Abstraction

Abstraction *A mental model that removes complex details*

This is a key concept. Abstraction will reappear throughout the text – be sure you understand it!

Internal and Abstract View

FIGURE 1.2 A car engine and the abstraction that allows us to use it

Definition

Abstraction is the act of representing essential features without including the background details or explanations. In the computer science and software engineering domain, the abstraction principle is used to reduce complexity and allow efficient design and implementation of complex software systems.

Problem Solving

Problem solving

The act of finding a solution to a perplexing, distressing, vexing, or unsettled question

*How do **you** define problem solving?*

Problem Solving

How do you solve problems?

Understand the problem

Devise a plan

Carry out the plan

Look back

Strategies

Ask questions!

- *What do I know about the problem?*
- *What is the information that I have to process in order the find the solution?*
- *What does the solution look like?*
- *What sort of special cases exist?*
- *How will I recognize that I have found the solution?*

Strategies

Ask questions! Never reinvent the wheel!

Similar problems come up again and again in different guises

A good programmer recognizes a task or subtask that has been solved before and plugs in the solution

Can you think of two similar problems?

Strategies

Divide and Conquer!

Break up a large problem into smaller units and solve each smaller problem

- Applies the concept of abstraction
- The divide-and-conquer approach can be applied over and over again until each subtask is manageable

Computer Problem-Solving

- Analysis and Specification Phase
  - Analyze
  - Specification
- Algorithm Development Phase
  - Develop algorithm
  - Test algorithm
- Implementation Phase
  - Code algorithm
  - Test algorithm
- Maintenance Phase
  - Use
  - Maintain

Can you name a recurring theme?

Phase Interactions

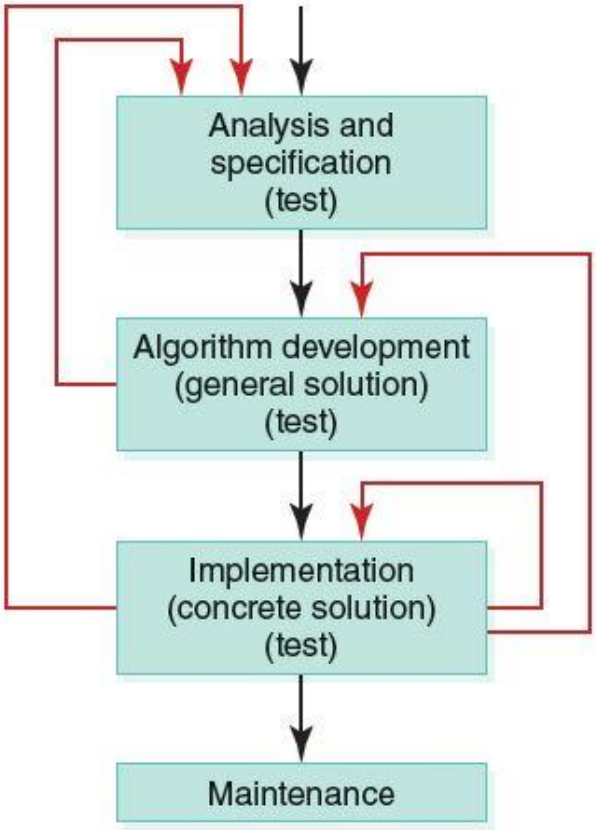


FIGURE 7.3 The interactions among the four problem-solving phases

Should we add another arrow?

(What happens if the problem is revised?)

Algorithms

Algorithm

A set of unambiguous instructions for solving a problem or subproblem in a finite amount of time using a finite amount of *data*

Abstract Step

An algorithmic step containing unspecified details

Concrete Step

An algorithm step in which all details are specified

Developing an Algorithm

Two methodologies used to develop computer solutions to a problem

- Top-down design focuses on the **tasks** to be done
- Object-oriented design focuses on the **data** involved in the solution

