

Program Design

Lecture 8

Principle of good programming

- The program requirements must be specified in full and in writing. These specifications will be prepared by a systems analyst.
- A programmer has the task of converting these specifications into a written program.
- In developing a program, programmers should keep “working papers”. The “working papers” might include a decision table or flowchart (or both). They can refer back to these papers later to check what they have done in case:
 - there is an error in the program for correction;
 - the user of the program asks for a change in the program, e.g. for an extra bit of processing on input data, perhaps to produce an additional report.

Pseudocode and Flowcharts

- Flowcharts and pseudocode can be a really useful tool for a programmer. They can use the designs from the pseudocode and flowchart to guide them in terms of what they need to program (code).
- These documents are to a programmer are like blueprints to a building developer.

What is Pseudocode

- Pseudocode is easy to read and write. It allows the algorithm designer and programmer to concentrate on the logic of the problem.
- Pseudocode uses the structural conventions of modern programming languages. It is English that has been formalised and abbreviated to look like the high-level computer languages.

No standard Pseudocode

In general:

- 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.

How to Write Pseudocode

- There are six basic computer operations.
 - There are common words and keywords used to represent these operations in pseudocode.
- Each operation can be represented as a straightforward instruction in English, with **keywords** and indentation to signify a particular control structure.

Operation 1: To Receive Information

- When a computer is required to receive information or input from a particular source, whether it be a terminal, a disk or any other device, the verbs Read and Get are used.
 - Read is usually used when to receive input from a record on a file
 - Get/input is used when to receive input from the keyboard.

Operation 1: To Receive Information

- For example, typical pseudocode instructions to receive information are:
 - **Read** student_name
 - **Read** number_1, number_2
 - **Get** system_date
 - **Get** tax_code
- Each example uses a single verb, Read or Get, followed by one or more nouns to indicate what data is to be obtained.

Operation 2: To Put Out information

- When a computer is required to supply information or output to a device, the verbs **Print** and **Write** are used in the pseudocode.
 - **Print** is usually used when the output is to be shown directly to the screen or sent to a printer.
 - **Write** is used when the output is to be written to a file.

Operation 2: To Put Out Information – cont.

- Often an output **Prompt** instruction is required before an input **Get** instruction. The Prompt verb causes a message to be sent to the screen, which requires the user to respond, usually by providing input. For example:
 - **Prompt** for student_mark
 - **Get** student_mark
- Python – does the prompt and get in one statement e.g.
 - `user_char = input("Please enter an character:\n")`

Operation 3: To Perform Arithmetic

- Most programs require the computer to perform some sort of mathematical calculation, or to apply a formula, and for these a programmer may use either actual mathematical symbols or the words for those symbols.

Operation 3: To Perform Arithmetic

- To be consistent with high-level programming language, the following symbols can be written in pseudocode:
 - + for add
 - for subtract
 - * for multiply
 - / for divide
 - = indicate assignment of a value as a result of some processing
 - () for parentheses

Operation 3: To Perform Arithmetic

Examples:

`average_marks = total_marks / student_count`

`sales_tax = cost_price * 0.10`

$C = (F - 32) * 5/9$

Operation 3: To Perform Arithmetic

- When writing mathematical calculations for the computer, the standard mathematical order of operations applies to pseudocode and to most computer language.
- The first operation carried out will be any calculation contained with parentheses. Next, any multiplication or division, as it occurs from left to right, will be performed. Then, any addition or subtraction, as it occurs from left to right, will be performed.

Operation 4: To assign a value to a variable or memory location

There are 3 instances in which you may write pseudocode to assign a value to a variable or memory location:

- To give data an initial value in pseudocode, the verbs Initialise or Set can be used, or the '=' symbol.
- To assign a value as a result of some processing, the symbols "=" or "<-" are written.
- To keep a variable for later use, the verbs Save or Store are used.

Operation 4: To assign a value to a variable or memory location

- As a convention in our class we will use = as the assignment operator:
 - `total_price = 0`
 - `student_count = 0`
 - `total_price = cost_price + sales_tax`

Operation 5: Decision (If Else)

- To represent this operation in pseudocode, special keywords are used: **If**, **Else**, **EndIf**.
- The comparison of data is established in the IF clause, and the choice of alternatives is determined by block after the IF or ELSE options. Only one of these alternatives will be performed.

