PROBLEM SOLVING WITH C
UE23CS151B

Prof. Sindhu R Pai
Department of Computer Science and Engineering

PROBLEM SOLVING WITH C

Recursion

Prof. Sindhu R Pai
Department of Computer Science and Engineering

PROBLEM SOLVING WITH C

Recursion

1. Introduction
2. Why Recursion?
3. Implementing Recursion – The Stack
4. Arguments and Return Values
5. Practice Programs

Introduction

Recursive Function

- A function that calls itself Directly or indirectly

- Each recursive call is made with a new, independent set of arguments
    - Previous calls are suspended

- Allows building simple solutions for complex problems

Why Recursion?

- Used to solve various problems by dividing it into smaller problems

- Some problems are *too hard* to solve without recursion
  - Most notably, the compiler!
  - Most problems involving linked lists and trees

Points to note while using Recursion

- The problem is broken down into smaller tasks
- Programmers need to be careful to define an exit condition from the function, otherwise results in an infinite recursion
- The exit condition is defined by the ***base case*** and the solution to the base case is provided
- The solution of the bigger problem is expressed in terms of smaller problems called as ***recursive relationship***

How is a particular problem solved using recursion?

- **Problem to be solved: To compute factorial of n.**
  - knowledge of factorial of (n-1) is required, which would form the recursive relationship
  - The base case for factorial would be n = 0
  - return 1 when n == 0 or n == 1 // base case
  - return n*(n-1)! when n!=0 // recursive relationship
  - Demo of C code

Implementing Recursion - The Stack

- **Definition –** ***The Stack***
  - A **last-in, first-out** data structure provided by the operating system for running each program
  - For temporary storage of automatic variables, arguments, function results, and other information
  - The storage for each function call.
  - Every single time a function is called, an area of the stack is reserved for that particular call.
- Known as ***activation record,*** similar to that in python

Implementing Recursion - The Stack continued..

- Parameters, results, and automatic variables allocated on the stack.
- Allocated when function or compound statement is entered
- Released when function or compound statement is exited
- Values are not retained from one call to next (or among recursions)

Arguments and Return values

1. Space for storing result is allocated by caller
   - On The Stack
   - Assigned by **return** statement of function
   - For use by caller
2. Arguments are values calculated by caller of function
   - Placed on The Stack by caller in locations set aside for the corresponding parameters
   - Function may assign new value to parameter
   - caller never looks at parameter/argument values again!
3. Arguments are removed when callee returns
   - Leaving only the result value for the caller

Practice Programs based on Recursion

1. Separate Recursive functions to reverse a given number and reverse a given string
2. Recursive function to print from 1 to n in reverse order
3. Find the addition, subtraction and multiplication of two numbers using recursion. Write separate recursive functions to perform these.
4. Find all combinations of words formed from Mobile Keypad.
5. Find the given string is palindrome or not using Recursion.
6. Find all permutations of a given string using Recursion

THANK YOU

Prof. Sindhu R Pai

Department of Computer Science and Engineering

sindhurpai@pes.edu