Chapter 8 (Part 3)

## High Level Programming Languages

Chapter Goals

- Define the concepts of a data type and strong typing

- Explain the concept of a parameter and distinguish between value and reference parameters

- Describe two composite data-structuring mechanisms

- Name, describe, and give examples of the three essential ingredients of an object-oriented language

- . . . Some Hands-On

A Little Hands On
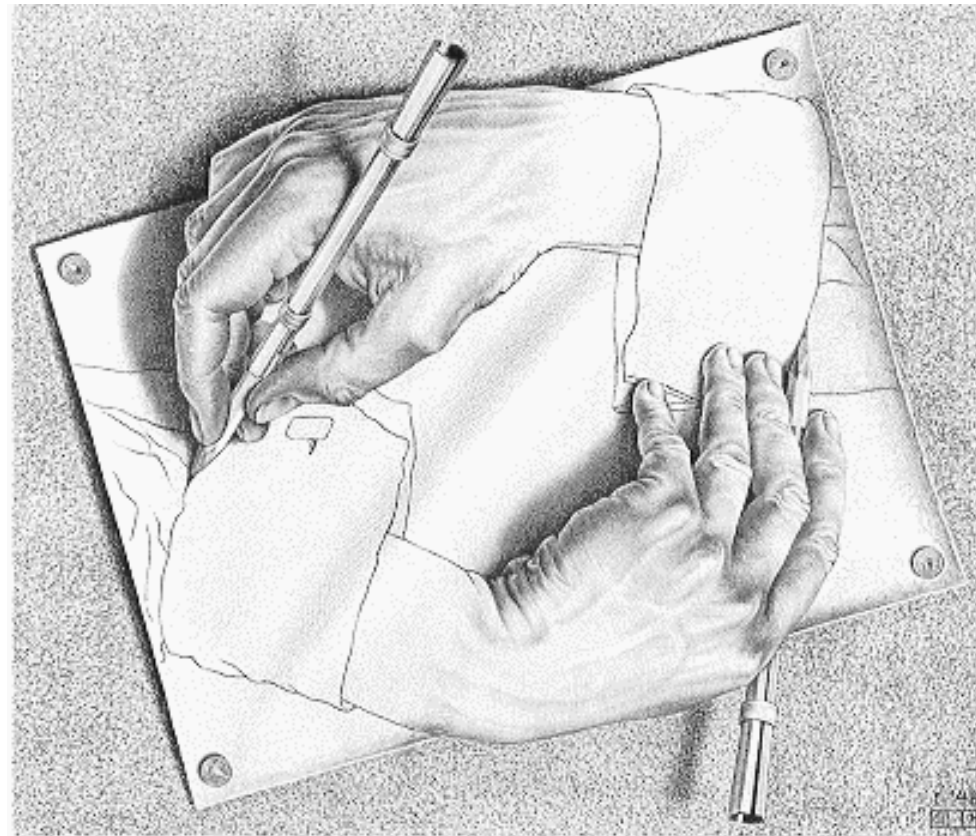
Hello World

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Looping Statements

- The ***while*** statement is used to repeat a course of action

- Let's look at two distinct types of repetitions

Looping Statements

- ***Count-controlled loops***
    - Repeat a specified number of times
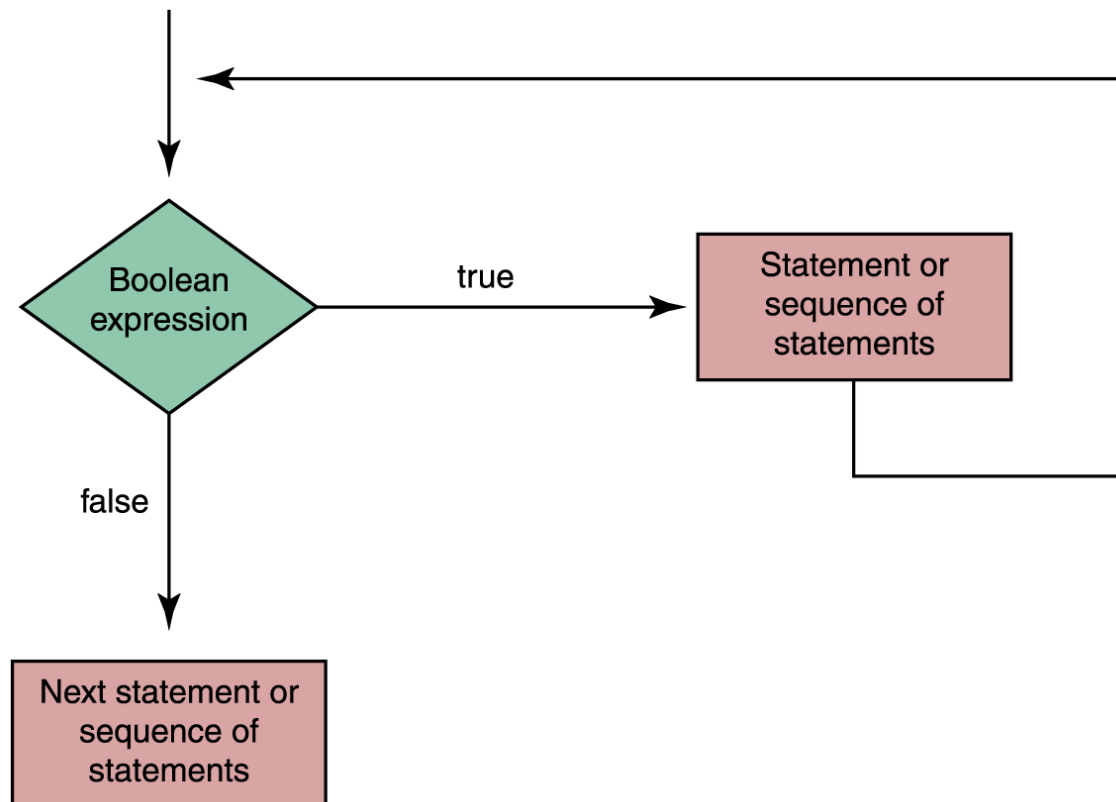   - Use of a special variable called a loop control variable

Figure 8.4
Flow of control of *while* statement

Looping Statements

- *Count-controlled* loops

Language	Count-Controlled Loop with a while Statement
Ada	```Count := 1;` `while Count <= Limit loop` ` ...` ` Count := Count + 1;` `end loop;```
VB.NET	```Count = 1` `While (count <= limit)` ` ...` ` count = count + 1` `End While```
C++/Java	```count = 1;` `while (count <= limit)` `{` ` ...` ` count = count + 1;` `}```

Looping Statements

- ***Event-controlled* loops**
  - The number of repetitions is controlled by an event that occurs within the body of the loop itself

Read a value	Initialize event
While (value >= 0)	Test event
...	Body of loop
Read a value	Update event
...	Statement(s) following loop

Looping Statements

- *Event-controlled* loops

Set sum to 0   Initialize sum to zero
Set posCount to 0   Initialize event
While (posCount <= 10)   Test event
    Read a value
    If (value > 0)   Test to see if event should be updated
        Set posCount to posCount + 1   Update event
        Set sum to sum + value   Add value into sum
...   Statement(s) following loop

Page 249

Looping Statement

```
<html>
<body>
<script type="text/javascript">
var i=0
while (i<=10)
{
document.write("The number is " + i)
document.write("<br />")
i=i+1
}
</script>
</body>
</html>
```

Looping Statement

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10

Subprogram Statements

- We can give a section of code a name and use that name as a statement in another part of the program

- When the name is encountered, the processing in the other part of the program halts while the named code is executed

Subprogram Statements

- There are times when the calling unit needs to give information to the subprogram to use in its processing

- A **parameter list** is a list of the identifiers with which the subprogram is to work, along with the types of each identifier placed in parentheses beside the subprogram name

