

Recursion II

CS2263 – Systems Software Development

Learning Outcomes

At the conclusion of this lecture students should be able to:

- List benefits of using recursion
- List the downsides of using recursion
- Evaluate whether to use recursion or iteration for a problem

References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 12, 13 and 15!)

Recursive function

```
return_type func (arguments) {  
    if (this is the base case) {  
        solve the problem  
    }  
    else {  
        func(simplified arguments)  
    }  
}
```

Simple programming examples (Ch. 13)

- Selecting balls with restrictions
- Towers of Hanoi
- Factorial
- Fibonacci numbers

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x992	
	n	0x993	2
	Return Location	0x994	RL2
Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x98f	
	n	0x990	1
	Return Location	0x991	RL2
Frame	Symbol	Address	Value
fac	val	0x992	
	n	0x993	2
	Return Location	0x994	RL2
Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x98c	1
	n	0x98d	0
	Return Location	0x98e	RL2
Frame	Symbol	Address	Value
fac	val	0x98f	
	n	0x990	1
	Return Location	0x991	RL2
Frame	Symbol	Address	Value
fac	val	0x992	
	n	0x993	2
	Return Location	0x994	RL2
Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x98f	1
	n	0x990	1
	Return Location	0x991	RL2
Frame	Symbol	Address	Value
fac	val	0x992	
	n	0x993	2
	Return Location	0x994	RL2
Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x992	2
	n	0x993	2
	Return Location	0x994	RL2
Frame	Symbol	Address	Value
fac	val	0x995	
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac	val	0x995	6
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Factorial

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) return 1

    val = n * fac(n - 1);
    return val; // RL2
}

int main(int argc, char** argv){
    int x = 3;
    int f = fac(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
main	f	0x998	6
	x	0x999	3

Factorial: Recursive vs Iterative

Recursive

```
#include <stdio.h>

int fac(int n) {
    int val;
    if (n == 0) {
        val = 1;
        return val;
    }
    val = n * fac(n-1);
    return val; // RL2
}

int main(int argc, char* argv[]){
    int x = 3;
    int f = fac(x);
    printf("%d\n", f); // RL1
}
```

Iterative

```
#include <stdio.h>

int fac2(int n) {
    int val = 1 ;
    if (n == 0) {
        val = 1;
        return val;
    }
    while (n > 0) {
        val *= n;
        n --;
    }
    return val;
}

int main(int argc, char* argv[]){
    int x = 3;
    int f = fac2(x);
    printf("%d", f); // RL1
}
```

Iterative Function: Factorial

```
#include <stdio.h>

int fac2(int n) {
    int val = 1 ;
    if (n == 0) return 1;
    while (n > 0) {
        val *= n;
    }
    return val;
}

int main(int argc, char * argv[]) {
    int x = 3;
    int f = fac2(x);
    printf("%d", f); // RL1
}
```

Frame	Symbol	Address	Value
fac2	val	0x995	6
	n	0x996	3
	Return Location	0x997	RL1
Frame	Symbol	Address	Value
main	f	0x998	
	x	0x999	3

Recursive Function: Fibonacci

```
#include <stdio.h>

int fib(int n) {
    if ((n == 1) || (n == 2)) {
        return 1;
    }
    return fib(n-1) + fib(n-2);
}

int main(int argc, char * argv[]) {
    int x = 5;
    for (int i=1; i<=x; i++) {
        int f = fib(i);
        printf("%d ", f); }
}
```

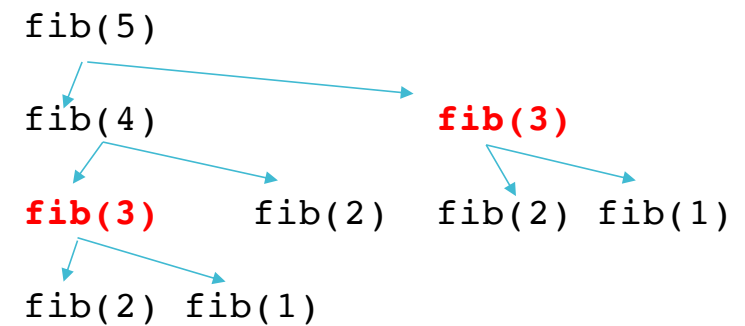


1 1 2 3 5

Recursive Function: Fibonacci

```
int fib(int n) {  
    if ((n == 1) || (n == 2)) {  
        return 1;  
    }  
    return fib(n-1) + fib(n-2);  
}
```

fib(5)



Thoughts on Recursion (I)

- Compute time?
- Memory?
- So why do we do it?

Thoughts on Recursion (II)

- Other examples in the text
- Recursion versus loops
 - *"You could do that." -- Rick*