Operation 5: Decision (If Else)

A typical pseudocode example to illustrate this operation is:

```
If student_attendance_status == part_time
    part_time_count = part_time_count + 1
Else
    full_time_count = full_time_count + 1
EndIf
```

Operation 6: To repeat a group of actions (Loop)

- When there is a sequence of processing steps that need to be repeated, two special keywords, **WHILE....END WHILE**, are used in pseudocode. The condition for the repetition of a group of actions is established in the WHILE clause, and the actions to be repeated are listed beneath DO. For example:

```
While student_total < 50
    Read student record
    Write student name, address to report
    student_total = student total + 1
EndWhile
```

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

- **Sequence** - The sequence control structure is the straightforward execution of one processing step after another.
- **Selection** – The selection control structure is the presentation of a condition and the choice between two actions.
- **Repetition** – The repetition control structure can be defined as the presentation of a set of instructions to be performed repeatedly, as long as a condition is true.

The Selection Control Structure

Simple selection

```
If...  
  ...  
Else  
  ...  
EndIf
```

Combined selection

```
If... And...  
  ...  
EndIf  
  
If... Or ...  
  ...  
EndIf
```

Nested selection

```
If ...  
  ...  
Else  
  If ...  
    ...  
  Else  
    ...  
  EndIf  
EndIf
```

Repetition Control Structure

While...

...

EndWhile

For ...

...

EndFor

Example - Exponentiation

Program Exponentiation

Prompt for base, expon

Get base, expon

i = 0

result = 1

While i<expon:

 result = result*base

 i = i+1

EndWhile

Print(result)

EndProgram

Example - Factorial

Program Factorial

Prompt for num

Get num

i=0

factorial = 1

While i < num:

 i=i+1

 factorial = factorial *i

EndWhile

Print(factorial)

EndProgram

Example Guessing Game

Program Guess

```
target = randint(1,10)
numGuesses = 0
won = False
Print(Instructions)
While numGuesses < 3 and won == False:
    Prompt for currGuess
    Get currGuess
    numGuesses = numGuesses+1
    If currGuess == target:
        print('Congratualtions you won! Dance around and celebrate')
        won = True
    EndIf
EndWhile
If won == False:
    print('Sorry you lost, better luck next time.')
EndIf
EndProgram
```

Flowchart and programming

- A flowchart can be a really useful tool for a programmer. They can use the flowchart to guide them in terms of what they need to program (code).
- Flowcharts to a programmer is like the blue print to a architect

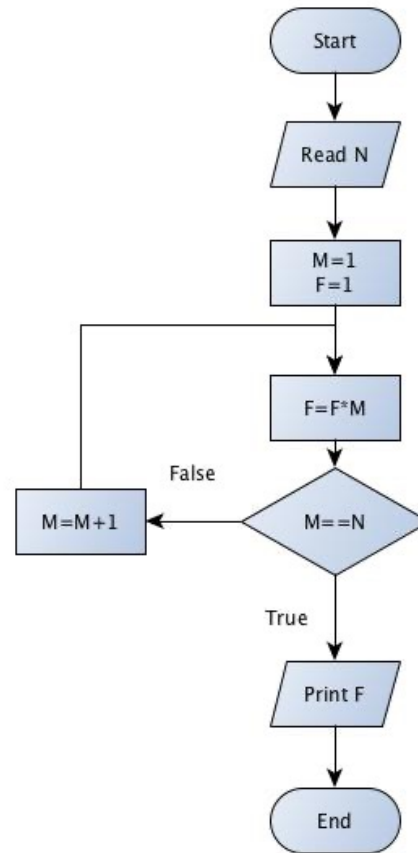
Uses (Advantages) of flowcharts

- To clarify the logic of a problem.
- To analyse the actions resulting from a set of conditions.
- To sort out the procedural steps in the program.
- As aids to program construction and coding.
- As communicating documents, e.g. to explain the program to other programmers and the system analyst.






Disadvantages of flowcharts






















- Complicated processing might require flowcharts that stretch on to several pages.
- They are not easy to amend. Alterations might involve a complete re-drawing of the flowchart, which could be a time consuming task.
- Not similar to code – need to be translated.