Subprogram Statements

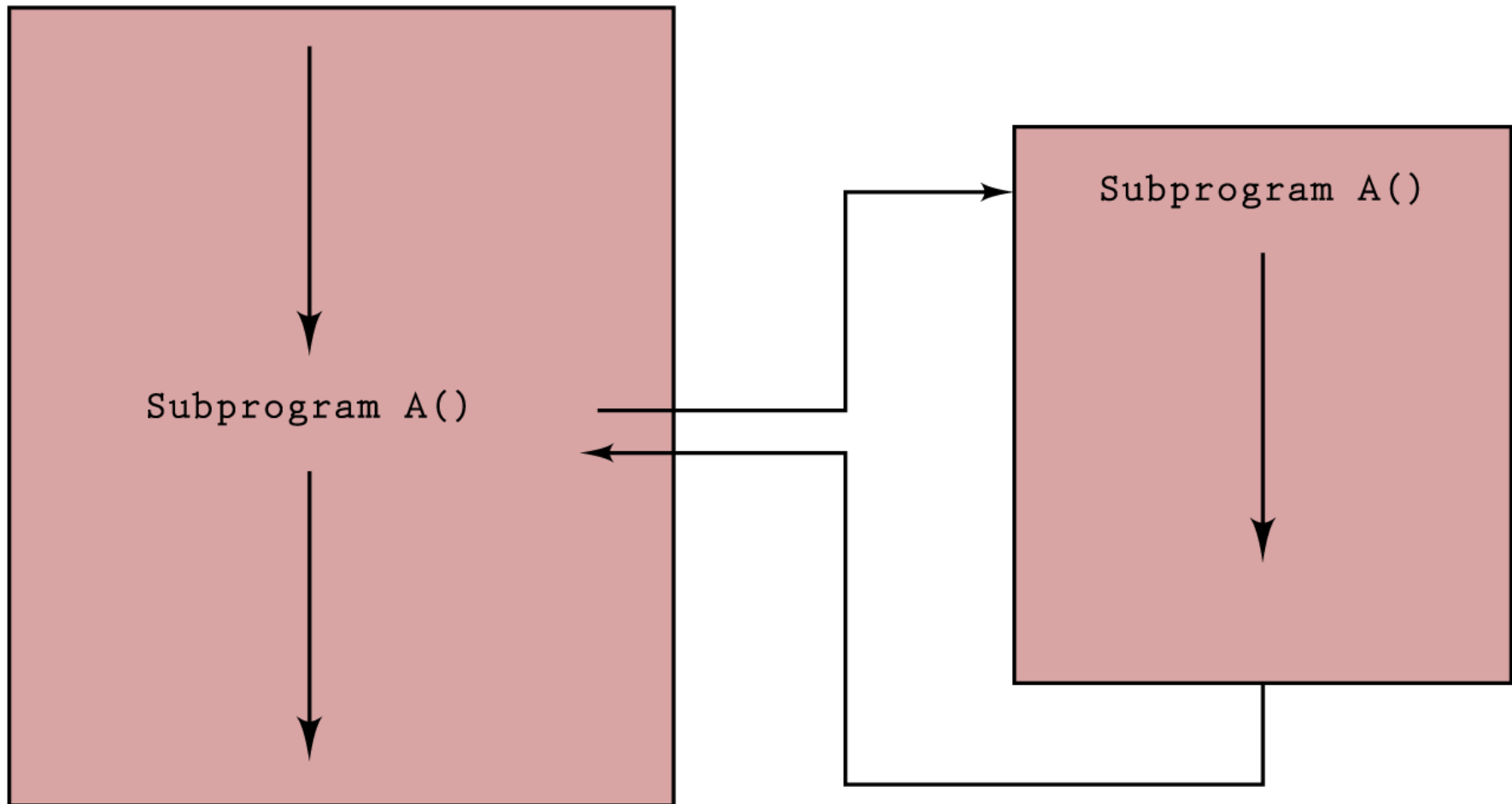(a) Subprogram A does its task and calling unit continues with next statement

Subprogram A()

Subprogram A()

Figure 8.5 **Subprogram flow of control**

Subprogram Statements

(b) Subprogram B does its task and returns a value that is added to 5 and stored in x