Summary of Methodology

Analyze the Problem

- Understand the problem!!
- Develop a plan of attack

List the Main Tasks

- Restate problem as a list of tasks
- Give each task a name

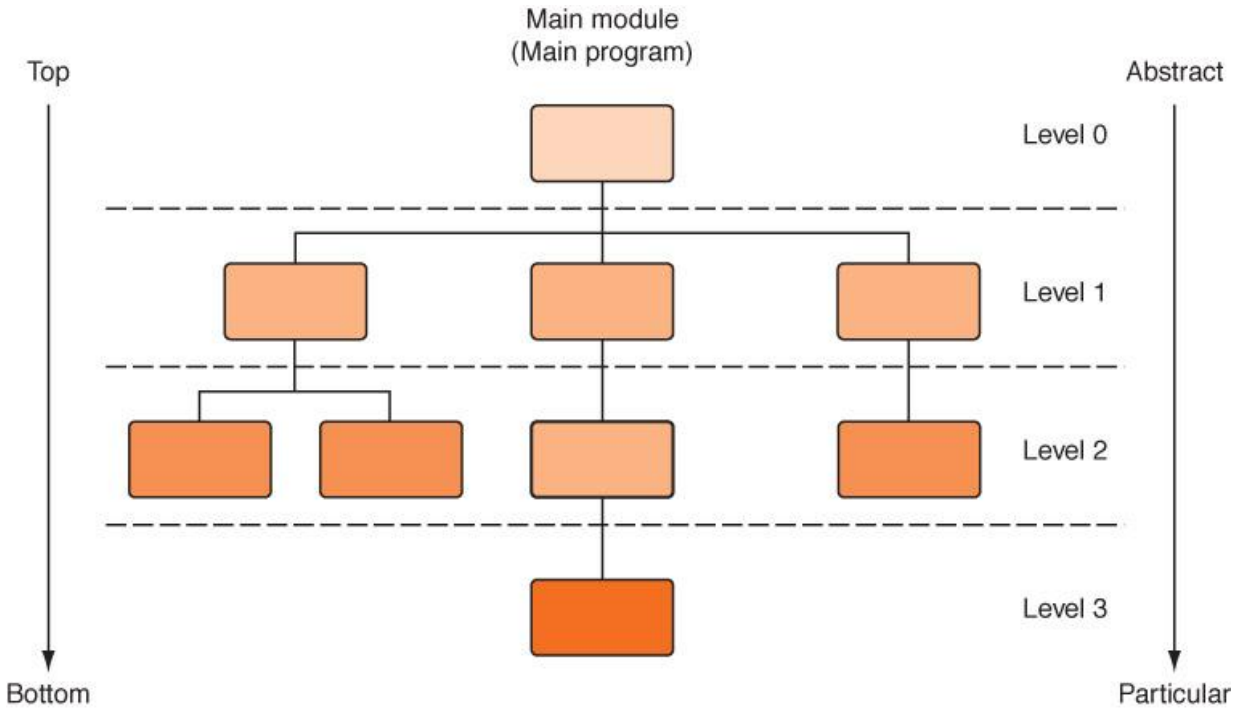
Write the Remaining Modules

- Restate each abstract module as a list of tasks
- Give each task a name

Re-sequence and Revise as Necessary

- Process ends when all steps (modules) are concrete

Top-Down Design



Process continues for as many levels as it takes to make every step concrete

Name of (sub)problem at one level becomes a module at next lower level

Pseudocode

- When we write programs, we assume that the computer executes the program starting at the beginning and working its way to the end.
- This is a basic assumption of all algorithm design.
- We call this SEQUENCE.

Pseudocode

- In Pseudo code it looks like this:

```
Statement1;
Statement2;
Statement3;
Statement4;
Statement5;
Statement6;
Statement7;
Statement8;
```

Pseudocode

- For example, for making a cup of tea:

```
Organise everything together;
Plug in kettle;
Put teabag in cup;
Put water into kettle;
Wait for kettle to boil;
Add water to cup;
Remove teabag with spoon/fork;
Add milk and/or sugar;
Serve;
```

