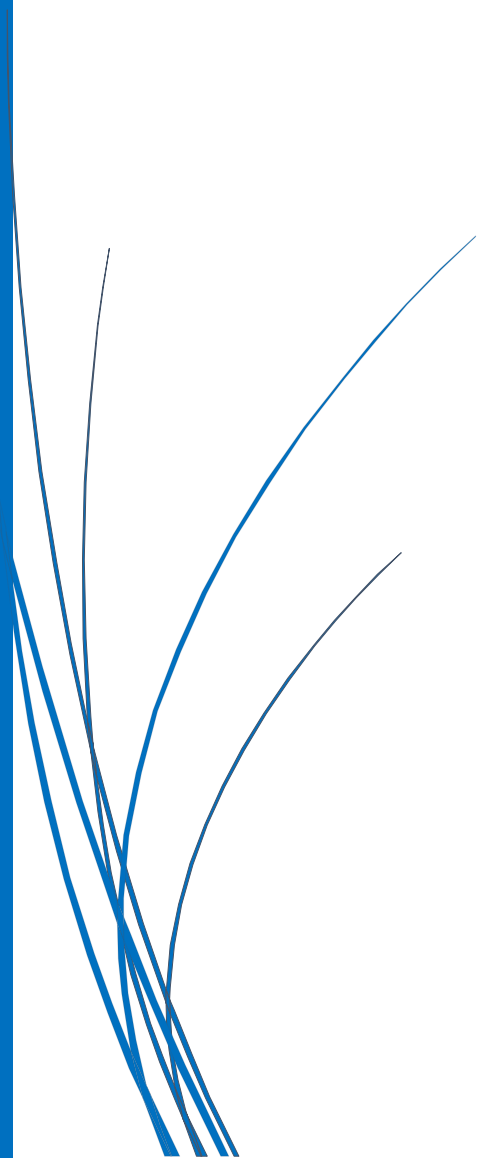
Made By:
Kenan Gazwan

Recursion

ICS202-Summary

King Fahd University of
Petroleum and Minerals



Telegram: @KenanGazwan

(Recursion)

Recursion is a technique that allows us to break down a problem into one or more simpler sub-problems that are similar in form to the original problem.

✓ Recursive Definition

It is a programming concept that defines itself.

✓ Recursive Data Structure

It is a data structure that is partially composed of smaller or simpler instances of the same data structure, such as Linked Lists and Binary Trees.

✓ Recursive Method

It is a method that calls itself either directly or indirectly.

The Recursive Method Parts:

Each Recursive Method has two main parts:

1) **Base Case**: the stopping condition
2) **The Recursive Case:** the solution of the problem expressed in 2 or more smaller parts.

Ex: return n * factorial(n – 1)
The method call is a first part & the arithmetic operation is a second part.

Tracing of Recursive Methods

A recursive method may be traced using the recursion tree it generates.

✓ Auxiliary (or Helper) Methods

Auxiliary or helper methods are used for one or more of the following reasons:

- ✓ To make recursive methods more efficient
- ✓ To make the user interface to a method simpler by hiding the method's initializations

✓ Direct and Indirect (mutually) Recursive Methods

- ✓ A method is directly recursive if it contains an explicit call to itself.
- ✓ A method x is indirectly recursive if it contains a call to another method which in turn calls x.

Ex: **MethodX calls MethodY**
MethodY calls MethodX

✓ Nested Recursive Methods

Nested Recursive is a method that calls itself and uses itself as a parameter.

Ex:

```
Example(int n, int m){
        if(-----)
                return ----;
        else
                return 1 + Example(2, Example(2, 4));
}
```

✓ Tail and Non-Tail Recursive Methods

Tail Recursive Method:
A method is tail recursive if in each of its recursive cases it executes one recursive call and there are no pending operations after that call.

Non-Tail Recursive Method:
A recursive method is non-tail recursive if at least one of the following holds:

- ✓ In at least one of its recursive cases, a recursive call is followed by another statement.
- ✓ In at least one of its recursive cases, it has a recursive call that is part of an expression.
- ✓ In at least one of its recursive cases, it has more than one recursive call.

✓ Why Tail Recursion?

It is desirable to have tail-recursive methods because:

- The amount of information that gets stored during computation is independent of the number of recursive calls.
- Some compilers can produce optimized code that replaces tail recursion by iteration.
  In general, an iterative version of a method is executed more efficiently in terms of time and space than a recursive version.

✓ Method Calls and Recursion Implementation

When a method is called an Activation Record is created. It contains:

- ✓ The values of the parameters.
- ✓ The values of the local variables.
- ✓ The return address (The address of the statement after the call statement).
- ✓ The previous activation record address.
- ✓ A location for the return value of the activation record.

When a method returns:

- ✓ The return value of its activation record is passed to the previous activation record, or it is passed to the calling statement if there is no previous activation record.
- ✓ The Activation Record is popped entirely from the stack.

✓ Common Errors in Writing Recursive Methods

- **The method does not call itself directly or indirectly.**
- **Non-terminating Recursive Methods (Infinite recursion)**
  - **Either no base case or the base case is never reached.**

☒ **Using Post-Increment or decrement instead of Pre-Increment or decrement.**

- **Post-Increment =** `method(index++)`
- **Pre-Increment =** `method(++index)`

☒ **Initialize local variables to accumulate the result of a recursive method. (Each call has its own copy of a local variable).**

- **The variable must be initialized in the helper method or passed each call without new initialization.**

☒ **Wrong placement of return statement.**

✓ Why Recursion ?

- Less Code Written → Easier to Understand
- Simpler Solution (Sometimes)
- The Most Suitable Choice for Recursive Data Structures.
- In Some Programming Languages, there are np loops.

✓ Disadvantages of Recursion ?

- It has greater space requirements than an equivalent iterative program as each function call activation record will remain in the stack until the base case is reached.
- Sometimes, it may have greater time requirements because each time the function is called, the stack grows, and the final answer is returned when the stack is popped completely.