12. Pseudocode 2

What did we do last time?

No standard pseudocode at present

- In general:
  - Statement are written in simple English.
  - Each instruction is written on a separate line.
  - Keywords and indentation are used to signify particular control structures.
  - Each set of instructions is written from top to bottom, with only one entry and one exit.
  - Groups of statements may be formed into modules, and that module given a name.

Six basic computer operations:

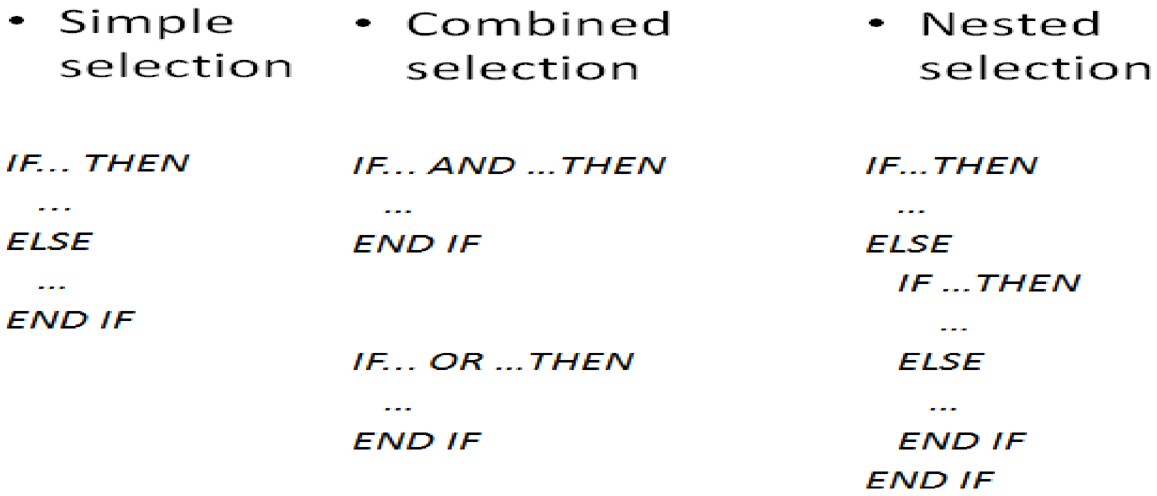
- Operation 1: To receive information, **Read, Get**
- Operation 2: To put out information, **Print, Write, Put, Output, Display; Prompt and Get**
- Operation 3: To perform arithmetic, **+, -, *, /, =, (), Compute, Calculate**
- Operation 4: To assign a value to a variable or memory location, **Initialise, Set, “=”, “<-”, Save, Store**
- Operation 5: To compare two variables and select one of two alternative actions, **IF, THEN, ELSE, END IF**
- Operation 6: To repeat a group of actions, **WHILE…DO…END WHILE**

Use meaningful variable names

- For example, number1, number2 and number3 are more meaningful names for three numbers than A, B and C.
- If more than one word is used in the name of a variable, then underscores are useful as word separators, for example, sales_tax.
- Most programming languages do not tolerate a space in a variable name.

The three basic control structures

1. **Sequence** - The sequence control structure.
2. **Selection** – The selection control structure.



- Simple selection

```
IF... THEN
  ...
ELSE
  ...
END IF
```

- Combined selection

```
IF... AND ...THEN
  ...
END IF

IF... OR ...THEN
  ...
END IF
```

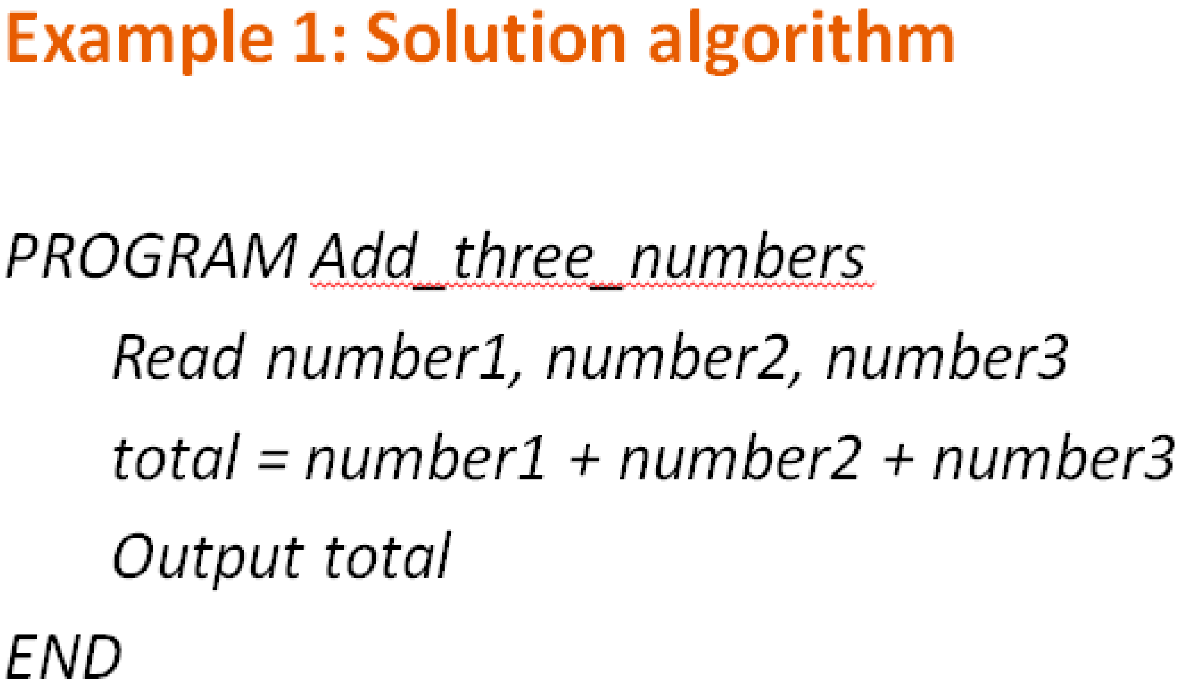
- Nested selection

```
IF...THEN
  ...
ELSE
  IF ...THEN
    ...
  ELSE
    ...
  END IF
END IF
```

3. **Repetition** – The repetition control structure.
   – Using the WHILE…DO…END WHILE structure
   – Using the REPEAT…UNTIL structure
   – Counted loops

Example 1:

- A program is required to read three numbers, add them together and output their total.

