

Recursion

You will learn what recursion is as well as how simple recursive programs work

James Tam

What Is Recursion?

“the determination of a succession of elements by operation on one or more preceding elements according to a rule or formula involving a finite number of steps” (Merriam-Webster online)

James Tam

What This Really Means

Breaking a problem down into a series of steps. The final step is reached when some basic condition is satisfied. The solution for each step is used to solve the previous step. The solution for all the steps together form the solution to the whole problem. (The “Tam” translation)

James Tam

Definition For Philosophy

“...state of mind of the wise man; practical wisdom...”¹

See Metaphysics

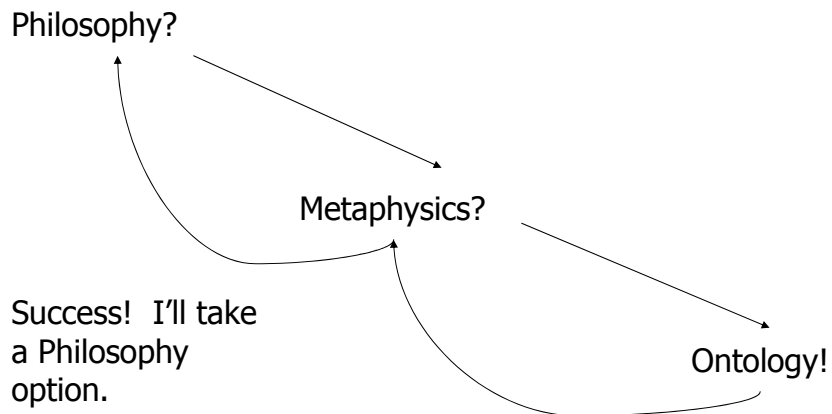
Metaphysics

“...know the ultimate grounds of being or what it is that really exists, embracing both psychology and *ontology*.”²

Result Of Lookup , Possibility One: Success

- I know what Ontology means!

Result Of Lookup, Possibility One



James Tam

Result Of Lookup, Possibility Two: Failure

- Lookups loop back.

James Tam

Result Of Lookup, Possibility Two

Philosophy?

Metaphysics?

Rats!!!

See
previous

Ontology?

James Tam

Ontology

“...equivalent to metaphysics.”³

³ The New Webster Encyclopedic Dictionary of the English Language

Wav file from "The Simpsons"

James Tam

Result Of Lookup, Possibility Three: Failure

- You've looked up everything and still don't know the definition!

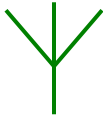
James Tam

Looking Up A Word

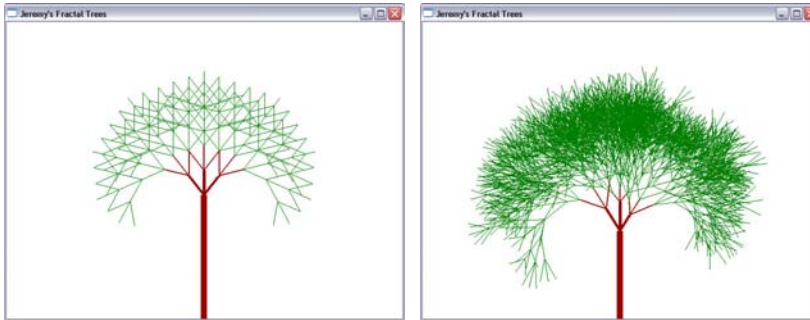
```
if (you completely understand a definition) then
    return to previous definition (using the definition that's
    understood)
else
    lookup (unknown word(s))
```

