

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Recursive Functions – Part A

Quick Recap of Relevant Topics



- Use of simple functions in programs
- Contract-centric view of programming with functions
- Flow of control in function call and return
- Activation records and call stack
- Parameter passing by value and reference

So far, the caller and callee were different functions!

Overview of This Lecture



- Recursive functions
 - How they work
 - Activation records and call stack

Recall: Encoding Example



- We want to store quiz 1 and quiz 2 marks of CS101 students in an encoded form
- Encoding strategy: $\text{encode}(m, n) = 2^m \times 3^n$
- Assume all marks are integers in $\{1, 2, \dots, 10\}$

Recall: C++ Program Structure

<pre>#include <iostream> using namespace std; int main() { ... // PRECONDITION: ... int myEncode(int q1Marks, int q2Marks) { ... twoRaisedQ1 = power(2, q1Marks); threeRaisedQ2 = power(3, q2Marks); ... } // POSTCONDITION: ... cipher = myEncode(q1Marks, q2Marks); ... } // PRECONDITION: ... int power(int base, int exponent) { ... } // POSTCONDITION: ...</pre>	<p>Function call from within a different function</p> <p>Function call from within a different function</p>
--	---

Calling a Function from Itself



Can we have a function call itself?

Can we have two functions mutually call each other?

Why not? What could possibly go wrong?

Recall: Encoding Example



- We want to store quiz 1 and quiz 2 marks of CS101 students in an encoded form
- Encoding strategy: $\text{encode}(m, n) = 2^m \times 3^n$
- Assume all marks are integers in $\{1, 2, \dots, 10\}$

Observe: $\text{encode}(m, n) = \text{encode}(m, n-1) \times 3$, if $m, n > 1$
 $= \text{encode}(m-1, 1) \times 2$, if $m > 1, n=1$
 $= 2 \times 3 = 6$, if $m=1, n=1$

An Alternate Encoding Program

```
#include <stdio.h>
using namespace std;

int newEnc(int q1Marks, int q2Marks)
{
    switch(q2Marks) {
        case 1:
            if (q1Marks == 1) {return 6;}
            else {return
                2*newEnc(q1Marks - 1, 1);
            }
        break;
        default: ... }
    }
}

// PRECONDITION: ...
int main() { ...
    for ( ... ) { ...
        cipher = newEnc(q1Marks, q2Marks);
    }
    ...
    return cipher;
}

// POSTCONDITION: ...
```

Function call from within a different function

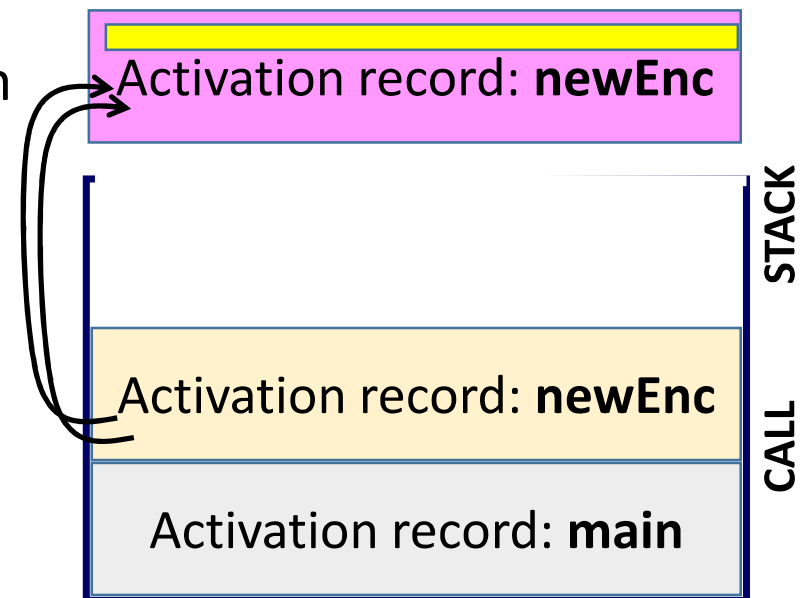
Function call from within the same function

Recall: Activation Records in Call Stack

When a caller (**newEnc**) calls a callee (**newEnc**)

- a **fresh** activation record for callee created
- Values of function parameters from caller copied to space allocated for formal parameters of callee
- PC of caller saved
- Other book-keeping information updated
- Activation record for callee pushed on call stack

```
int newEnc(int q1Marks, int q2Marks)
{ ....
  return 2*newEnc(q1Marks - 1, 1);
...}
```

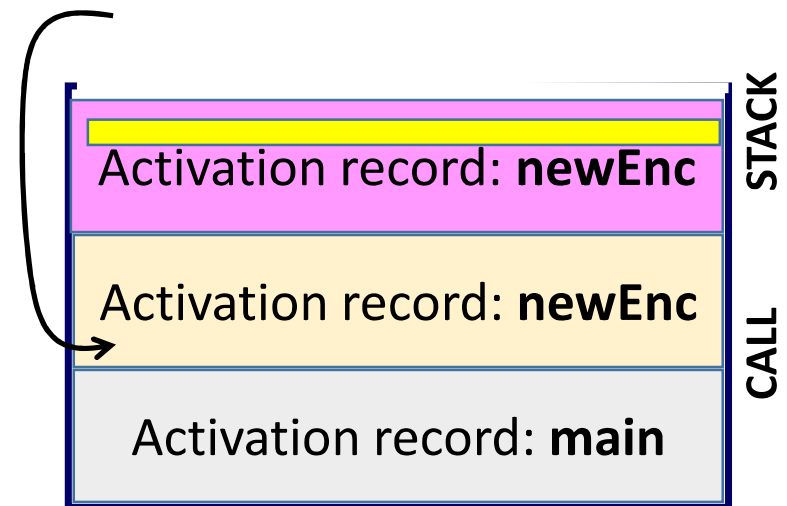


Activation Records in Call Stack

When a callee (**newEnc**) returns

- Callee's activation record popped from call stack
- Return value from popped activation record copied to activation record of caller (now on top of stack)
- Value of PC saved in popped activation record loaded in PC of CPU
- Free activation record of callee
- Resume execution of instruction at location given by updated PC

```
int newEnc(int q1Marks, int q2Marks)
{ ....
  return 2*newEnc(q1Marks - 1, 1);
...}
```



Calling a Function From Itself

- There doesn't seem to be any problem !!!
- Same mechanism of function calls and returns we studied earlier works perfectly !!!

Recursive Function: One that can call itself
Elegant and natural way to solve several problems

Mutually recursive functions
func1 calls func2, which calls func3,
which calls func1

A Recursive Solution: All Is Well So Far!



```
#include <iostream>
using namespace std;
int newEnc(int q1Marks,int q2Marks);
int main() { ...
    for ( ... ) { ...
        cipher = newEnc(q1Marks, q2Marks);
    ...}
    ...
    return 0;
}
```

```
// PRECONDITION: ...
int newEnc(int q1Marks,
           int q2Marks)
{ switch(q2Marks) {
    case 1:
        if (q1Marks == 1) {return 6;}
        else {return
            2*newEnc(q1Marks - 1, 1);
        }
        break;
    default: ... }
}
// POSTCONDITION: ...
```

Summary



- Recursion as a programming construct
- Activation records and call stack in recursive calls