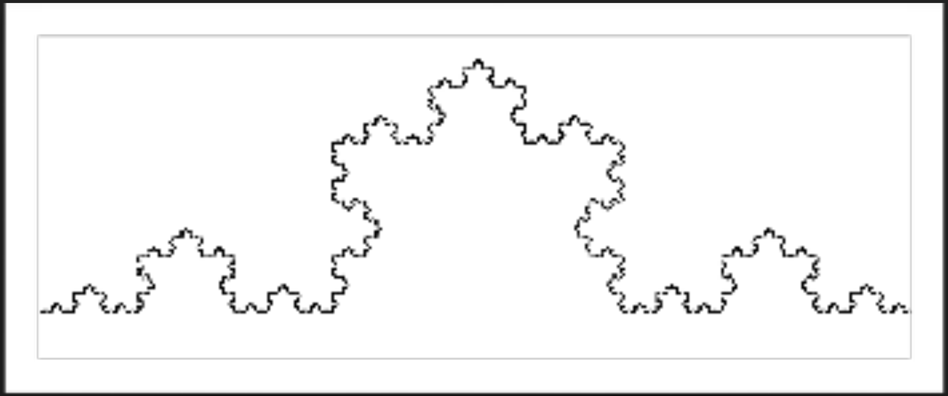
Based on material developed by Prof. Abhiram G. Ranade

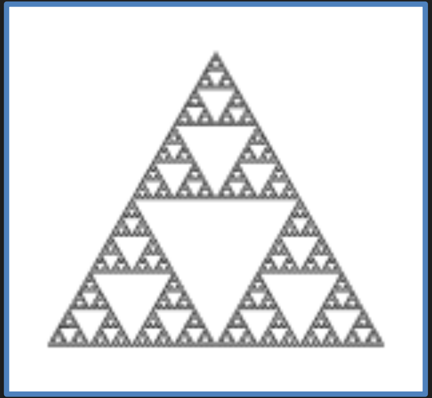
Example: Koch Curve

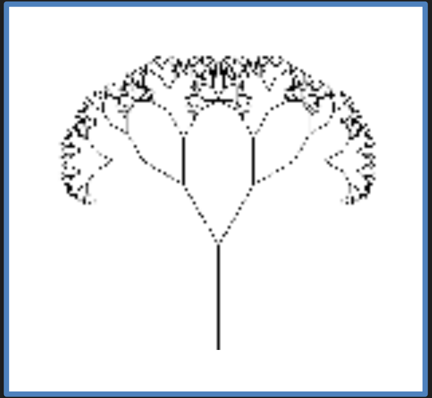


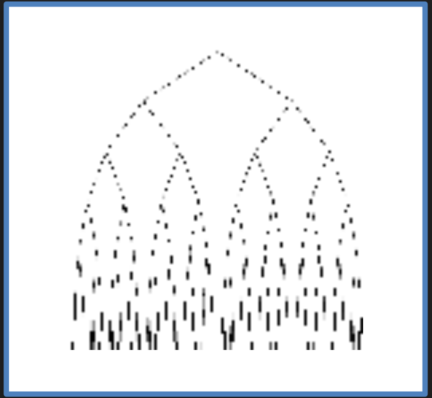
More Fractals

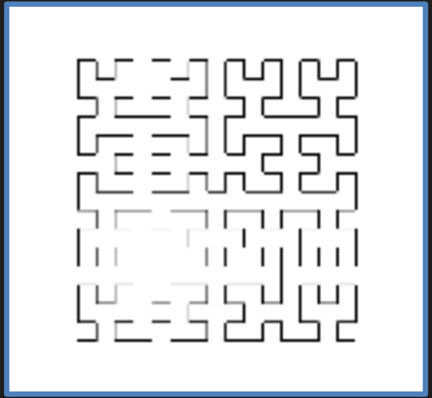
Demo











Example: Permutations

- Given an array of n objects (say characters), print all n! permutations
- E.g.,

```
char A[3] = {'a', 'b', 'c'};
permute(A,3);
```