Pseudocode

- Or as a program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 Put water into kettle;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk and/or sugar;
 Serve;
END.
```

Pseudocode

Pseudocode

A way of expressing algorithms that uses a mixture of *English phrases* and *indentation* to make the steps in the solution explicit

There are no grammar rules in pseudocode, but it's important to be consistent and unambigous

Pseudocode Functionality

Variables

Names of places to store values

quotient, decimalNumber, newBase

Assignment

Storing the value of an expression into a variable

Set quotient to 64

quotient <-- 64

*quotient <-- 6 * 10 + 4*

Pseudocode Functionality

Output

Printing a value on an output device

Write, Print

Input

Getting values from the outside word and storing them into variables

Get, Read

Pseudocode

- What if we want to make a choice, for example, do we want to add sugar or not to the tea?

Pseudocode

- What if we want to make a choice, for example, do we want to add sugar or not to the tea?
- We call this SELECTION.

Pseudocode

- So, we could state this as:

```
IF (sugar is required)
    THEN add sugar;
    ELSE don't add sugar;
ENDIF;
```

Pseudocode

- Or, in general:

```
IF (<CONDITION>)
    THEN <Statements>;
    ELSE <Statements>;
ENDIF;
```

Pseudocode

- Or to check which number is biggest:

```
IF (A > B)
    THEN Print A + "is bigger";
    ELSE Print B + "is bigger";
ENDIF;
```

Pseudocode

- Adding a selection statement in the program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 Put water into kettle;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

Pseudocode

- Adding a selection statement in the program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 Put water into kettle;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

Pseudocode Functionality

Selection

Making a choice to execute or skip a statement (or group of statements)

```
Read number
IF (number < 0)
    Write number + " is less than zero."
```

or

```
Write "Enter a positive number."
Read number
IF(number < 0)
    Write number + " is less than zero."
    Write "You didn't follow instructions."
```

Pseudocode Functionality

Selection

Choose to execute one statement (or group of statements) or another statement (or group of statements)

```
IF ( age < 12 )
    Write "Pay children's rate"
    Write "You get a free box of popcorn"
ELSE IF ( age < 65 )
    Write "Pay regular rate"
ELSE
    Write "Pay senior citizens rate"
```

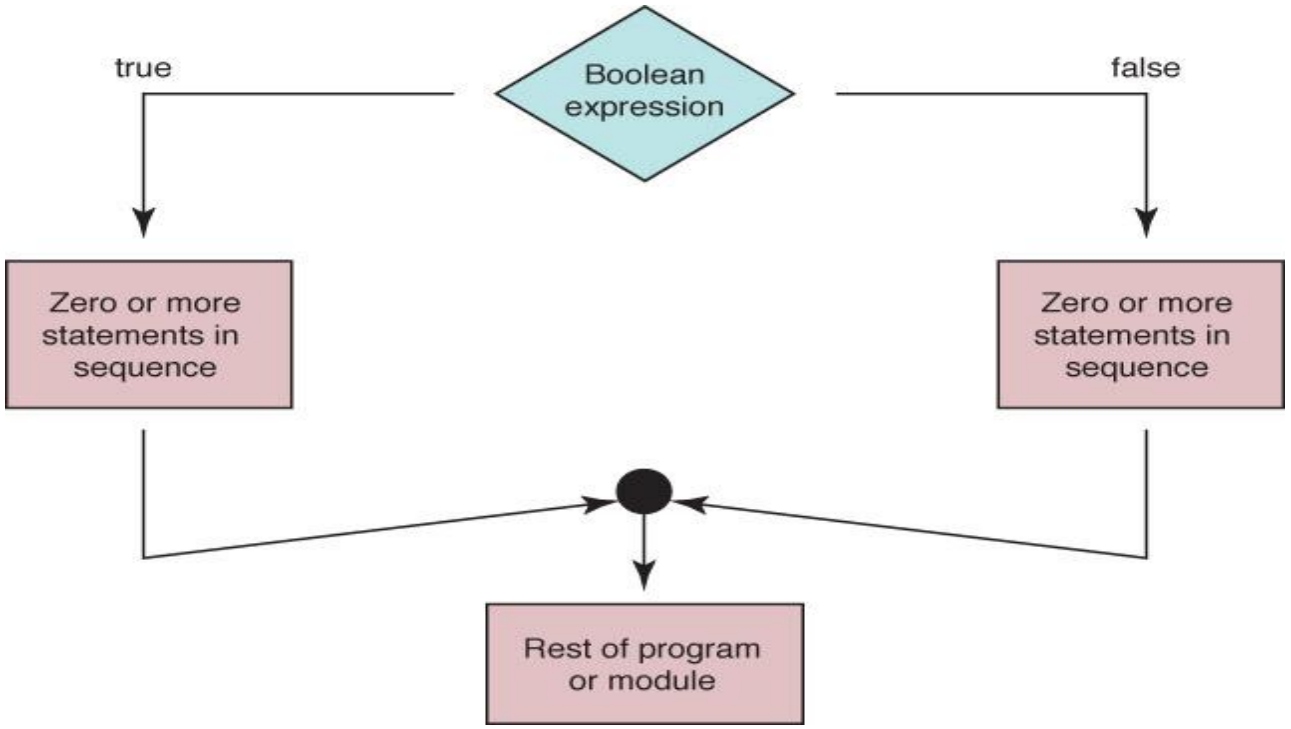
Control Structures

Control structure

An instruction that determines the order in which other instructions in a program are executed

Can you name the ones we defined in the functionality of pseudocode?

Selection Statements



Flow of control of if statement

Algorithm with Selection

Problem: Write the appropriate dress for a given temperature.

Write "Enter temperature"
Read temperature
Determine Dress

Which statements are concrete?
Which statements are abstract?

Algorithm with Selection

Determine Dress

IF (temperature > 90)

Write "Texas weather: wear shorts"

ELSE IF (temperature > 70)

Write "Ideal weather: short sleeves are fine"

ELSE IF (temperature > 50)

Write "A little chilly: wear a light jacket"

ELSE IF (temperature > 32)

Write "Philadelphia weather: wear a heavy coat"

ELSE

Write "Stay inside"

Pseudocode Functionality

Repetition Or Iteration

Repeating a series of statements