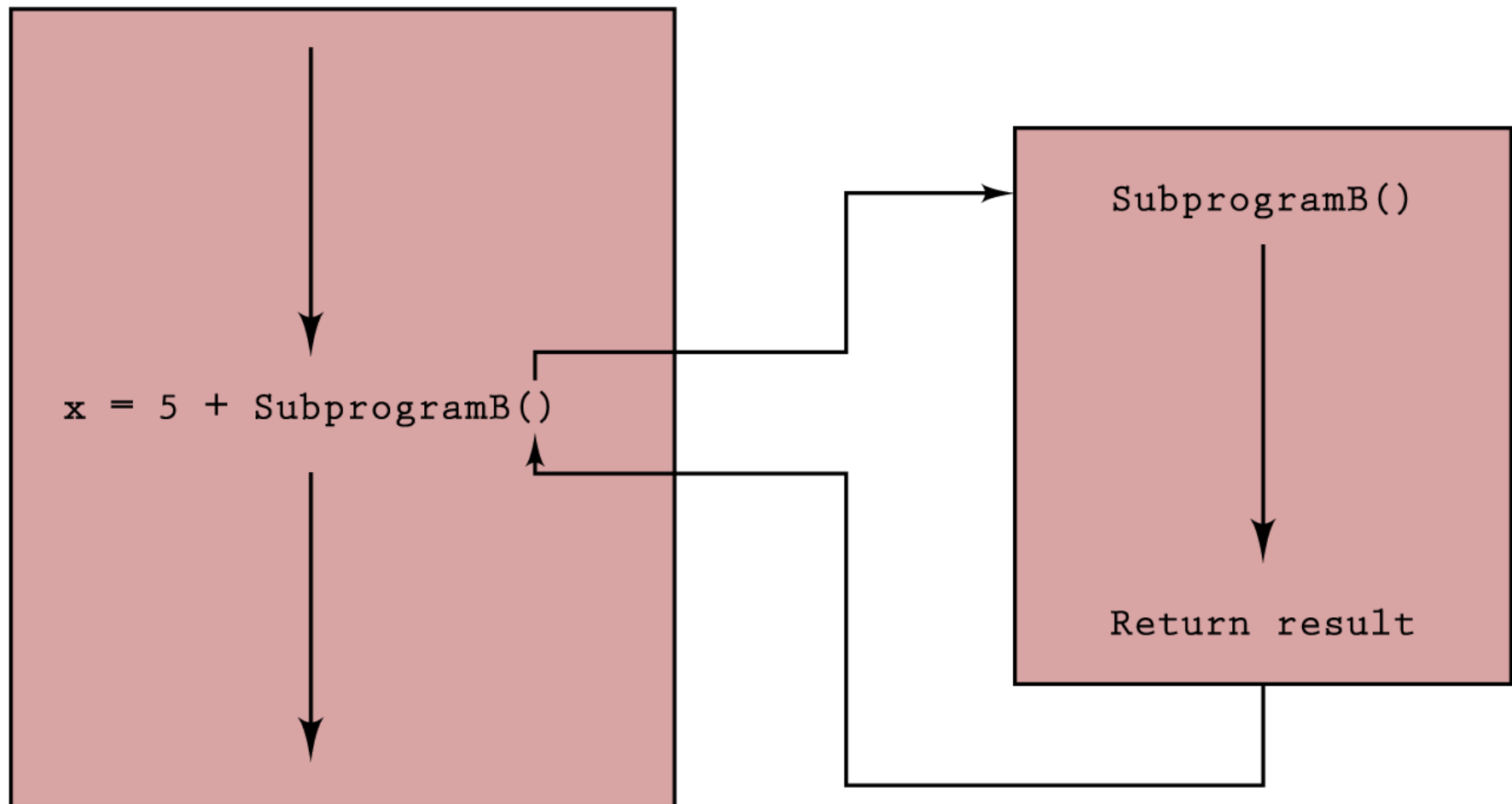x = 5 + SubprogramB()

SubprogramB()

Return result

Figure 8.5 **Subprogram flow of control**

Subprogram Statements

- **Parameters** Identifiers listed in parentheses beside the subprogram declaration; sometimes they are called **formal parameters**

- **Arguments**   Identifiers listed in parentheses on the subprogram call; sometimes they are called **actual parameters**

Subprogram Statements

- **Value parameter** A parameter that expects a **copy** of its argument to be passed by the calling unit (put on the message board)

- **Reference parameter** A parameter that expects the **address** of its argument to be passed by the calling unit (put on the message board)

Hofstra University, CSC005

Subprogram Statements

Language	Subprogram Declaration

VB.NET	`Public Sub Example(ByVal one As Integer,`
	`    ByVal two As Integer,`
	`    ByRef three As Single)`
	
