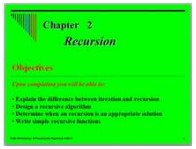


CS 1037
Computer Science Fundamentals II

Part Eight:
Recursion

1

Recursive Definitions

- ***Recursion***:
  defining something *in terms of itself*
- ***Recursive definition***
  - Uses the word or concept being defined ***in the definition itself***
  - Includes a ***base case*** that is defined directly, ***without*** self-reference

Recursive Definitions

- A recursive definition consists of two parts:
  - The ***base case***: this defines the "***simplest***" case or starting point
  - The ***recursive part***: this is the "***general case***", that describes all the other cases in terms of "***smaller***" versions of itself
- Why is a base case needed?
  - A definition without a non-recursive part causes ***infinite recursion***

Recursive Definitions

- Mathematical formulas are often expressed recursively
- ***Example***: the formula for ***factorial***
  *for any positive integer **n**, **n!** (n factorial) is defined to be the product of all integers between **1** and **n** inclusive*
- Express this definition recursively

  1! = 1 (the base case)
  n! = n * (n-1)! for n>=2

- Now determine the value of **4!**

Discussion

- ***Recursion*** is an alternative to ***iteration***, and can be a very powerful problem-solving technique
- What is ***iteration***? repetition, as in a loop
- What is ***recursion***? defining something in terms of a ***smaller*** or ***simpler*** version of itself (why smaller/simpler?)

Recursive Programming

- ***Recursion*** is a programming technique in which a method can ***call itself*** to solve a problem
- A function in C that invokes itself is called a ***recursive method*** and must contain code for
  - The ***base case***
  - The ***recursive part***

Example of Recursive Programming

- Consider the problem of computing the sum of all the numbers between **1** and **n** inclusive

  e.g. if **n** is **5**, the sum is (in an iterative processes)

  1 + 2 + 3 + 4 + 5

- How can this problem be expressed recursively?

  Hint: the above sum is the same as

  5 + 4 + 3 + 2 + 1

Recursive Definition of Sum of **1** to **n**

$$\sum_{k=1}^{n} k = n + \sum_{k=1}^{n-1} k \qquad \text{for } n > 1$$

This reads as:

the sum of 1 to n = n + the sum of 1 to n-1

What is the base case?
the sum of 1 to 1 = 1

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
totalSum	444 - 447	0
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
totalSum	444 - 447	0
k	448 - 451	1
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
totalSum	444 - 447	0
k	448 - 451	1
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
totalSum	444 - 447	6
k	448 - 451	1
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumIter(int );

int main()
{
   int i = 3 , x;

   x = SumIter(i);
   printf("\nIteration x: %d\n", x);

   return 0;
}

int SumIter(int ir)
{
    if (ir == 1)
      return (1);
    else
    {
        int totalSum = 0;
        for (int k = 1; k <= ir; k++)
        {
            totalSum = totalSum + k;
        }
        return (totalSum);
    }
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

OUTPUT:
Iteration x: 6

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
ir	468 - 471	1
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
ir	468 - 471	1
result	472 - 475	
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
ir	468 - 471	1
result	472 - 475	1
	...	
	...	
	...	
	...	
	...	
	...	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

2 + 1

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

3

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	3
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	3
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    3 + 3
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    6
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

6

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```c
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}

int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        reault = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	6
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

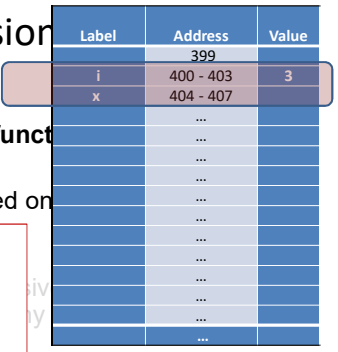
OUTPUT:
Factorial x: 6

How Recursion Works

- What happens when ***any*** **function** is called?
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

- What happens when a recursive method "***calls itself***"?
  It's actually just like calling any other method!
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

How Recursion

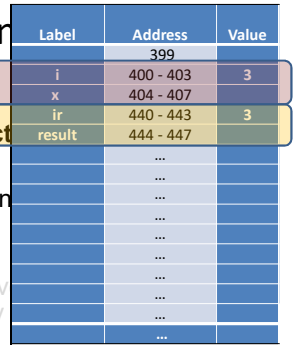


- What happens when ***any*** **funct**
  - A ***call frame*** is set up
  - That call frame is pushed on

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}
int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        result = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

onto the ***runtime stack***

How Recursion



- What happens when ***any*** **funct**
  - A ***call frame*** is set up
  - That call frame is pushed on