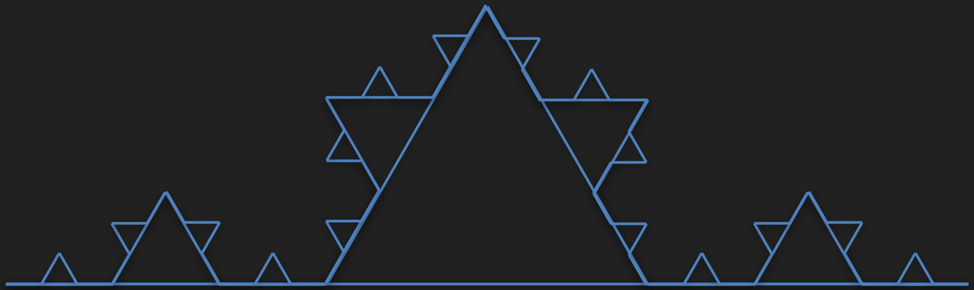
→

```
a b c
a c b
b a c
b c a
c b a
c a b
```

Example: GCD and Bézout's Identity

- Recall Greatest Common Divisor (a.k.a. Highest Common Factor)
  - Divisors(12) = { 1, 2, 3, 4, 6, 12}. Divisors(20) = { 1, 2, 4, 5, 10, 20}
    CommonDivisors(12,20) = { 1, 2, 4}. gcd(12,20) = 4.
  - Divisors(0) = {0, 1, 2, ... }. gcd(a,0) = a, for all a.
    (interpreting p greater than q to mean q divides p, gcd(0,0) = 0.)
- Bézout's Identity: For any two integers a, b, there exist integers m, n such that
  gcd(a,b) = $ma + nb$
- Examples:
  gcd(12, 20) = 4 = 2×12 + (-1)×20
  gcd(10, 7) = 1 = (-2)×10 + 3×7
  gcd(100, 49) = 1 = (-24)×100 + 49×49
- Problem: Write a program to compute such a pair (m,n) given (a,b)

Example: Koch Curve



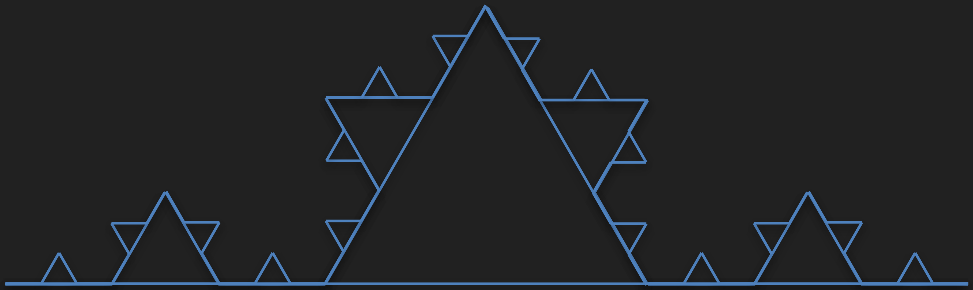
Recursion:
A function calling itself.

```
void draw(double L, int level) {
  if (level == 0) { forward(L); return; }
  if (level == 1) {
    draw(L/3,0);  left(60);
    draw(L/3,0);  right(120);
    draw(L/3,0);  left(60);
    draw(L/3,0);
  }
```

```
  if (level == 2) {
    draw(L/3,1); left(60);
    draw(L/3,1); right(120);
    draw(L/3,1); left(60);
    draw(L/3,1);
  }
  if (level == 3) ...
}
```