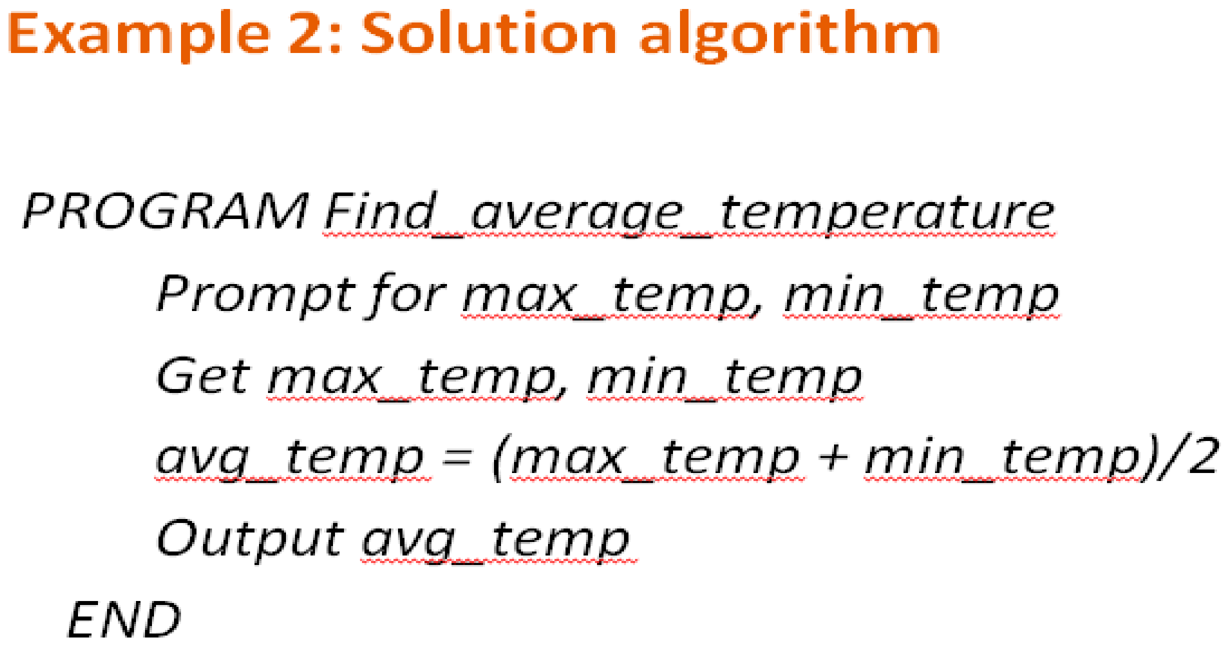


Example 1: Solution algorithm

```
PROGRAM Add_three_numbers
    Read number1, number2, number3
    total = number1 + number2 + number3
    Output total
END
```

Example 2:

- A program is required to prompt the terminal operator for the maximum and minimum temperature readings on a particular day, accept those readings as integers, and calculate and display to the screen the average temperature.



Example 2: Solution algorithm

```
PROGRAM Find_average_temperature
        Prompt for max_temp, min_temp
        Get max_temp, min_temp
        avg_temp = (max_temp + min_temp)/2
        Output avg_temp
   END
```

Example 3:

- A program required to read from the screen the length and width of a rectangular house block, and the length and width of the rectangular house that has been built on the block. The algorithm then compute and display the mowing time required to cut the grass around the house, at the rate of two square metres per minute.

Example 3: Possible solution algorithm

```
PROGRAM Calculate_mowing_time
        Prompt for block_length, block_width
        Get block_length, block_width
        block_area = block_length * block_width

        Prompt for house_length, house_width
        Get house_length, house_width
        house_area = house_length * house_width

        mowing_area = block_area – house_area
        mowing_time = mowing_area / 2
        Output mowing_time
END
```

Example 4: Use nested selection/Case

- A program is required to read a customer’s name, a purchase amount and a tax code. The tax code has been validated and will be one of the following:
  - 0 tax exempt (0%)
  - 1 sales tax (17%)
  - 2 special sales tax (20%)

  The program must then compute the sales tax and the total amount due, and print the customer’s name, purchase amount, sales tax and total amount due.