Example of a Flowchart



Flowchart Symbols

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

	Process Any processing function.		Sequential Data Data that is accessible sequentially, such as data stored on magnetic tape.		Parallel Mode Indicates the synchronization of two or more parallel operations.
	Terminator Indicates the beginning or end of a program flow in your diagram.		Direct Data Data that is directly accessible, such as data stored on disk drives.		Loop limit Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.
	Decision Decision point between two or more paths in your flowchart.		Manual Input Data that is entered manually, such as with a keyboard or barcode reader.		On-page Reference Use this shape to create a cross-reference from one process to another on the same page of your flowchart.
	Document Data that can be read by people, such as printed output.		Card Data that is input by means of cards, such as punch cards or mark-sense forms.		Off-page Reference shapes Use this shapes to create a cross-reference and hyperlink from a process on one page to a process on another page.
	Data Can represents any type of data in a flowchart.		Paper Tape Data that is stored on paper tape.		Yes/No decision indicators
	Predefined Process A named process, such as a subroutine or a module.		Display Data that is displayed for people to read, such as data on a monitor or projector screen.		Condition
	Stored Data Any type of stored data.		Manual Operation Any operation that is performed manually (by a person).		Control Transfer A location in your diagram where control is transferred. The triangle can be positioned anywhere on the line.

Parallel Mode

Indicates the synchronization of two or more parallel operations.

Loop limit

Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

On-page Reference

Use this shape to create a cross-reference from one process to another on the same page of your flowchart.

Off-page Reference shapes

Use this shapes to create a cross-reference and hyperlink from a process on one page to a process on another page.

Yes/No decision indicators

Condition






Control Transfer

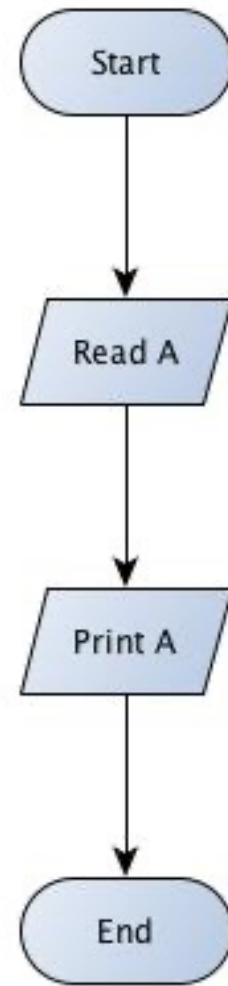
A location in your diagram where control is transferred. The triangle can be positioned anywhere on the line.

Flowcharts (Problem 1)

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

Read in a number and print it out.






Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

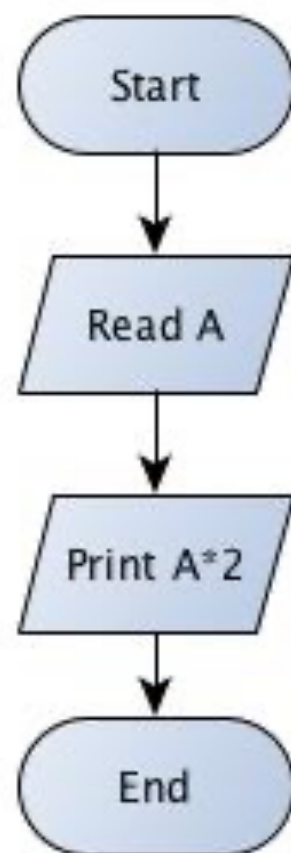


Flowcharts (Problem 2)

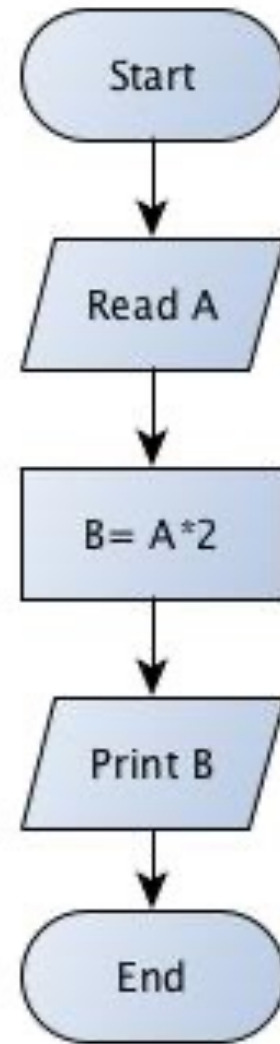
- So let's say we want to express the following algorithm:

Read in a number and print out double the number.