Example: Koch Curve

Demo



Base case:
Ensures that the recursion is not infinite

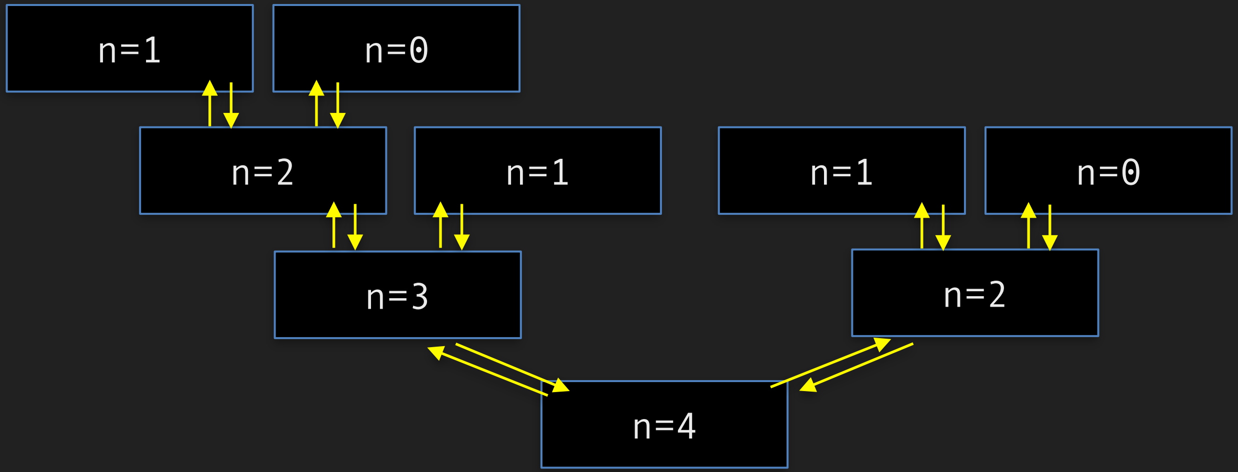
```
void draw(double L, int level) {
  if (level == 0) { forward(L); return; }
  draw(L/3, level-1); left(60);
  draw(L/3, level-1); right(120);
  draw(L/3, level-1); left(60);
  draw(L/3, level-1);
}
```

Example: Fibonacci Sequence

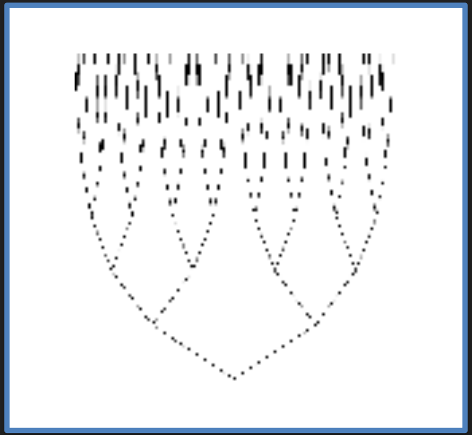
A very inefficient implementation!

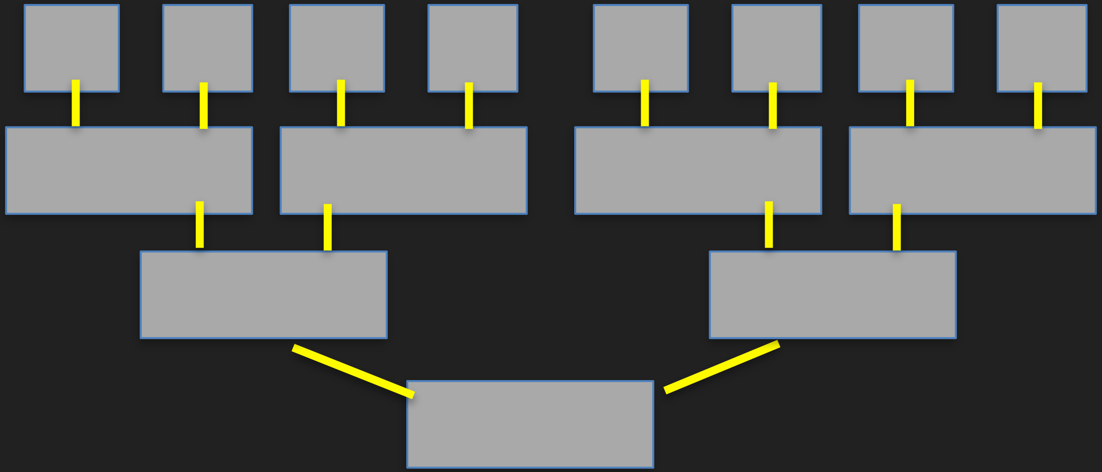
```
int Fibonacci(unsigned int n) {
  if(n==0) return 0;
  if(n==1) return 1;
  return Fibonacci(n-1) + Fibonacci(n-2);
}
```