Example 4: Possible solution algorithm - 1

```
PROGRAM Process_customer_record
    Read cust_name, purch_amt, tax_code
    IF tax_code = 0 THEN
        sales_tax = 0
    ELSE
        IF tax_code = 1 THEN
            sales_tax = purch_amt * 0.17
        ELSE
            sales_tax = purch_amt * 0.2
        END IF
    END IF
    total_amt = purch_amt + sales_tax
    Print cust_name, purch_amt, sales_tax, total_amt
END
```

Example 4: Possible solution algorithm - 2

```
PROGRAM Process_customer_record
    Read cust_name, purch_amt, tax_code
    CASE OF tax_code
        0: sales_tax = 0
        1: sales_tax = purch_amt * 0.17
        2: sales_tax = purch_amt * 0.2
     END CASE
    total_amt = purch_amt + sales_tax
    Print cust_name, purch_amt, sales_tax, total_amt
END
```

More examples of pseudocode

Example 5: Use WHILE…DO…END WHILE

- Every day, a weather station receives 15 temperatures expressed in degree Fahrenheit. A program is to be written that will accept each Fahrenheit temperature, convert it to Celsius and display the converted temperature to the screen. After 15 temperatures have been processed, the words “All temperatures processed” are to be displayed on the screen.
- Hint: *Celsius temperatures*

  *= (Fahrenheit temperature – 32) * 5 / 9*

Example 5: Possible solution algorithm

```
PROGRAM Fahrenheit_Celsius_conversion
    Set temperature_count to zero
    WHILE temperature_count < 15
    DO
        Prompt for f_temp
        Get f_temp
        Compute c_temp = (f_temp – 32) * 5 / 9
        Display c_temp
        add 1 to temperature_count
    END WHILE
    Display “All temperatures processed”
END
```

Example 6: Use Counted loop

- The possible solution algorithm for Example 5 can also be give as:

```
PROGRAM Fahrenheit_Celsius_conversion
    FOR temperature_count = 1 to 15
        Prompt for f_temp
        Get f_temp
        Compute c_temp = (f_temp – 32) * 5 / 9
        Display c_temp
    END FOR
    Display “All temperatures processed”
END
```

Example 7: Use REPEAT…UNTIL

- A program is required to read a series of inventory records that contain an item number, an item description and a stock figure. The last record in the file has an item number of zero. The program is to produce a low stock items report, by printing only those records that have a stock figure of less than 20 items. A heading is to be printed at the top of the report and a total low stock item count printed at the end.

Example 7: Possible solution algorithm

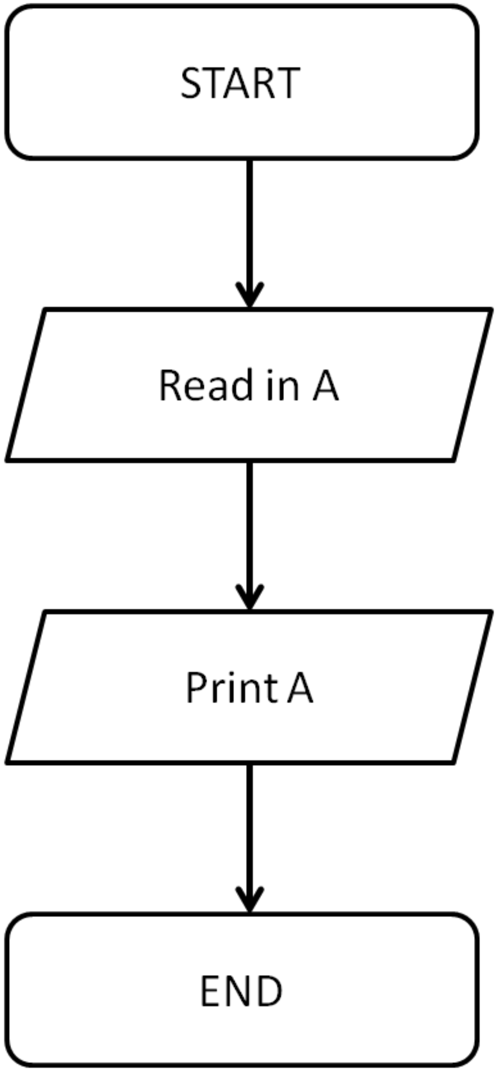
```
PROGRAM Process_inventory_records
    Set total_low_stock_items to zero
    Write "Low Stock Items" heading
    REPEAT
    Read inventory record
    IF item_number >  zero THEN
        IF stock_figure < 20 THEN
            Write item_number, item_description, stock_figure
            increment total_low_stock_items
        END IF
    END IF
    UNTIL item_nubmer = zero
    Write total_low_stock_items
END
```

FLOW CHARTING, STRUCTURED ENGLISH, AND PSEUDOCODE

Flowcharts – read in, print out (Problem 1)

- So let's say we want to express the following algorithm:

Read in a number and print it out.



Structured English and Pseudocode

Structured English

```
PROGRAM PrintNumber
    Read in a number and print
    it out
END
```