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 3 , x;

   x = SumRec(i);
   printf("\nFactorial x: %d\n", x);

   return 0;
}
int SumRec(int ir)
{
    int result;
    if (ir == 1)
    {
        result = 1;
    }
    else
    {
        result = (ir + SumRec(ir - 1));
    }
    return (result);
}
```

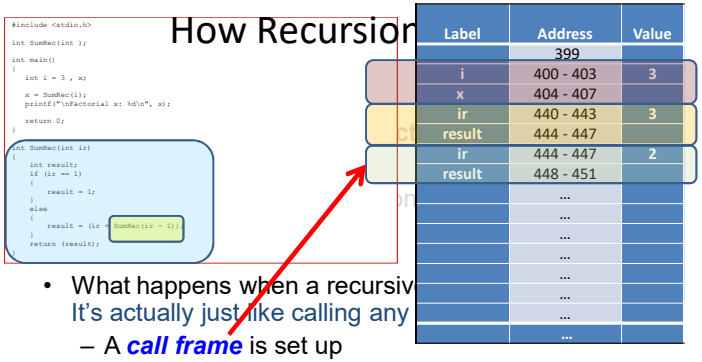
onto the ***runtime stack***

How Recursion Works

- What happens when ***any*** **function** is called?
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

- What happens when a recursive method "***calls itself***"?
  It's actually just like calling any other method!
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

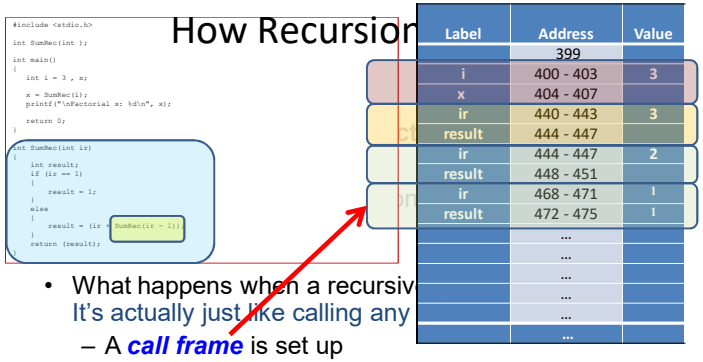
How Recursion



Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

- What happens when a recursiv
  It's actually just like calling any
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

How Recursion



Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	
ir	440 - 443	3
result	444 - 447	
ir	444 - 447	2
result	448 - 451	
ir	468 - 471	1
result	472 - 475	1
	...	
	...	
	...	
	...	
	...	
	...	

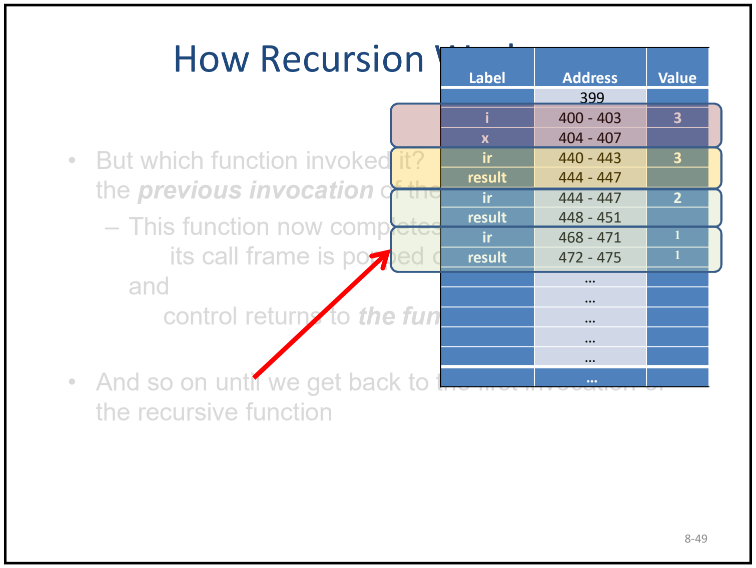
- What happens when a recursiv
  It's actually just like calling any
  - A ***call frame*** is set up
  - That call frame is pushed onto the ***runtime stack***

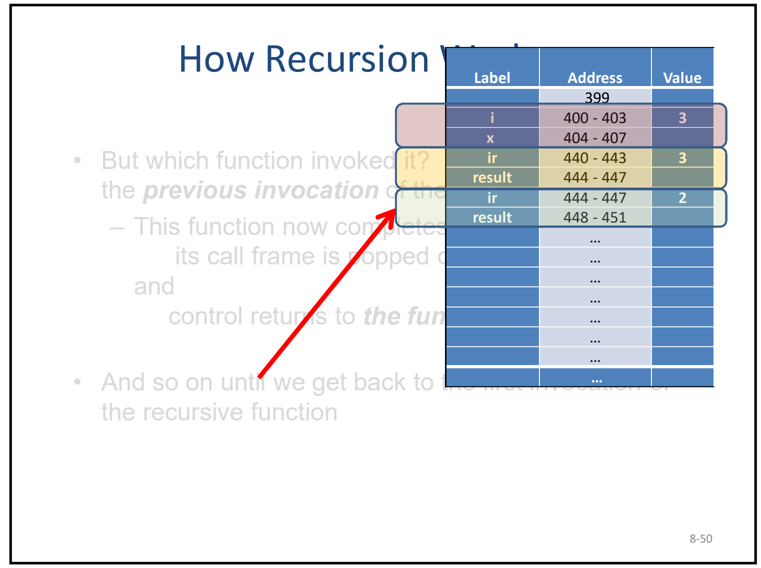
How Recursion Works

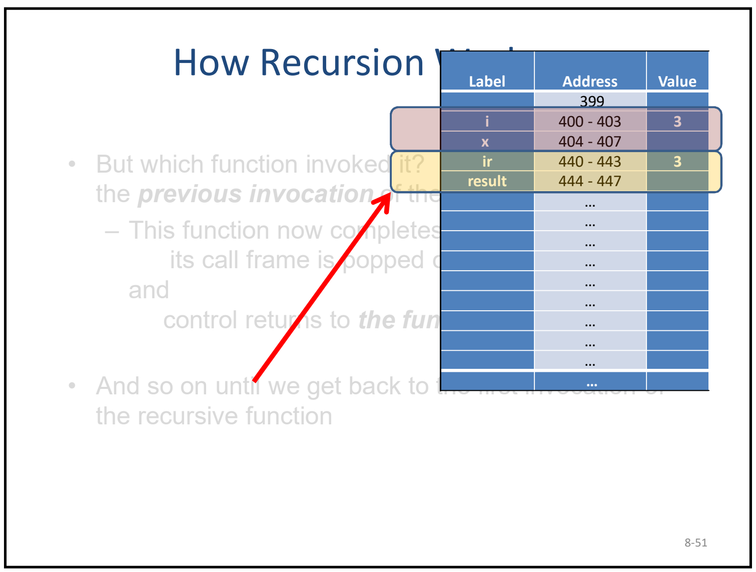
- Note: For a recursive **function** , how many copies of the code are there?
  - Just one! (like any other **function**)
- When does the recursive function stop calling itself?
  - When the base case is reached
- What happens then?
  - ***That invocation*** of the function completes,
    its call frame is popped off the runtime stack,
    and
    control returns to ***the function that invoked it***

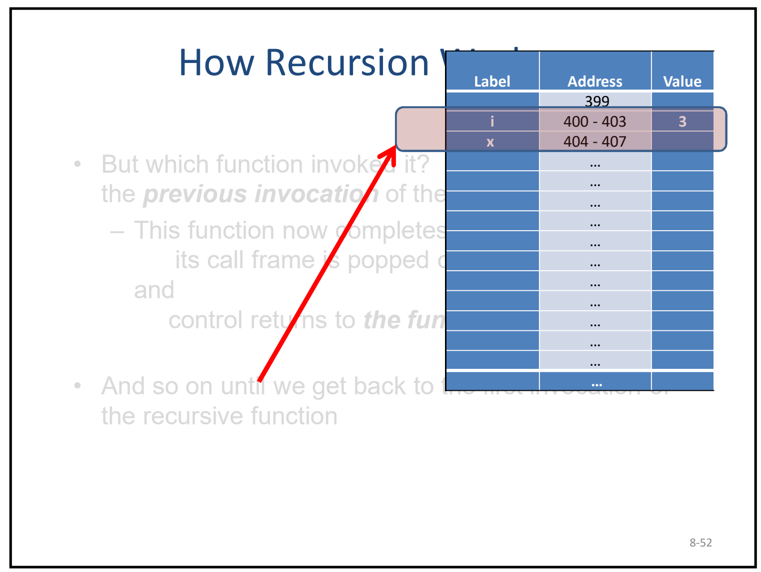
How Recursion Works

- But which function invoked it?
  the ***previous invocation*** of the recursive function