Base cases

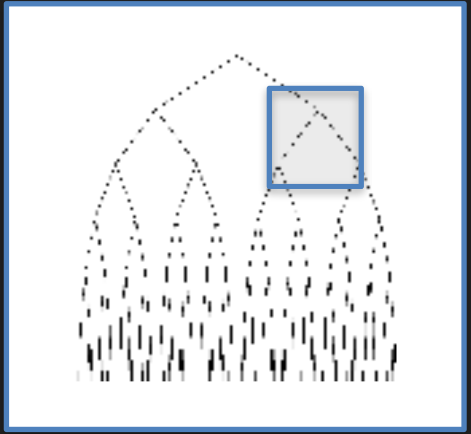


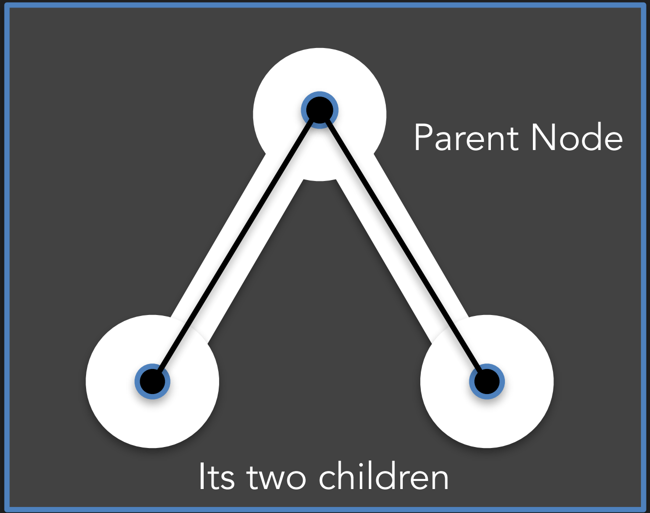
Example: A Binary Tree





Example: A Binary Tree





- Recursive structure: A (complete) binary tree of height h is:
  - Base case: If h is 0, a single point (root node)
  - Else, a root node and two (complete) binary trees of height h-1, each connected to the root node by a line (edge)

Example: A Binary Tree

Demo

```
binaryTree(double boxHeight, double boxWidth, int levels) {
  if (levels == 0) return;
  // Not shown: calculate angle, edgeLen, childBoxHeight, childBoxWidth
  left(angle); forward(edgeLen); left(-angle);
  binaryTree(childBoxHeight, childBoxWidth, levels-1);
  left(angle); forward(-edgeLen); left(-angle);
  left(-angle); forward(edgeLen); left(angle);
  binaryTree(childBoxHeight, childBoxWidth, levels-1);
  left(-angle); forward(-edgeLen); left(angle);
}
```

Right child

Left child

Important: Turtle is back at the original position

- Recursive structure: A (complete) binary tree of height h is:
  - Base case: If h is 0, a single point (root node)
  - Else, a root node and two (complete) binary trees of height h-1, each connected to the root node by a line (edge)

