Recursion

Recursion

- Recursion is a fundamental programming technique that can provide an elegant solution certain kinds of problems
- We will focus on:
    - thinking in a recursive manner
    - programming in a recursive manner
    - the correct use of recursion
    - recursion examples

Outline

Recursive Thinking

Recursive Programming

Using Recursion

Recursion in Graphics

Recursive Thinking

- A recursive definition is one which uses the word or concept being defined in the definition itself
- When defining an English word, a recursive definition is often not helpful
- But in other situations, a recursive definition can be an appropriate way to express a concept
- Before applying recursion to programming, it is best to practice thinking recursively

Recursive Definitions

- Consider the following list of numbers:
- 24, 88, 40, 37
- Such a list can be defined as follows:

  A List is a: number

  or a: number comma List
- That is, a List is defined to be a single number, or a number followed by a comma followed by a List
- The concept of a List is used to define itself

Recursive Definitions

- The recursive part of the LIST definition is used several times, terminating with the non-recursive part:

```
LIST:  number   comma   LIST
         24       ,     88, 40, 37
                        number   comma   LIST
                          88       ,     40, 37
                                         number   comma   LIST
                                           40       ,     37
                                                          number
                                                            37
```

Infinite Recursion

- All recursive definitions have to have a non-recursive part called the **base case**

- If they didn't, there would be no way to terminate the recursive path
- Such a definition would cause infinite recursion
- This problem is similar to an infinite loop, but the non-terminating "loop" is part of the definition itself
- **You must always have some base case which can be solved without recursion**

Outline

Recursive Thinking

Recursive Programming

Using Recursion

Recursion in Graphics

Recursive Programming

- **A recursive method is a method that invokes itself**
- A recursive method must be structured to **handle both the base case and the recursive case**
- Each call to the method sets up a new execution environment, with new parameters and local variables
- As with any method call, when the method completes, control returns to the method that invoked it (which may be an earlier invocation of itself)

Sum of 1 to N

- Consider the problem of computing the sum of all the numbers between 1 and any positive integer N
- This problem can be recursively defined as:

$$\sum_{i=1}^{N} i \;=\; N + \sum_{i=1}^{N-1} i \;=\; N + N-1 + \sum_{i=1}^{N-2} i$$

$$= N + N-1 + N-2 + \sum_{i=1}^{N-3} i$$

$$\vdots$$

$$= N + N-1 + N-2 + \cdots + 2 + 1$$

Sum of 1 to N

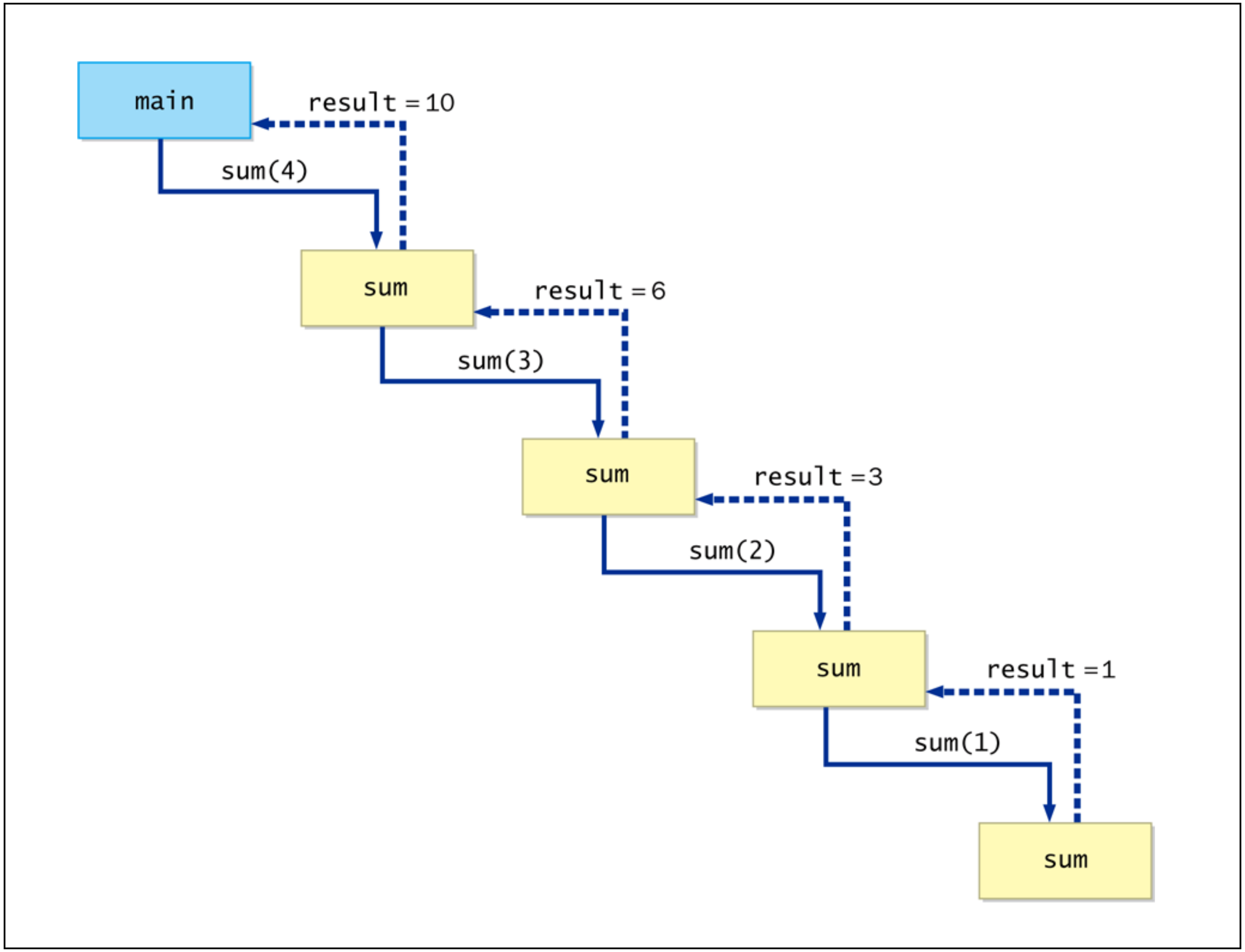
- The summation could be implemented recursively as follows:

```
// This method returns the sum of 1 to num
public int sum (int num)
{
   int result;

   if (num == 1)
      result = 1;
   else
      result = num + sum (n-1);

   return result;
}
```

Sum of 1 to N


main
result = 10
sum(4)
sum
result = 6
sum(3)
sum
result = 3
sum(2)
sum
result = 1
sum(1)
sum

Recursive Programming

- **Note that just because we can use recursion to solve a problem, doesn't mean we should**

- We usually would not use recursion to solve the summation problem, because the iterative version is easier to understand
- However, for some problems, recursion provides an elegant solution, often cleaner than an iterative version
- You must carefully decide whether recursion is the correct technique for any problem

Recursive Factorial

- N!
- For any positive integer N, is defined to be the product of all integers between 1 and N inclusive
- This definition can be expressed recursively as:

  1! = 1

  N! = N * (N-1)!
- A factorial is defined in terms of another factorial
- Eventually, the base case of 1! is reached