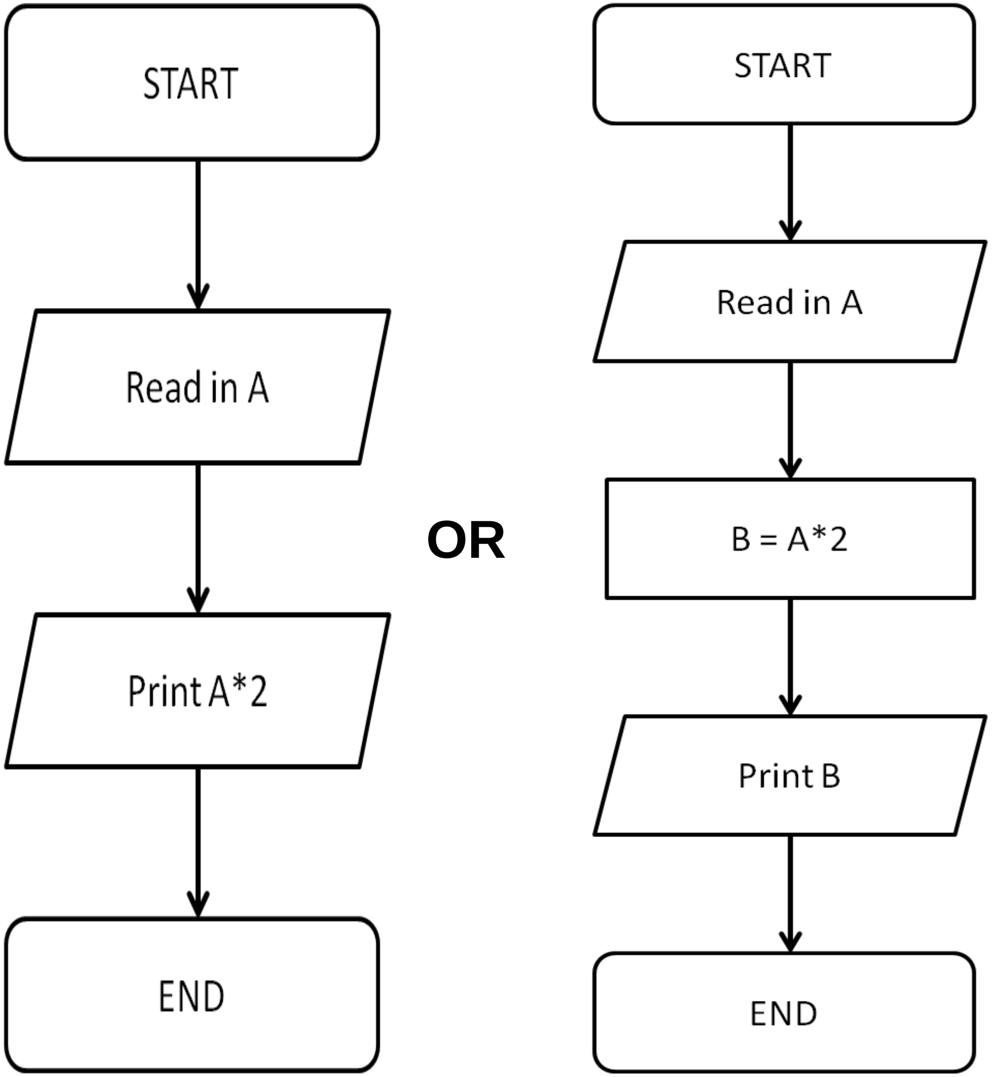
Pseudocode

```
PROGRAM PrintNumber
    Read A
    Print A
END
```

Flowcharts – multiply by 2 (Problem 2)

- So let’s say we want to express the following algorithm:

Read in a number and print it out double the number.



Structured English and Pseudocode

Structured English

```
PROGRAM DoubleNumber
    Read in a number and print
    double the number out
END
```

Pseudocode

```
PROGRAM DoubleNumber
    Read A
    Print A*2
END
```

Or alternatively…

Structured English and Pseudocode

Structured English

```
PROGRAM DoubleNumber
    Read in a number and print
    double the number out
END
```

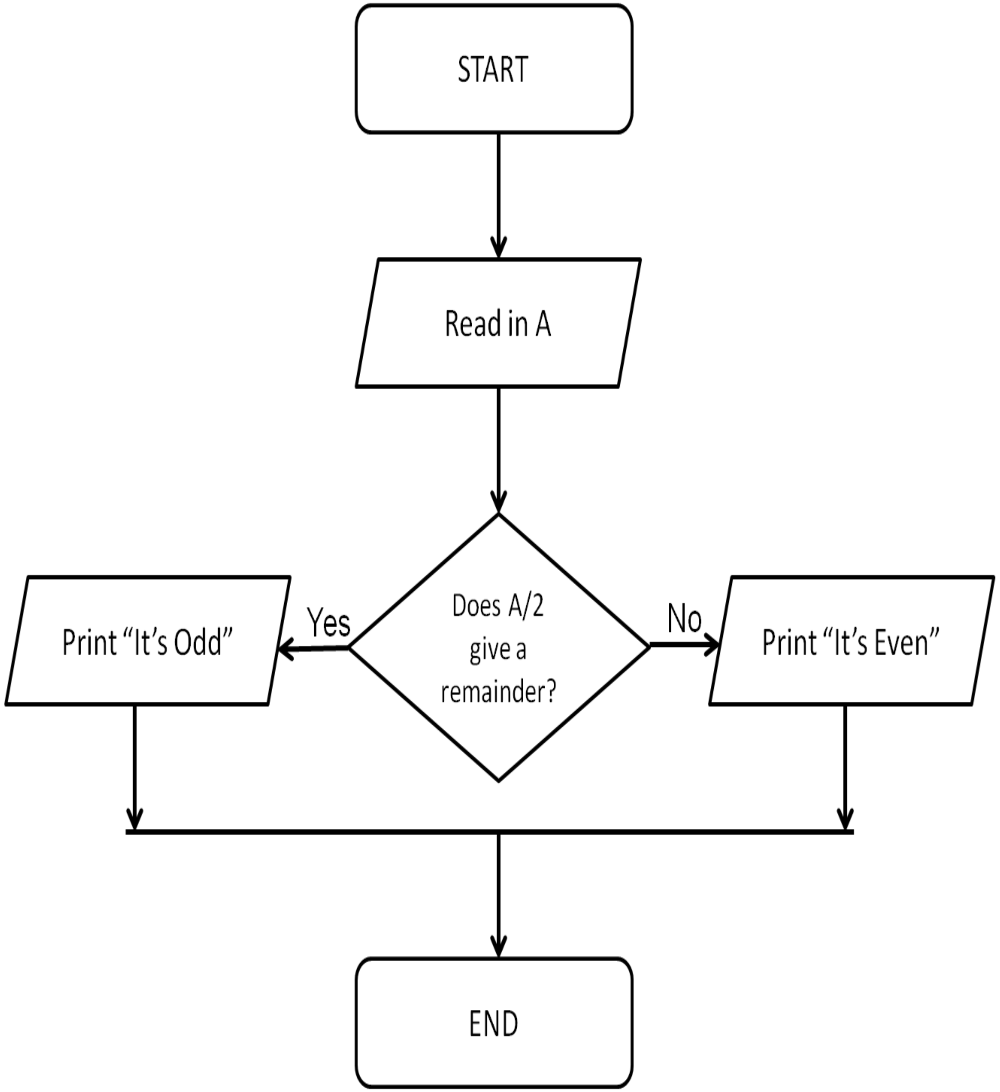
Pseudocode

```
PROGRAM DoubleNumber
    Read A
    B = A*2
    Print B
END
```