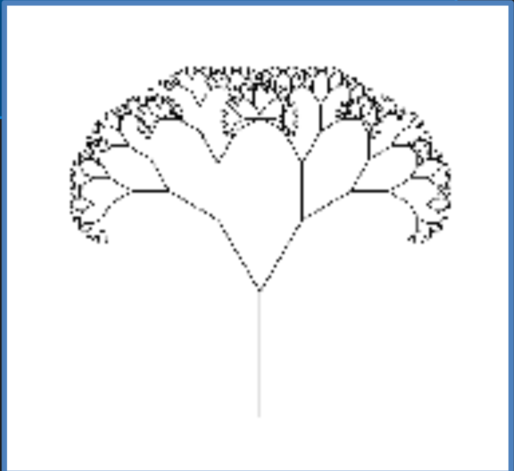
Example: Canopy

Demo

```
void canopy(double step, int level, double angle, double ratio) {
    if (level == 0) return;
    forward(step);
    left(angle/2);
    canopy(step*ratio,level-1,angle,ratio);
    right(angle);
    canopy(step*ratio,level-1,angle,ratio);
    left(angle/2);
    forward(-step);
}
```

Left child

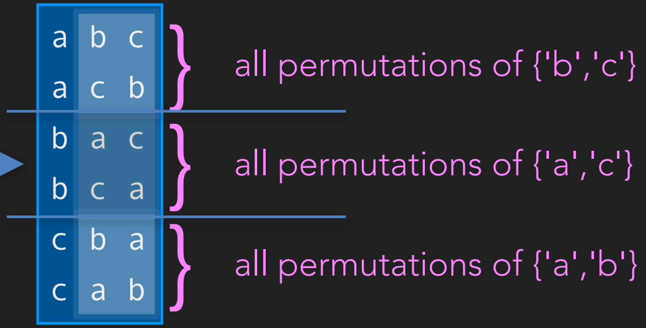
Right child



Example: Permutations

- Given an array of n objects (say characters), print all n! permutations
- E.g.,

```
char A[3] = {'a', 'b', 'c'};
permute(A,3);
```



- Recursive structure of permute(A,L)
  - The first character can be each of the L characters given
  - Once the first character is fixed, the rest is a permutation of the remaining characters
  - Base case: when L=1, nothing more to do

Example: Permutations

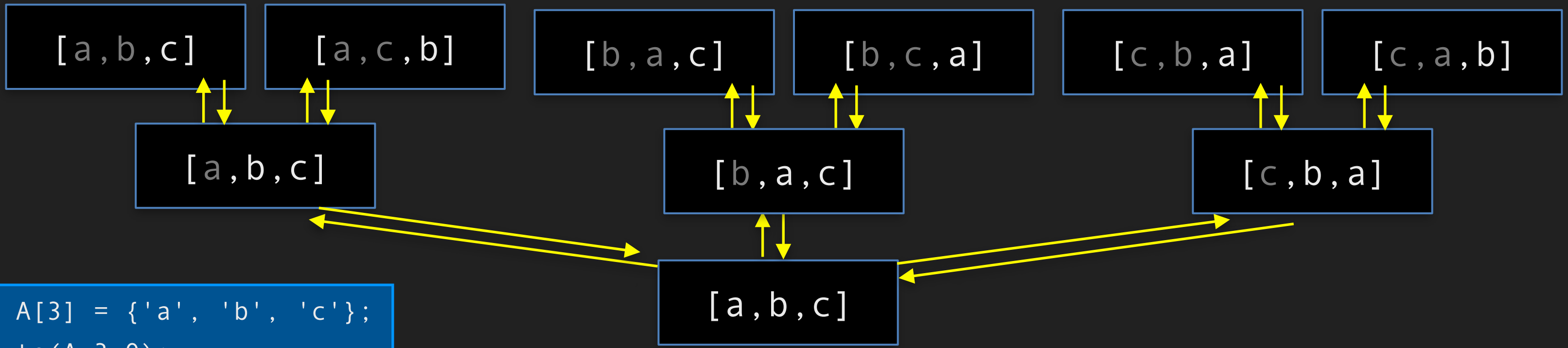
```
void permute(char A[], int n, int fixed=0) { // L=n-fixed elements to be permuted
    if(n-fixed == 1) {    // Base case: L==1
        printArray(A,n);  // print the whole array
        return;
    }
    for(int i=fixed; i < n; i++) {
        swap(A[i], A[fixed]);  // pick A[i] from among unfixed and make it first
        permute(A, n, fixed+1);// fix it too, permute rest; returned array unchanged
        swap(A[i], A[fixed]);  // to return array unchanged, undo each time we swap
    }
}
```