James Tam

Recursion: Can Be Used To Produce Graphics



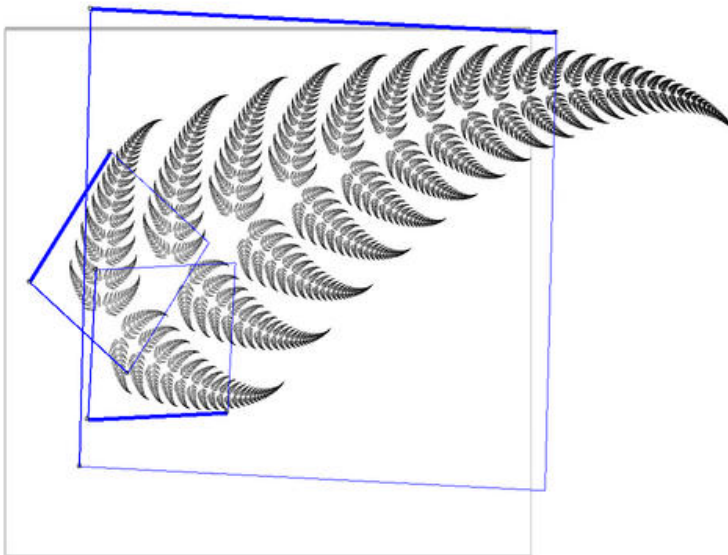
Produce a picture by repeating
a pattern



Images from <http://www.csis.gvsu.edu/~marzkaj/CS367/project1.htm>

James Tam

Recursion: Can Be Used To Produce Graphics (2)



<http://charm.cs.uiuc.edu/users/olawlor>

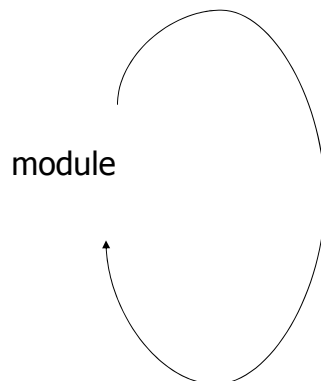
James Tam

Recursion In Programming

“A programming technique whereby a function or procedure calls itself either directly or indirectly.”

James Tam

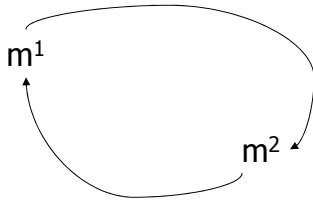
Direct Call



```
procedure proc;  
begin  
  :  
  proc ();  
  :  
end;
```

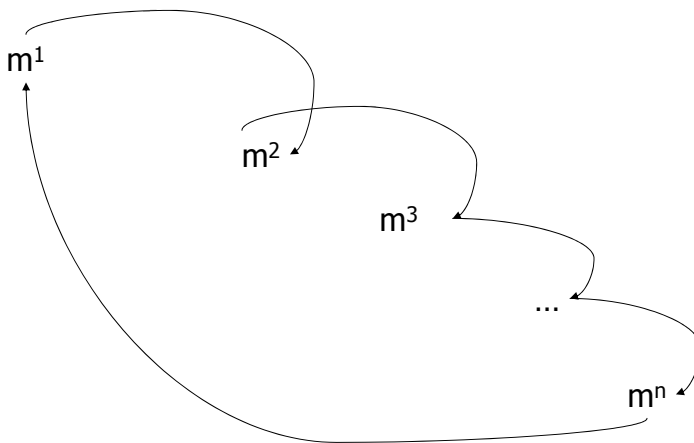
James Tam

Indirect Call



James Tam

Indirect Call



James Tam

Indirect Call (2)

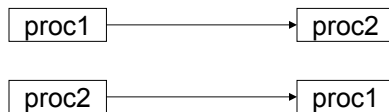
```
procedure proc1;  
begin  
  :  
  proc2;  
end;  
  
procedure proc2;  
begin  
  :  
  proc3;  
end;  
  
procedure proc3;  
begin  
  :  
  proc1;  
end;
```

James Tam

An Issue With Indirect Recursion

For a full example look under
</home/231/examples/recursion/indirect.p>

Example Scenario:



Which one should be defined first?

James Tam

Procedure Proc1 First?

```
procedure proc1;  
begin  
  :  
  proc2;  
  :  
end;
```

```
procedure proc2;  
begin  
  :  
  proc1;  
  :  
end;
```

What is proc2?

James Tam

Procedure Proc2 First?

```
procedure proc2;  
begin  
  :  
  proc1;  
  :  
end;
```

```
procedure proc1;  
begin  
  :  
  proc2;  
  :  
end;
```

What is proc1?