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.



Or Alternatively








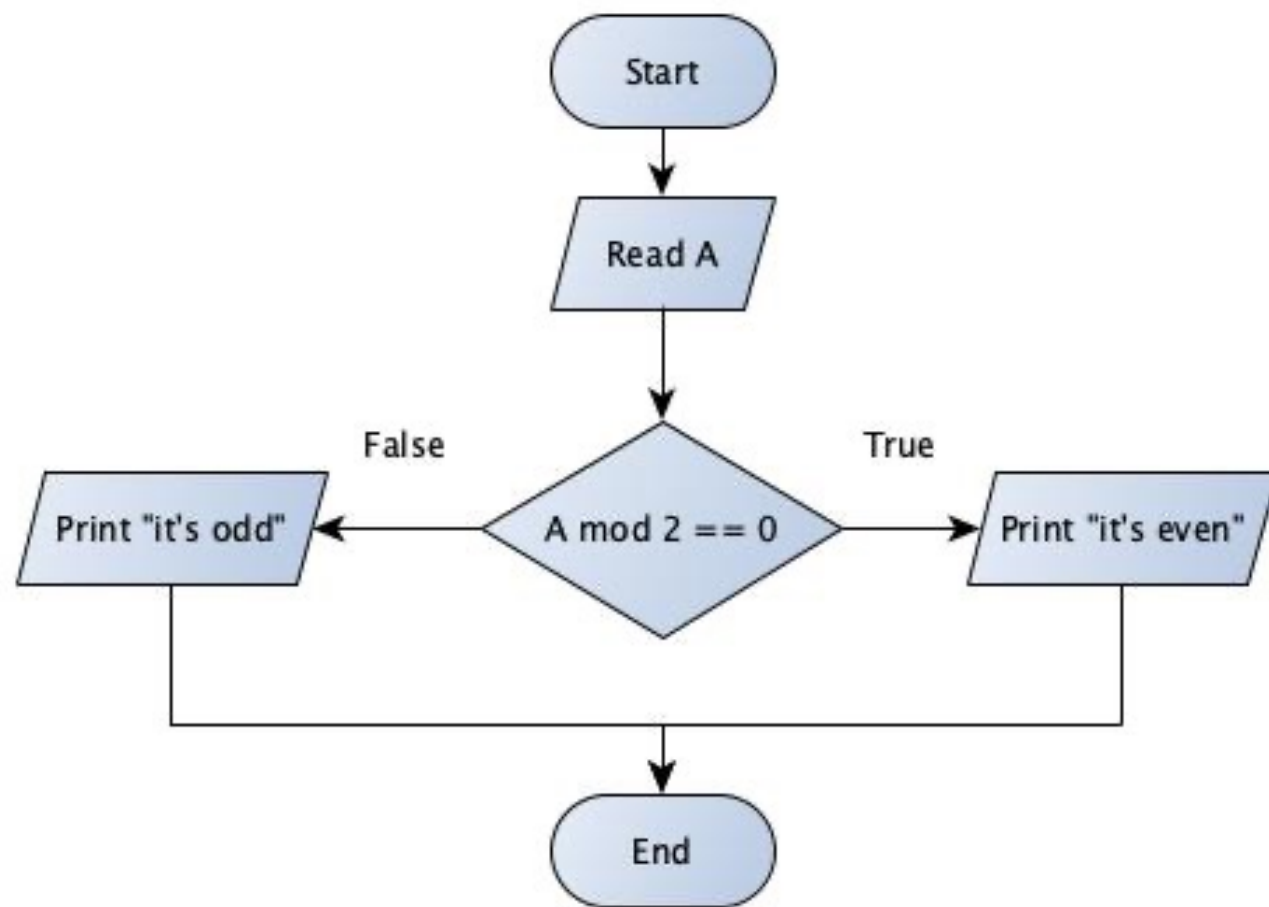
Flowcharts (Problem 3)

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

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

.

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.