```
Set count to 1
WHILE ( count < 10)
    Write "Enter an integer number"
    Read aNumber
    Write "You entered " + aNumber
    Set count to count + 1
```

How many values were read?

Pseudocode Example

Problem: Read in pairs of positive numbers and print each pair in order.

WHILE (not done)
    Write "Enter two values separated by blanks"
    Read number1
    Read number2
Print them in order

Pseudocode Example

How do we know when to stop?

Let the user tell us how many

Print them in order?

If first number is smaller

print first, then second

Otherwise

print second, then first

Pseudocode Example

```
Write "How many pairs of values are to be entered?"
Read numberOfPairs
Set numberRead to 0
WHILE (numberRead < numberOfPairs)
        Write "Enter two values separated by a blank; press return"
        Read number1
        Read number2
        IF(number1 < number2)
            Print number1 + " " + number2
        ELSE
            Print number2 + " " number1
        Increment numberRead
```

Pseudocode

- What if we need to tell the computer to keep doing something until some condition occurs?

Pseudocode

- What if we need to tell the computer to keep doing something until some condition occurs?
- Let's say we wish to indicate that the you need to keep filling the kettle with water until it is full.

Pseudocode

- What if we need to tell the computer to keep doing something until some condition occurs?
- Let's say we wish to indicate that the you need to keep filling the kettle with water until it is full.
- We need a loop, or ITERATION.

Pseudocode

- So, we could state this as:

```
WHILE (Kettle is not full)
    DO keep filling kettle;
ENDWHILE;
```

Pseudocode

- Or, in general:

```
WHILE (<CONDITION>)
    DO <Statements>;
ENDWHILE;
```

Pseudocode

- Or to print out the numbers 1 to 5:

```
A = 1;
WHILE(A < 5)
    DO Print A;
        A = A + 1;
ENDWHILE;
```

Pseudocode

- What is the benefit of using a loop?

Pseudocode

- Consider the problem of searching for an entry in a phone book with only condition:

Pseudocode

- Consider the problem of searching for an entry in a phone book with only condition:

```
Get first entry
If this is the required entry
  Then write down phone number
  Else get next entry
  If this is the correct entry
   then write done entry
   else get next entry
          if this is the correct entry
                  …….
```

Pseudocode

- This could take forever to specify.

Pseudocode

- This could take forever to specify.
- There must be a better way to do it.

Pseudocode

- We may rewrite this as follows:

```
Get first entry;
Call this entry N;
WHILE N is NOT the required entry
DO Get next entry;
     Call this entry N;
ENDWHILE;
```

Pseudocode

- We may rewrite this as follows:

```
Get first entry;
Call this entry N;
WHILE N is NOT the required entry
DO Get next entry;
     Call this entry N;
ENDWHILE;
```

- This is why we love loops!

Pseudocode

- Or as a program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 WHILE (Kettle is not full)
   DO keep filling kettle;
 ENDWHILE;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

Pseudocode

- Or as a program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 WHILE (Kettle is not full)
   DO keep filling kettle;
 ENDWHILE;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

Logical operators

- AND
  - True only if both the conditions are correct
- OR
  - True if one of the conditions is true
- NOT
  - True of the condition is False

What is the output of following:

```
x = 10

y  = 5

z = 25

if(x>y AND x>z)

        print “x is the largest”

else if (y > x AND y > z)

        print “y is the largest”

else

        print “z is the largest”
```

The output depends on values of x, y and z.

For above examples, the output is - z is the largest.

Write the output

```
x = 10

y = 40

if (x < 10 OR y>40)

        print x

else if (x < 5 OR y> 100)

        print y

else if(x>5 OR y>100)

        print x

        print y
```

The output should

10

40

Find the output

```
x = 50
if( NOT (x ==50))
        print “x is not equal to 50”
else
        print “x is equal to 50”
```

Output:

```
x is equal to 50
```

Find the output

```
number = 5
while (number < 51)
    print number
    number = number + 5
```

Output

5

10

15

20

25

30

35

40

45

50

Find Output

```
sum = 0

number = 0

while (number < 5)

    sum = sum + number

    number = number + 1

print sum
```

EXAMPLES

Pseudocode

- So let's say we want to express the following algorithm:
  - *Read in a number and print it out.*

Pseudocode

```
PROGRAM PrintNumber:
    Read A;
    Print A;
END.
```

Pseudocode

- So let's say we want to express the following algorithm:
  - *Read in a number and print it out double the number.*

Pseudocode

```
PROGRAM PrintDoubleNumber:
    Read A;
    B = A*2;
    Print B;
END.
```

Pseudocode

- So let's say we want to express the following algorithm:
  - *Read in a number, check if it is odd or even.*

Pseudocode

```
PROGRAM IsOddOrEven:
     Read A;
     IF (A/2 gives a remainder)
               THEN Print "It's Odd";
               ELSE Print "It's Even";
     ENDIF;
END.
```

Pseudocode

- So let's say we want to express the following algorithm to print out the bigger of two numbers:
  - *Read in two numbers, call them A and B. Is A is bigger than B, print out A, otherwise print out B.*

Pseudocode

```
PROGRAM PrintBiggerOfTwo:
    Read A;
    Read B;
    IF (A>B)
            THEN Print A;
            ELSE Print B;
    ENDIF;
END.
```

Pseudocode

- So let's say we want to express the following algorithm to print out the bigger of three numbers:
  - *Read in three numbers, call them A, B and C.*
    - *If A is bigger than B, then if A is bigger than C, print out A, otherwise print out C.*
    - *If B is bigger than A, then if B is bigger than C, print out B, otherwise print out C.*

Pseudocode

```
PROGRAM BiggerOfThree:
    Read A;
    Read B;
    Read C;
    IF (A>B)
            THEN IF (A>C)
                            THEN Print A;
                            ELSE Print C;
                       END IF;
            ELSE IF (B>C)
                            THEN Print B;
                            ELSE Print C;
                     END IF;
    END IF;
END.
```

Pseudocode

- So let's say we want to express the following algorithm:
  - *Print out the numbers from 1 to 5*

Pseudocode