James Tam

Solution: Use A Dummy Definition

A "placeholder" for the compiler (definition comes later)

Example problem

```
procedure proc1;  
begin  
  :  
  proc2;  
  :  
end;
```

```
procedure proc2;  
begin  
  :  
  proc1;  
  :  
end;
```

James Tam

Solution: Use A Dummy Definition

A "placeholder" for the compiler (definition comes later)

Example problem

```
procedure proc2; FORWARD;
```

```
procedure proc1;  
begin  
  :  
  proc2;  
  :  
end;
```

```
procedure proc2;  
begin  
  :  
  proc1;  
  :  
end;
```

James Tam

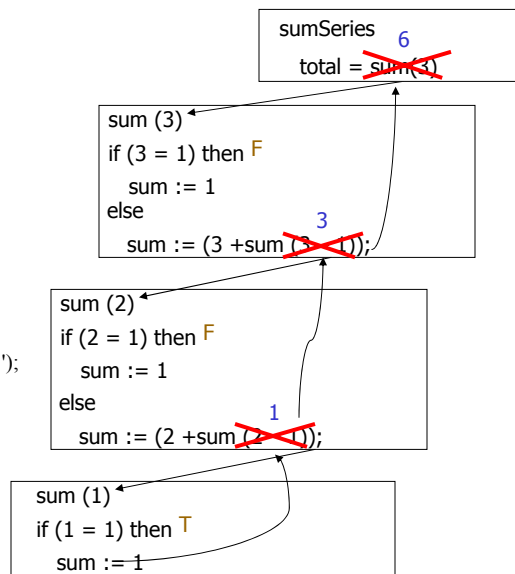
Requirements For Sensible Recursion

- 1) Base case
- 2) Progress is made (towards the base case)

James Tam

Example Program

```
program sumSeries (input, output);  
  
function sum (no : integer): integer;  
begin  
  if (no = 1) then  
    sum := 1  
  else  
    sum := (no + sum (no - 1));  
end;  
  
begin  
  var lastNumber, total : integer;  
  
  write('Enter the last number in the series :');  
  readln(lastNumber);  
  total := sum(lastNumber);  
  writeln('Sum of the series from 1 to  
  `lastNumber, ' is, ' total);  
end.
```



James Tam

When To Use Recursion

- When a problem can be divided into steps.
- The result of one step can be used in a previous step.
- There is scenario when you can stop sub-dividing the problem into steps and return to previous steps.
- All of the results together solve the problem.

James Tam

When To Consider Alternatives To Recursion

- When a loop will solve the problem just as well
- Types of recursion:
 - Tail recursion
 - A recursive call is the last statement in the recursive module.
 - This form of recursion can easily be replaced with a loop.
 - Non-tail recursion
 - A statement which is not a recursive call to the module comprises the last statement in the recursive module.
 - This form of recursion is very difficult to replace with a loop.

James Tam

Drawbacks Of Recursion

Function/procedure calls can be costly

- Uses up memory
- Uses up time

James Tam

Benefits Of Using Recursion

- Simpler solution that's more elegant (for some problems)
- Easier to visualize solutions (for some people and certain classes of problems – typically require either: non-tail recursion to be implemented or some form of “backtracking”)

James Tam

Common Pitfalls When Using Recursion

- These three pitfalls can result in a segmentation fault occurring
 - No base case
 - No progress towards the base case
 - Using up too many resources (e.g., variable declarations) for each function call

James Tam

No Base Case

```
function sum (no : integer): integer;  
begin  
    sum := (no + sum (no - 1));  
end;
```

James Tam

No Base Case

```
function sum (no : integer): integer;  
begin  
  sum := (no + sum (no - 1));  
end;
```

← When does it stop???

James Tam

No Progress Towards The Base Case

```
function sum (no : integer): integer;  
begin  
  if (no = 1) then  
    sum := 1  
  else  
    sum := (no + sum (no));  
end;
```

James Tam

Using Up Too Many Resources

For full example look under
/home/231/examples/recursion/resourceHog.p

```
procedure proc;  
var  
  arr : array [1..1000000] of char;  
begin  
  proc;  
end;
```

James Tam

Undergraduate Definition Of Recursion

Word: **re·cur·sion**

Pronunciation: ri-'k&r-zh&n

Definition: See recursion

You Should Now Know

- What is a recursive computer program
- How to write and trace simple recursive programs
- What are the requirements for recursion/What are the common pitfalls of recursion