	`    ...`
	`End Sub`

C++/Java	`void Example(int one; int two; float& three)`
	`{`
	`    ...`
	`}`

Page 253

Functions

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!")
}
</script>
</head>

<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

Recursion

- **Recursion** The ability of a subprogram to call itself

- Each recursive solution has at least two cases
  - **Base case**   The case to which we have an answer
  - **General case** The case that expresses the solution in terms of a call to itself with a smaller version of the problem

- For example, the factorial of a number is defined as the number times the product of all the numbers between itself and 0:

$$N! = N * (N - 1)!$$

Asynchronous Processing

- **Asynchronous processing** The concept that input and output can be accomplished through windows on the screen

  - *Clicking* has become a major form of input to the computer

  - Mouse clicking is not within the sequence of the program

  - A user can click a mouse at any time during the execution of a program

  - This type of processing is called **asynchronous**

Composite Data Types

- ## **Records**
  - A record is a named *heterogeneous* collection of items in which individual items are accessed by name
  - The elements in the collection can be of various types

Composite Data Types

Language	Record Type Declaration

Ada	```
	type Name_String is String (1..10);
	type Employee_Type is
	   record
	      Name : Name_String;
	      Age : Integer range 0..100;
	      Hourly_Wage : Float range 1.0..5000.0;
	end record;
	``

VB.NET	```
	Structure Employee
	   Dim Name As String
	   Dim Age As Integer
	   Dim HourlyWage As Single
	End Structure
	``

C++	```
	struct EmployeeType
	{
	   string name;
	   int age;
	   float hourlyWage;
	};
	``

Composite Data Types

Language	Record Variable Declaration and Usage
| Ada | ```
An_Employee : Employee_Type;
...
   An_Employee.Name = "Sarah Gale";
   An_Employee.Age = 32;
   An_Employee.Hourly_Wage = 95.00;
``` |
| VB.NET | ```
Dim AnEmployee As EmployeeType
...
   AnEmployee.Name = "Sarah Gale"
   AnEmployee.Age = 32
   AnEmployee.HourlyWage   95.00
``` |
| C++ | ```
EmployeeType anEmployee;
...
   anEmployee.name = "Sarah Gale";
   anEmployee.age = 32;
   anEmployee.hourlyWage = 95.00;
``` |

Arrays

- An **array** is a named collection of homogeneous items in which individual items are accessed by their place within the collection
  - The place within the collection is called an *index*

Language	Array Declaration
Ada	`type Index_Range is range 1..10;` `type Ten_Things is array (Index_Range) of Integer;`
VB.NET	`Dim TenThings(10) As Integer`
C++/Java	`int tenThings[10];`

Arrays

[0]	1066
[1]	1492
[2]	1668
[3]	1945
[4]	1972
[5]	1510
[6]	999
[7]	1001
[8]	21
[9]	2001

Figure 8.8
Array variable
tenThings
accessed from
0..9

Functionality of Object-Oriented Languages

- **Encapsulation**

- **Inheritance**

- **Polymorphism**

Encapsulation

- **Encapsulation** A language feature that enforces information hiding

- **Class** A language construct that is a pattern for an object and provides a mechanism for encapsulating the properties and actions of the object class

- **Instantiate** Create an object from a class

Inheritance

- **Inheritance**   A construct that fosters reuse by allowing an application to take an already-tested class and derive a class from it that inherits the properties the application needs

- **Polymorphism** The ability of a language to have duplicate method names in an inheritance hierarchy and to apply the method that is appropriate for the object to which the method is applied

Inheritance

Problem space of objects

(a) Problem-solving phase

Abstract to classes (descriptions of objects)

Oval
Circle
Square

Class type definitions

Program space of objects

(b) Implementation phase

- Inheritance and polymorphism combined allow the programmer to build useful hierarchies of classes that can be reused in different applications

Figure 8.9
Mapping of problem into solution

Homework

- **Read Chapter Eight, Sections 8.3 – 8.4**

- **"PLAY" with JavaScript**
  http://www.w3schools.com/js/js_howto.asp

- **Do some of the hands-on examples in class**

- **Program Assignment (#2) Next Class**

Mid-Term

- Good Results
- Some incompletes
- Will review next class

Have A Great Weekend

Hofstra University, CSC005