Recursion

Last updated 5/8/20

Recursion

- Motivation
  - Recursion is an efficient method to solve some problems

Recursion

- Concept
  - Recursion is when a function calls itself
    - Directly
    - Indirectly – through intermediate functions
  - Requirement
    - Base case to terminate the recursion

      or
    - Terminator

Recursion

- Example
  - print function – no terminator

```
void printme(void){
    cout << “oops”;
    printme();
}
```

  - This will print oops - infinitely

Recursion

- Example
  - print function - terminator

```
/*
 * print_ex.cpp
 *
 *  Created on: May 4, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

void printme(int num);

int main(void){
  printme(4);

  return 0;
}

void printme(int num){
  cout << "entering printme(" << num << ")" << endl;
  while(num != 0){
    cout << "Oi!" << endl;
    printme(num--);
  }
  cout << "printme(" << num <<") is returnimg" << endl;

  return;
}
```

```
entering printme(4)
Oi!
entering printme(3)
Oi!
entering printme(2)
Oi!
entering printme(1)
Oi!
entering printme(0)
printme(0) is returnimg
printme(1) is returnimg
printme(2) is returnimg
printme(3) is returnimg
printme(4) is returnimg
```

Recursion

- Example - Factorial
  - 4! = 4*3*2*1
  - Terminator is multiplier = 0

```
/*
 * factorial_ex.cpp
 *
 *  Created on: May 4, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

int factorial(int num);

int main(void){
  cout << factorial(7) << endl;

  return 0;
}

int factorial(int num){
  int tmp;
  cout << "calling factorial with " << num << endl;

  //Test for base case
  if(num == 0)
    return 1;
  else{
    tmp = num * factorial(num - 1);
    cout << "returning " << tmp << endl;
    return tmp;
  }
}
```

```
calling factorial with 7
calling factorial with 6
calling factorial with 5
calling factorial with 4
calling factorial with 3
calling factorial with 2
calling factorial with 1
calling factorial with 0
returning 1
returning 2
returning 6
returning 24
returning 120
returning 720
returning 5040
5040
```

Recursion

- Example
  - Fibonacci Sequence

  1 1 2 3 5 8 13 21 …

  - Each value is the sum of the previous 2 numbers
  - 1st number is 1
  - Special case for the second number
    - Assign fib(0) = 0
    - 0 1 1 2 3 5 8 13 21 …

    fib(i) = fib(i-1) + fib(i-2)

Recursion

- Example
  - Fibonacci

```
/*
 * fibonacci_ex.cpp
 *
 *  Created on: May 4, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

int fibonacci(int num);
int fib(int num);

int main(void){
  cout << "The first 10 fibonacci numbers are:"  << endl;
  for(int i=0; i<10; i++)
    cout << fibonacci(i) << " ";
  cout << endl;
  cout << "------------" << endl;
  cout << "The first 6 fibonacci numbers are:"  << endl;
  for(int i=0; i<6; i++)
    cout << fib(i) << endl;
  cout << endl;
  return 0;
}
```

```
int fibonacci(int num){
  //Test for base cases
  if(num <=0)
    return 0;
  else if(num == 1)
    return 1;
  else
    return fibonacci(num - 1) + fibonacci(num - 2);
}

int fib(int num){
  int tmp;
  //Test for base cases
  if(num <=0)
    return 0;
  else if(num == 1)
    return 1;
  else{
    tmp = fib(num -1) + fib(num - 2);;
    cout << "returning " << tmp << endl;
    return tmp;
  }
}
```

```
The first 10 fibonacci
numbers are:
0 1 1 2 3 5 8 13 21 34
------------
The first 6 fibonacci numbers
are:
0
1
returning 1
1
returning 1
returning 2
2
returning 1
returning 2
returning 1
returning 3
3
returning 1
returning 2
returning 1
returning 3
returning 1
returning 2
returning 5
5
```

Recursion

- Example
  - Towers of Hanoi
  - Rules:
    - one move at a time
    - small on top

Recursion

- Example
  - Towers of Hanoi

  - To move n disks from Peg 1 to Peg 3 using Peg 2 as a temporary peg
    - if n > 0 then
      - move n-1 disks from peg 1 to peg 2, using peg 3 as a temp
      - move a disk from peg 1 to peg 3
      - move n-1 disks from peg 2 to peg 3 using peg 1 as a temp

Recursion

- Example
  - Towers of Hanoi

```
/*
 * hanoi_ex.cpp
 *
 *  Created on: May 4, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

void hanoi(int num, string source, string dest, string tmp);

int main(void){
  cout << "Towers of Hanoi - 5 disks"  << endl;
  hanoi(5, "peg_1", "peg_3", "peg_2");
  cout << "done!" << endl;

  return 0;
}

void hanoi(int num, string source, string dest, string tmp){
  //Test for base cases
  if(num > 0){
    hanoi(num - 1, source, tmp, dest);

    cout << "moved disk " << num << " from " << source << " to " << dest << endl;

    hanoi(num - 1, tmp, dest, source);
  }
  return;
}
```

```
Towers of Hanoi - 5 disks
moved disk 1 from peg_1 to peg_3
moved disk 2 from peg_1 to peg_2
moved disk 1 from peg_3 to peg_2
moved disk 3 from peg_1 to peg_3
moved disk 1 from peg_2 to peg_1
moved disk 2 from peg_2 to peg_3
moved disk 1 from peg_1 to peg_3
moved disk 4 from peg_1 to peg_2
moved disk 1 from peg_3 to peg_2
moved disk 2 from peg_3 to peg_1
moved disk 1 from peg_2 to peg_1
moved disk 3 from peg_3 to peg_2
moved disk 1 from peg_1 to peg_3
moved disk 2 from peg_1 to peg_2
moved disk 1 from peg_3 to peg_2
moved disk 5 from peg_1 to peg_3
moved disk 1 from peg_2 to peg_1
moved disk 2 from peg_2 to peg_3
moved disk 1 from peg_1 to peg_3
moved disk 3 from peg_2 to peg_1
moved disk 1 from peg_3 to peg_2
moved disk 2 from peg_3 to peg_1
moved disk 1 from peg_2 to peg_1
moved disk 4 from peg_2 to peg_3
moved disk 1 from peg_1 to peg_3
moved disk 2 from peg_1 to peg_2
moved disk 1 from peg_3 to peg_2
moved disk 3 from peg_1 to peg_3
moved disk 1 from peg_2 to peg_1
moved disk 2 from peg_2 to peg_3
moved disk 1 from peg_1 to peg_3
done!
```