Flowcharts – odd or even? (Problem 3)

- So let’s say we want to express the following algorithm:

Read in a number, check if it is odd or even.



Structured English and Pseudocode

Structured English

```
PROGRAM OddOrEven
    Read in a number.
    Divide it by two
    If there is a remainder, the number is odd, otherwise it’s even
END
```

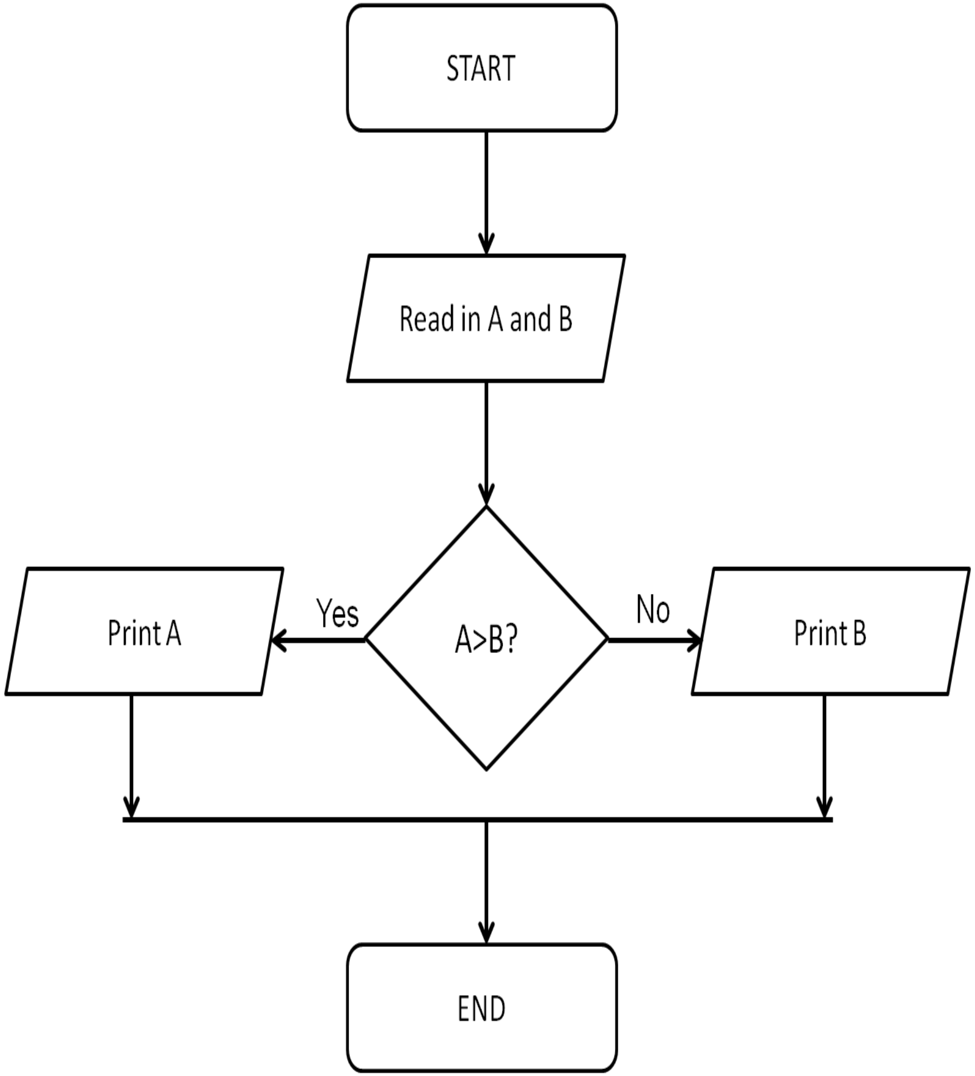
Pseudocode

```
PROGRAM OddOrEven
    Read A
    IF A/2 gives a remainder THEN
        Print “It’s Odd”
    ELSE
        Print “It’s Even”
    ENDIF
END
```

Flowcharts – is A bigger than B? (Problem 4)

- 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. If A is bigger than B, print out A, otherwise print out B.