  - This function now completes,
    its call frame is popped off the runtime stack,
    and
    control returns to ***the function that invoked it***
- And so on until we get back to the first invocation of the recursive function


How Recursion
Label | Address | Value
399
i | 400 - 403 | 3
x | 404 - 407
ir | 440 - 443 | 3
result | 444 - 447
ir | 444 - 447 | 2
result | 448 - 451
ir | 468 - 471 | 1
result | 472 - 475
But which function invoked it?
the previous invocation
– This function now completes
its call frame is popped
and
control returns to the fun
And so on until we get back to
the recursive function
8-49


How Recursion
Label | Address | Value
399
i | 400 - 403 | 3
x | 404 - 407
ir | 440 - 443 | 3
result | 444 - 447
ir | 444 - 447 | 2
result | 448 - 451
But which function invoked it?
the previous invocation
– This function now completes
its call frame is popped
and
control returns to the fun
And so on until we get back to
the recursive function
8-50


How Recursion
Label | Address | Value
399
i | 400 - 403 | 3
x | 404 - 407
ir | 440 - 443 | 3
result | 444 - 447
But which function invoked it?
the previous invocation of the
– This function now completes
its call frame is popped
and
control returns to the fun
And so on until we get back to
the recursive function
8-51


How Recursion
Label | Address | Value
399
i | 400 - 403 | 3
x | 404 - 407
But which function invoked it?
the previous invocation of the
– This function now completes
its call frame is popped
and
control returns to the fun
And so on until we get back to
the recursive function
8-52

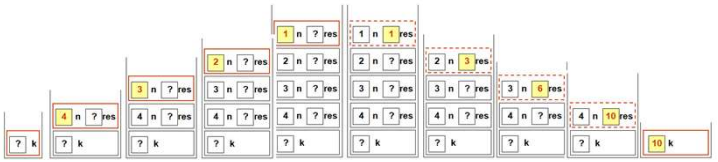
How Recursion

Label	Address	Value
	399	
i	400 - 403	3
x	404 - 407	6
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

- But which function invoked it? the ***previous invocation*** of the
  - This function now completes its call frame is popped and control returns to ***the fun***
- And so on until we get back to ... the recursive function

8-53

