Assignment 4

Design: The Tower of Brahma

Linhao Chen
CSE 13S - Fall 2019

Date:10/26/2019

1 Introduction

This program will play the game named The Tower of Hanoi, which have three pegs and the first peg has n-disks, the sizes of them increases from top to bottom, we need to move these disks from first peg to second peg. There are two rules: 1. only one disk can be moved at a time. 2. No larger disk can be placed on top of a smaller disk. The program can exactly achieve this goal by using recursion and stack, which can be selected by user through command line. User also required to enter the number of disks in this game, otherwise the default value will be 5. The game will print the process of disk moving and counts the total steps of game. Expect we will use Command Line, getopt, recursion, stack, etc.

2 Program

Main {
  Set two Boolean variables (check_st and check_re) and one integer variable (num)
Read the command-line option
if less than one options, or the command is unknown, or more than three options: Print Error and program ends
Switch to each case depend on the command-line option {
In case ‘s’:
  check_st equals to true;

In case ‘r’:
  check_re equals to true;

In case ‘n’:
  Set the value of num equals to the parameter;
  If n equals zero, print error;
}

if (check_st is true) do function (st) by providing parameter (n)
if (check_re is true) do function (re) by providing parameter (n)
return 0;

```
Ends
}

Function st {
   create three stacks for peg A, peg B, and peg C (allocate memory and initialize)
   push n disks to peg A
   print necessary format for STACKS
   calculate total steps for iteration (step equals 2^n-1)
   calculate mode equals n % 2(decide use odd mode or even mode)
   set an int array length n to represent binary counter (initial to 0)
   set an int array length n to represent the location of each disk (initial to 0)
   do for loop(loop equals 0; loop smaller than n; loop++)
      set an int variable mark represent the disk we want to move
      set an int variable place represent the current location of disk we want to move
      if counter[n-1] equals to 0 (e.g. binary is 10000)
         counter[n-1] plus one
         mark equals 1
         place equals location[0]
      else
         do a while loop(start with index n-2)
            if counter[index] equals to 0 (e.g. binary is 10011, then it will be 10100)
               do a for loop to set all binary elements after index to 0
               set counter[index] to 1
               mark equals n minus index
               location equals location[mark]
               jump out the while loop
            index --
      test mark is even number or odd number
      if (mark is even number)
         if (mode is even mode)
            do function move_disk (mark, peg origin, peg destination)
            (origin peg depends on the location of current disk, 0 , 1 ,2 represents A, B, C)
            (destination peg is the one which at the right of origin peg, e.g. B to C, C to A)
            update the location of disk
         else
            do function move_disk (mark, peg origin, peg destination)
            (origin peg depends on the location of current disk, 0 , 1 ,2 represents A, B, C)
            (destination peg is the one which at the left of origin peg, e.g. B to A, A to C)
            update the location of disk
      else
         if(mode is even mode)
            do function move_disk (mark, peg origin, peg destination)
            (origin peg depends on the location of current disk, 0 , 1 ,2 represents A, B, C)
            (destination peg is the one which at the left of origin peg, e.g. B to A, A to C)
```

```
            update the location of disk
        else
            do function move_disk (mark, peg origin, peg destination)
            (origin peg depends on the location of current disk, 0 , 1 ,2 represents A, B, C)
            (destination peg is the one which at the left of origin peg, e.g. B to C, C to B)
            update the location of disk
    print the total steps
    delete the stacks for pegs
}

Function re {
    create three stacks for peg A, peg B, and peg C (allocate memory and initialize)
    push n disks to peg A
    print necessary format for RECURSION
    do function (find_solution) by using parameter (n, peg A, peg B, peg C) [return as integer]
    print the total steps by using the value returned by function (find_solution);
    delete the stacks for pegs
}

Function int find_solution {
    set a int variable to express the accumulated sum
    if i = 1
        do function move_disk (n, peg origin(A), peg destination(B))
        (Hint: at this time, peg may not the exactly peg A or B due to exchange the order of them)
        sum ++
    else
        recur on # of disk(n-1) and exchange peg B and peg C, then add the returned value to an
accumulating sum
        do function move_disk (n, peg A, peg B)
        recur on # of disk(n-1) and exchange peg A and peg C, then add the returned value to an
accumulating sum
    return accumulating sum
}

Function move_disk {
    pop the top item from origin peg
    push the item to destination peg
    print this process
}
```