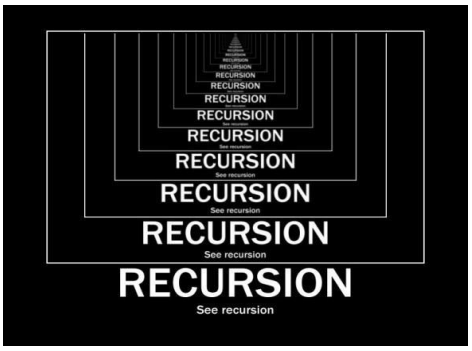
Module 17 - Recursion.

Recursion is REPETITION…



Recursion is a repeating sequence. It is code REUSE.

There are many different programming paradigms, ways to organize your code. Look for the one called recursion in the list of paradigm types on Wikipedia.

Website: http://en.wikipedia.org/wiki/Programming_paradigm

We will take a look at one paradigm called 'Recursion'.

Video Watch me…Recursion - http://www.youtube.com/watch?v=4agL-MQq05E
Video Watch me…Recursion - http://www.youtube.com/watch?v=UFL7GkAHnTQ

The Basic definition of Recursion

Recursion is the repetition of series of steps, and while doing so, ***breaks a problem down into successive smaller parts*** *until they reach a situation where you can no longer break it into smaller parts… This smallest part is called the* ***base*** *or ending case.* ***Recursion is repetitive CODE REUSE. It is implemented by using functions.***

Pros and Cons of Recursion.

You can write repetitive code using regular while or for loops, OR use recursion. Why use recursion or not? Sometimes your program may be shorter or longer, or the performance time may be shorter or longer, depending on weather you use recursion or not. As a programmer, part of your responsibility is to pick the best way to write your code. Matter of fact, there are entire programming languages built entirely around recursion: LISP.

Uses of Repetition

We will take a look at **linear recursion** in this class. Linear recursion is when a function has a **single** call to itself.

Non linear recursion, **two or more** calls to itself, is studied in CIS 252 Introduction to Data Structures.

So far we have concentrated on using loops for *contiguous* repetition.
We have also looked at using Functions for *non-contiguous* repetition.
It may surprise you, but you can also use Functions for contiguous repetition!

- Procedural programming using 'for', 'while' or 'do while' loops for **contiguous** repetition
- A different type of solution using function Recursion for **contiguous** repetition.

Recursion Function: A function which calls itself… until the problem is reduced to its simplest/base case.

Base Case: An 'If condition, with in the body of function, which checks a condition to determines whether to call the called function again or stop calling itself.

What is recursion: http://www.cprogramming.com/tutorial/lesson16.html

Key IDEA: A function can **pass values** into itself, AND **return values**.

Actions can be performed when:

1) Call Direction – Passes values into the function (Use function input parameters)
2) Return Direction – Returns values from a function call (Use function return statement)

Example 1 – One direction recursion – The call


```
void myFunction( int counter)
{
  if(counter == 0)      return;  // if base case condition is true, then
stops, and returns..
  else
  {  // if base case condition is greater than zero, call itself again
cout <<counter<<endl;
     myFunction(--counter); // only -- action performed on call
return; // no action performed on return
  }
}
int main() {

    int x = 12;
    myFunction(x);    //call function

}
```

Assignment 1 – Draw a diagram illustrating the values as the function is called… and the values as each call is retuned….

Use filename: Week#YourNameProg1

Assignment 2 - Write a non-recursive version of this function… using a regular while loop. Use filename: Week#YourNameProg2

Example 2 - Two (2) direction recursion – The call and the return.

Let's consider writing a function to find the factorial of an integer, N!. For example 8! equals 8*7*6*5*4*3*2*1 .

```
int myFactorial( int integer)
{
    if( integer == 1)
    {
        return 1; // Base case action ending recursive looping
    }     else
{
        return (integer * (myFactorial(integer-1)));
        // action performed on call – pass into function "integer – 1"
        // action performed on return  *
    }
}
```

Video Watch me – Recursion Factorial http://www.youtube.com/watch?v=AbxyL_r1cJQ

Assignment 3 – For example 2, draw a diagram illustrating the values as the function is called… and the values as each call is retuned…. Use filename: Week#YourNameProg3

Assignment 4 - For example 2, write a non-recursive version of this function… using a regular while loop.
Use filename: Week#YourNameProg4

Assignment 5 - Write a recursive function that adds up the numbers 1 to 12. Use filename: Week#YourNameProg5

Assignment 6 - Write a non-recursive version of assignment 5… using a regular for loop.
Use filename: Week#YourNameProg6

Questions 12 - Use filename: Week#YourNameQ12

1) Define recursion
2) What is one way recursion ?
3) What is two way recursion ?
4) What is linear recursion ?
5) What is non-linear recursion ?
6) How do you prevent recursion from being an 'Infinite' loop ?

7) What is the recursive base case ?
8) What programming languages are built entirely around recursion ?
9) What is the Tower Of Hanoi problem ? What children's game is it like ?
10) Run the following C++ code to test a recursive version of the Tower of Hanoi (6 disks).
    Https://www.geeksforgeeks.org/implement-tower-of-hanoi-in-cpp/
11) Run the following C++ code to test the non-recursive version of the Tower of Hanoi (6 disks).
    Https://www.geeksforgeeks.org/iterative-tower-of-hanoi/
12) How many lines of code is the recursive version vers the non-recursive version listed here. Which is the shorter code to write ?