```
#include <stdio.h>

int SumRec(int );

int main()
{
   int i = 4 , k;

   k = SumRec(i);
   printf("\nFactorial k: %d\n", x);

   return 0;
}

int SumRec(int n)
{
    int res;
    if (n == 1)
    {
        res = 1;
    }
    else
    {
        res = (n + SumRec(n - 1));
    }
    return (res);
}
```



2-1 Factorial - A Case Study

We begin the discussion of recursion with a case study and use it to define the concept.
This section also presents an iterative and a recursive solution to the factorial algorithm.

- Recursive Defined
- Recursive Solution

Data Structures: A Pseudocode Approach with C 55

$$\text{Factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1 & \text{if } n > 0 \end{cases}$$

FIGURE 2-1 Iterative Factorial Algorithm Definition

Data Structures: A Pseudocode Approach with C 56

$$\text{Factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times (\text{Factorial}(n-1)) & \text{if } n > 0 \end{cases}$$

FIGURE 2-2 Recursive Factorial Algorithm Definition

ALGORITHM 2-1 Iterative Factorial Algorithm

```
Algorithm iterativeFactorial (n)
Calculates the factorial of a number using a loop.
   Pre  n  is the number to be raised factorially
   Post n! is returned
1 set i to 1
2 set factN to 1
3 loop (i <= n)
   1  set factN to factN * i
   2  increment i
4 end loop
5 return factN
end iterativeFactorial
```


ALGORITHM 2-2 Recursive Factorial

```
Algorithm recursiveFactorial (n)
Calculates factorial of a number using recursion.
   Pre    n  is the number being raised factorially
   Post   n! is returned
1 if (n equals 0)
   1  return 1
2 else
   1  return (n * recursiveFactorial (n - 1))
3 end if
end recursiveFactorial
```

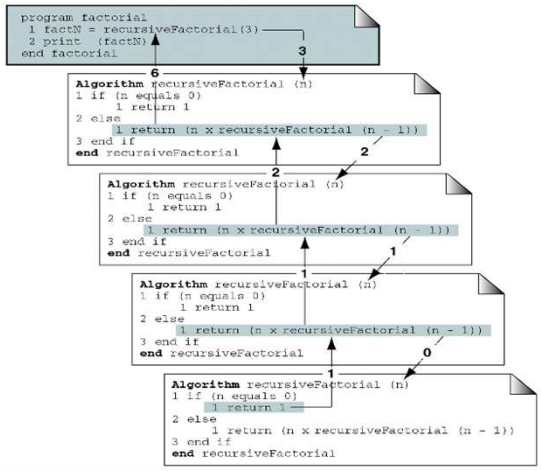



FIGURE 2-4 Calling a Recursive Algorithm