```
PROGRAM Print1to5:
    A = 1;
    WHILE (A != 6)
    DO  Print A;
        A = A + 1;
    ENDWHILE;
END.
```

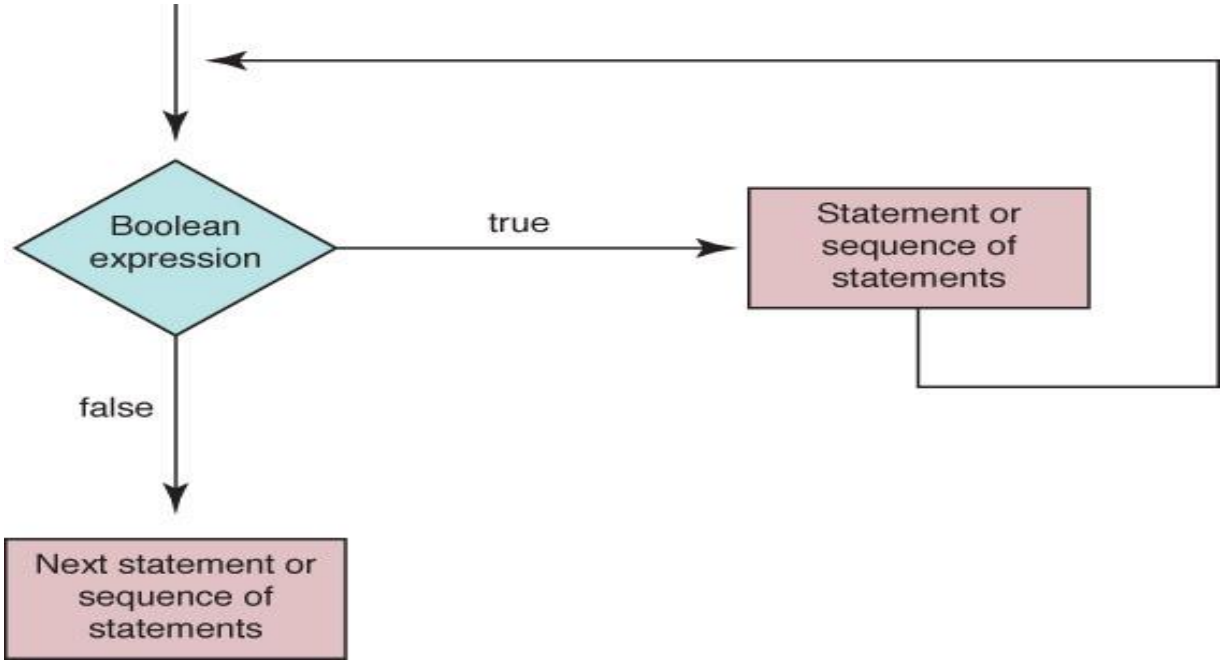
Pseudocode

- So let's say we want to express the following algorithm:
  - *Add up the numbers 1 to 5 and print out the result*

Pseudocode

```
PROGRAM PrintSum1to5:
    Total = 0;
    A = 1;
    WHILE (A != 6)
    DO  Total = Total + A;
        A = A + 1;
    ENDWHILE;
    Print Total;
END.
```

Looping Statements



Flow of control of while statement

Looping Statements

A count-controlled loop

Set sum to 0
Set count to 1
While (count <= limit)
    Read number
    Set sum to sum + number
    Increment count
Write "Sum is " + sum

Why is it called a count-controlled loop?

Looping Statements

An event-controlled loop