Structured English and Pseudocode

Structured English

```
PROGRAM BiggerOfTwo
    Read in a number
    Read in a second number
    If the first number is bigger, then print out that number, otherwise print out the other number
END
```

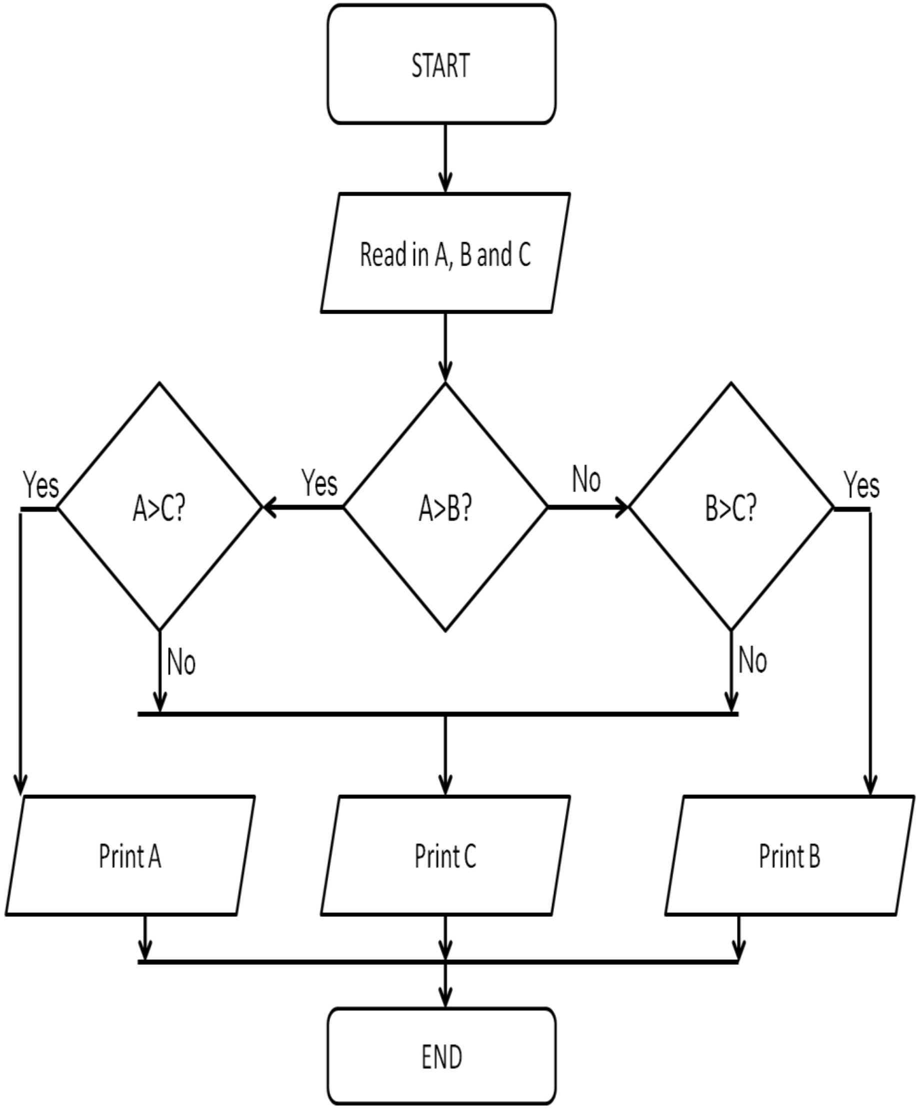
Pseudocode

```
PROGRAM BiggerOfTwo
    Read A
    Read B
    IF A>B THEN
        Print A
     ELSE
        Print B
    ENDIF
END
```

Flowcharts – which is bigger, A,B,C (Problem 7)

- 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.



Structured English and Pseudocode

Structured English

```
PROGRAM BiggerOfThree
    Read in a number
    Read in a second number
    Read in a third number
    If the first number is bigger than the second, then if the first number is bigger than the third, then print out the first number, otherwise print out the third number
    If the first number is smaller than the second, then if the second number is bigger than the third, then print out the second number, otherwise print out the third number
END
```

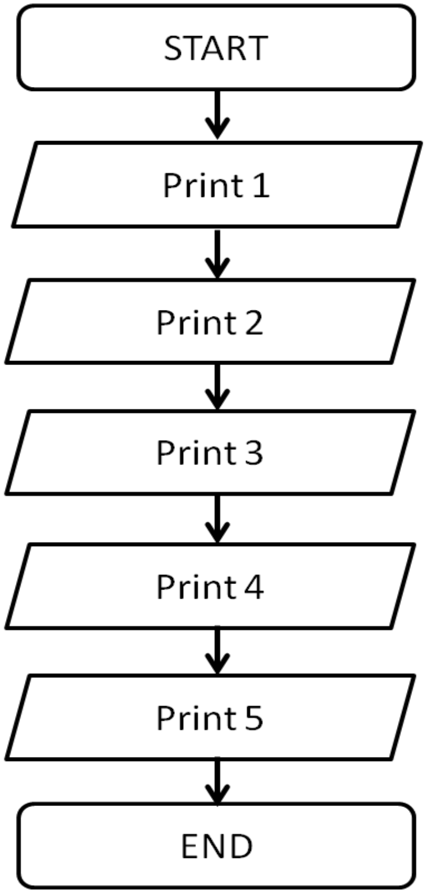
Pseudocode

```
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
```

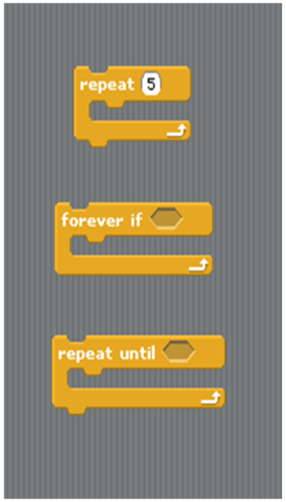
Flowcharts – print out 5 numbers (Problem 8)

- So let’s say we want to express the following algorithm:

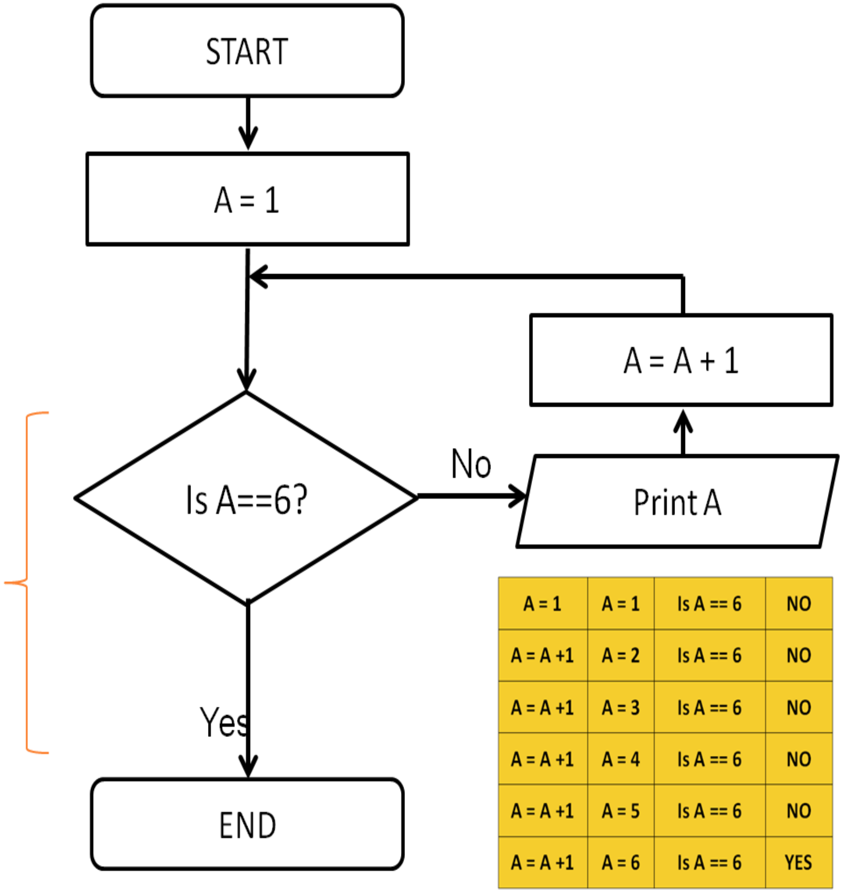
Print out the numbers from 1 to 5



A LOOP



In scratch



A = 1	A = 1	Is A == 6	NO
A = A +1	A = 2	Is A == 6	NO
A = A +1	A = 3	Is A == 6	NO
A = A +1	A = 4	Is A == 6	NO
A = A +1	A = 5	Is A == 6	NO
A = A +1	A = 6	Is A == 6	YES

Structured English and Pseudocode

Structured English

```
PROGRAM Print1to5
    Print out 1
    Print out 2
    Print out 3
    Print out 4
    Print out 5
END
```

Pseudocode

```
PROGRAM Print1to5
    Print 1
    Print 2
    Print 3
    Print 4
    Print 5
END
```

Or alternatively…

Structured English and Pseudocode

Structured English

```
PROGRAM Print1to5
    Keep printing out numbers
    until you reach 5
END
```

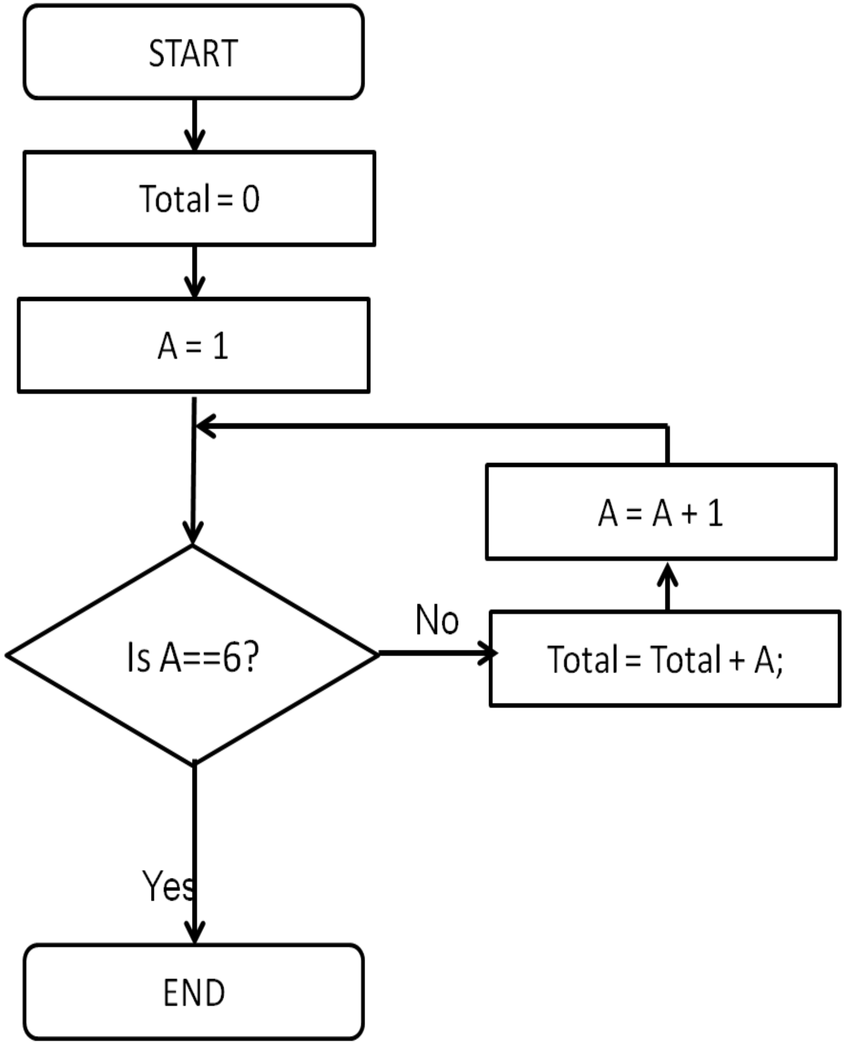
Pseudocode

```
PROGRAM Print1to5
   Set A = 1
   WHILE (A != 6)
   DO
        Print A
        A = A + 1
   END WHILE
END
```