- Recursive structure of permute(A,L)
    - The first character can be each of the L characters given
    - Once the first character is fixed, the rest is a permutation of the remaining characters
    - Base case: when L=1, nothing more to do

Example: Permutations

Demo

```
void permute(char A[], int n, int fixed=0) { // L=n-fixed elements to be permuted
    if(n-fixed == 1) {     // Base case: L==1
        printArray(A,n);  // print the whole array
        return;
    }
    for(int i=fixed; i < n; i++) {
        swap(A[i], A[fixed]);  // pick A[i] from among unfixed and make it first
        permute(A, n, fixed+1);// fix it too, permute rest; returned array unchanged
        swap(A[i], A[fixed]);  // to return array unchanged, undo each time we swap
    }
}
```




```
a b c
a c b
b a c
b c a
c b a
c a b
```

```
char A[3] = {'a', 'b', 'c'};
permute(A,3,0);
```

Example: GCD and Bézout's Identity

- Bézout's Identity: For any two integers *a*, *b*, there exist integers *m*, *n* such that $\gcd(a,b) = ma + nb$
- Recursive structure of GCD:
  - Base case (b==0): `gcd(a,0) = |a|`
  - Else (b!=0): `gcd(a,b) = gcd(b,a%b)`
    - Because, for any *q*, $\gcd(a,b) = \gcd(b,a-qb)$ and `a%b = a - (a/b)*b`

Exercise (optional): x is a common divisor of (a,b) if and only if it is a common divisor of (a-qb,b).
i.e., set of common divisors of (a,b) is exactly the same as that of (a-qb,b).

The problem size decreases: If |a|>|b| (which holds after at most one recursive call) and b≠0, (|b|,|a%b|) < (|a|,|b|).

```
int gcd (int a, int b) {
    if(b==0)              // Base case: GCD(a,0)= |a|
        return abs(a);
    return gcd(b,a%b); // Recursing: GCD(a,b) = GCD(b,a-qb), where q=a/b
}
```

Example: GCD and Bézout's Identity

- Bézout's Identity: For any two integers a, b, there exist integers m, n such that $\gcd(a,b) = ma + nb$
- Recursive structure of GCD:
  - Base case (`b==0`): `gcd(a,0) = |a|`
  - Else (`b!=0`): `gcd(a,b) = gcd(b,a%b)`
    - Because, for any q, $\gcd(a,b) = \gcd(b,a-qb)$, and `a%b = a - (a/b)*b`
- How about Bézout's Identity?
  - Suppose `g = m1*b + n1*(a%b) = m1*b + n1*(a - (a/b)*b)`. Then `g = ma + nb`, where `m = n1` and `n = m1 - n1*(a/b)`

Example: GCD and Bézout's Identity

```
// returns GCD(a,b) and sets m,n so that ma+nb=GCD(a,b)
int gcd (int a, int b, int& m, int& n) {
    if(b==0) {                // Base case: GCD(a,0)= |a|; m = sign(a), n arbitrary
        n = 0; m = a<0? -1 : 1;
        return abs(a);
    }
    int g = gcd(b,a%b,n,m); // Recursing: GCD(a,b) = GCD(b,a-qb), where q=a/b
    n -= m*(a/b);             // g = m'b + n'(a-qb) = n'a+(m'-n'q)b => m=n', n=m'-n'q
    return g;
}
```

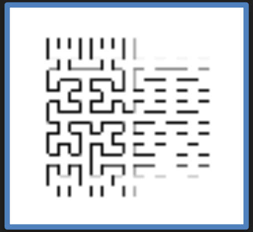
- How about Bézout's Identity?
  - Suppose g `=` `m1*b + n1*(a%b) = m1*b + n1*(a - (a/b)*b)`. Then g `= ma + nb`, where m `= n1` and n `= m1 - n1*(a/b)`

Examples Today

- Fractals







- Fibonacci sequence
- Permutations
- GCD and Bézout's Identity