```
Set sum to 0
Set allPositive to true
WHILE (allPositive)
    Read number
    IF (number > 0)
        Set sum to sum + number
    ELSE
        Set allPositive to false
Write "Sum is " + sum
```

Why is it called an event-controlled loop?
What is the event?

Looping Statements

Calculate Square Root

Read in square

Calculate the square root

Write out square and the square root

Are there any abstract steps?

Looping Statements

Calculate Square Root

Set epsilon to 1
WHILE (epsilon > 0.001)
    Calculate new guess
    *Set epsilon to abs(square - guess * guess)*

Are there any abstract steps?

Looping Statements

Calculate New Guess

Set newGuess to

(guess + (square/guess)) / 2.0

Are there any abstract steps?

Looping Statements

Read in square
Set guess to 0.1

Set epsilon to 1

WHILE (epsilon > 0.001)

    guess = (guess + (square/guess))/2.0

    *Set epsilon to abs(square - guess * guess)*

Write out square and the guess

Pseudocode for Complete Computer Solution

Write "Enter the new base"

Read newBase

Write "Enter the number to be converted"

Read decimalNumber

Set quotient to 1

WHILE (quotient is not zero)

    Set quotient to decimalNumber DIV newBase

    Set remainder to decimalNumber REM newBase

    Make the remainder the next digit to the left in the answer

    Set decimalNumber to quotient

Write "The answer is "

Write answer

Pseudocode for Complete Computer Solution

Write "Enter the new base"

Read newBase

Write "Enter the number to be converted"

Read decimalNumber

Set quotient to 1

WHILE (quotient is not zero)

    Set quotient to decimalNumber DIV newBase

    Set remainder to decimalNumber REM newBase

    Make the remainder the next digit to the left in the answer

    Set decimalNumber to quotient

Write "The answer is "

Write answer

Loops

The while loops have three parts:

1. Initialization - fix the initial value
2. Condition - when will we stop? until while shall we continue?
3. Update - update the value of variable so that the condition is changed.

Let’s explain with example.

Q. Write pseudocode to print numbers from 1 to 100

1. Initialization - where to start? We start from 1

   set number = 1 (initial value)

2. Condition - until while to continue.

   continue until number becomes 101

   means while (number < 101)

3. Update - But the value of number is 1. So we need to increase the value and print it. We increase by 1.

number = number + 1

The complete pseudocode

```
num = 1                          (Initialization)
while (num < 101)                (Condition)
    print num
    num = num + 1                (Update)
```

Q. Pseudocode to calculate the sum of numbers from 1 to n where n can be entered by user.

sum = 1+2+3+...+n

Ans:

Read n from user

number = 1 **Initialization**

sum = 0

```
while (number <=  n)                    Condition
    sum = sum + number
    print “Number:” + number
    print “Sum:” + sum
    number = number + 1                 Update
print “Final Sum:” + sum
```

Output: Let’s suppose the user entered n to be 6

Iteration	Number	Sum
1	1	1
2	2	3
3	3	6
4	4	10
5	5	15
6	6	21

Iteration means the number of times the loop is run. For this loop the total number of iterations will be n i.e. 6.

Number: 1

Sum: 1

Number: 2

Sum: 3

Number: 3

Sum: 6

Number: 4

Sum: 10

Number: 5

Sum: 15

Number: 6

Sum: 21

Final Sum: 21

We can actually print the final sum only as that's what we need.

Q. Pseudocode to calculate the sum of squares from 1 to 5.

$$sum = 1^2+2^2+3^2+4^2+5^2$$

Ans:

Instead of adding the number we just need to add the sum of number

```
sum = 0

number = 1

while (number < 6)

    sum = sum +number*number
```

```
        number = number + 1

print “Final sum:” + sum
```

What should be the final sum?

Q. Pseudocode to print the multiplication table of 5. You just need to print the values like 5, 10, … , 50

Ans:

```
num = 5
while (num < 51)
    print num
    num = num + 5
```

Can you figure out the output?

Q. Pseudocode to convert decimal to binary

```
n = 2
read decimalNumber
while (decimalNumber > 0)
    quotient = decimalNumber / n
    remainder = decimalNumber % 2
    print remainder
    move remainder to the left of answer
    decimalNumber = quotient
```

Remainder is binary digit.
Check Chapter 1 if confused.

Q. Extract digits from a decimal number

```
Read number
while (number > 0)
    remainder = number % 10
    quotient = number / 10
    number = quotient
    print remainder
```

% calculates the remainder
The digits here are remainders

How to swap two numbers in variables?

```
a = 20

b = 30

c = a

a = b

b = c

print “a=” + a

print “b=” + b
```

This is similar to changing the glass of juice and beer. To do that, we need one extra glass. We first pour beer in the extra glass, then pour juice in the beer glass and beer from extra glass to the juice glass.

So what should be the value of a & b.

Q. Pseudocode to find the factorial of a number.

factorial (5) = 1x2x3x4x5

limit = 5 (this could be any number)

factorial = 1

num = 1

while (num <= limit)

    factorial = factorial * num

    num = num + 1

Q. Pseudocode to calculate 3^n.

3^n = 3x3x3x….n times

Read n from user (or use n = 10 or any number)

num = 1

power = 1

while(num <= n)

power = power * 3

num = num + 1

Difference between factorial and power.