Flowcharts – add up numbers 1 to 5 (Problem 9)

- So let’s say we want to express the following algorithm:

Add up the numbers 1 to 5



Structured English and Pseudocode

Structured English

```
PROGRAM PrintSum1to5
    Keep adding the numbers
    from 1 to 5
END
```

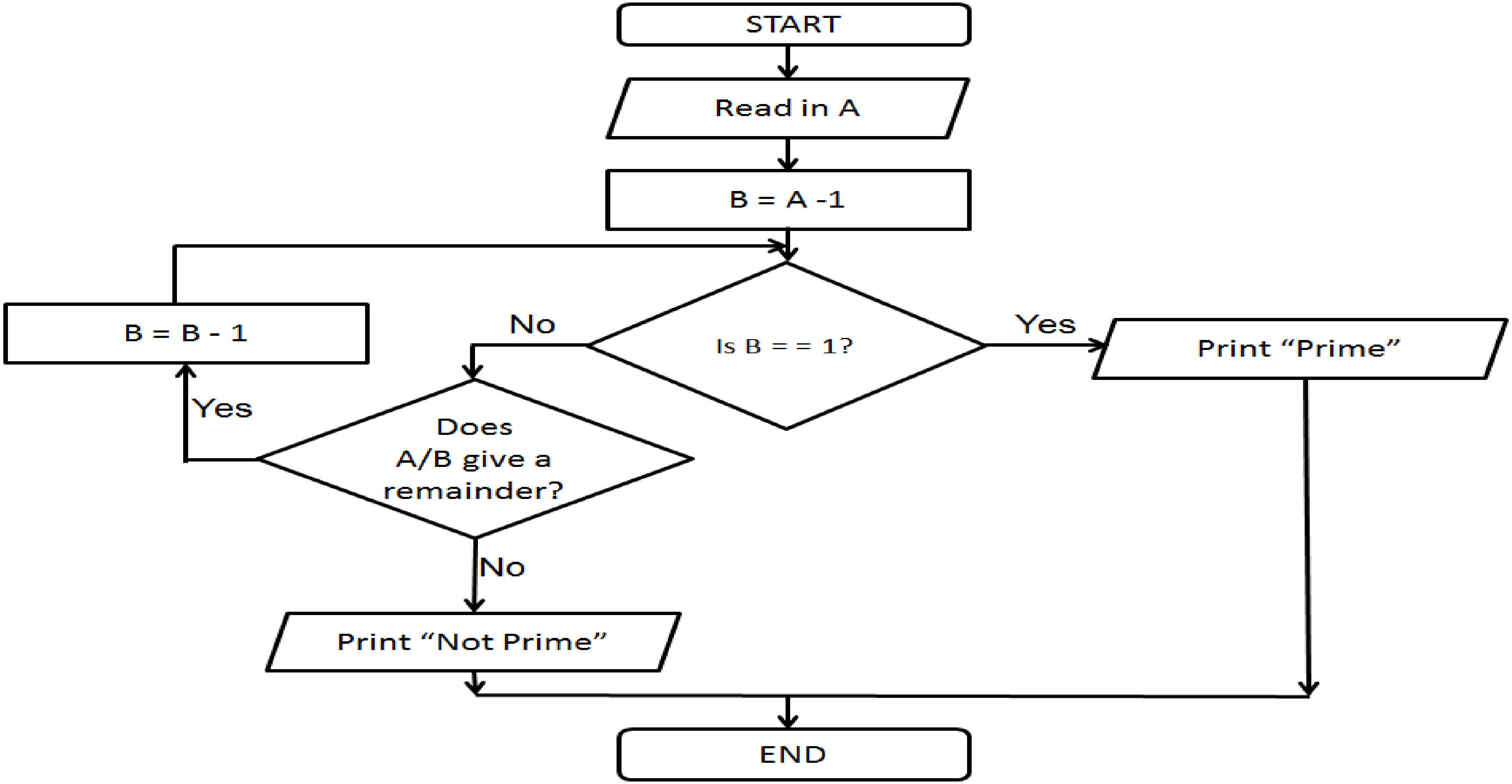
Pseudocode

```
PROGRAM PrintSum1to5
   Set Total = 0
   Set A = 1
   WHILE (A != 6)
   DO
      Total = Total + A
      A = A + 1
   END WHILE
END
```

Flowcharts – check if a prime number (Problem 10)

- So let's say we want to express the following algorithm:

Read in a number and check if it's a prime



Structured English and Pseudocode

Structured English

```
PROGRAM Prime
    Read in a number
    Divide that number by
    every number less than it,
    but greater than one
    If any of them divide in
    evenly, then it’s not prime,
    otherwise it is prime
END
```

Pseudocode

```
PROGRAM Prime
  Read A
  Compute B = A - 1
  IsPrime=True
  WHILE (B != 1)
  DO
     IF (A/B gives no remainder) THEN
       IsPrime= False
     ENDIF
     B = B – 1
  END WHILE
  IF (IsPrime == true) THEN
     Print “Prime”
  ELSE
     Print “Not Prime”
  ENDIF
END
```

References

- 2007, Lesley Anne Robertson, Simple Program Design A Step-by-Step Approach, Fifth edition.