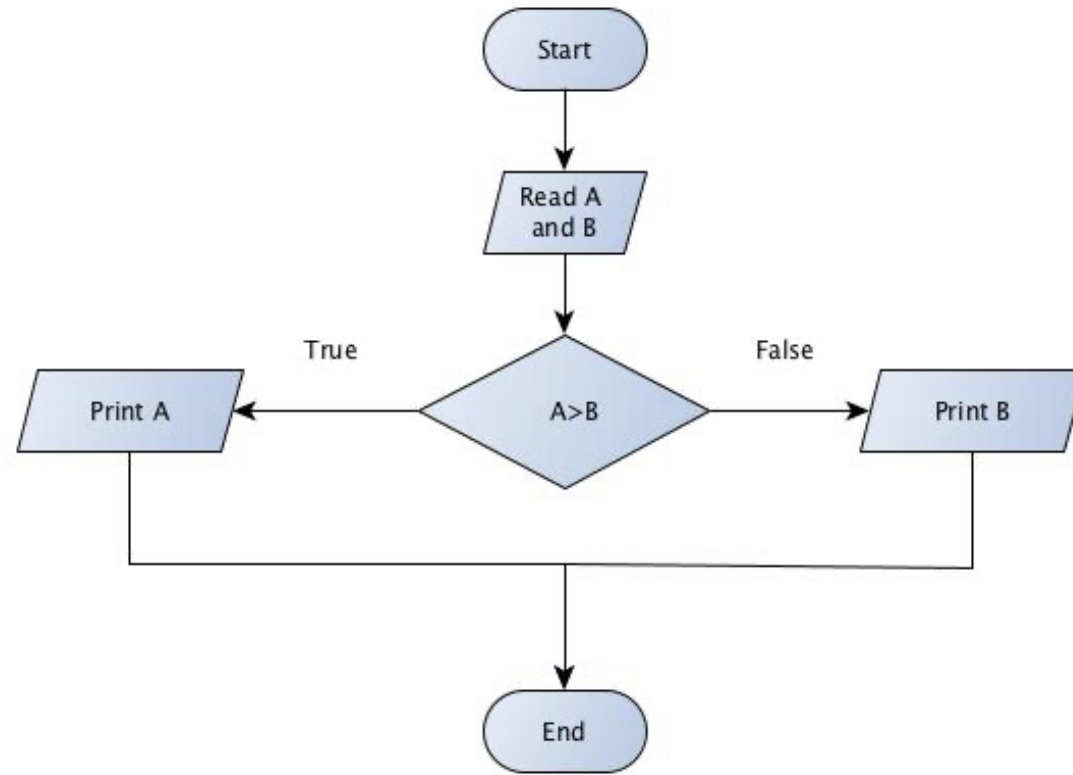
Flowcharts (Problem 4)

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

Read in two numbers, call them A and B. If A is bigger than B, print out A, otherwise print out B

.






Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

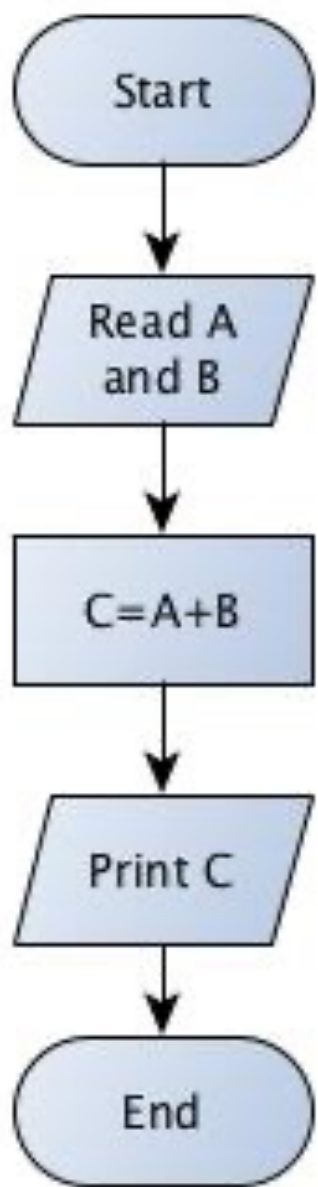


Flowcharts (Problem 5)

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

Read in two numbers, call them A and B. Sum A and B, print out the result






Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.



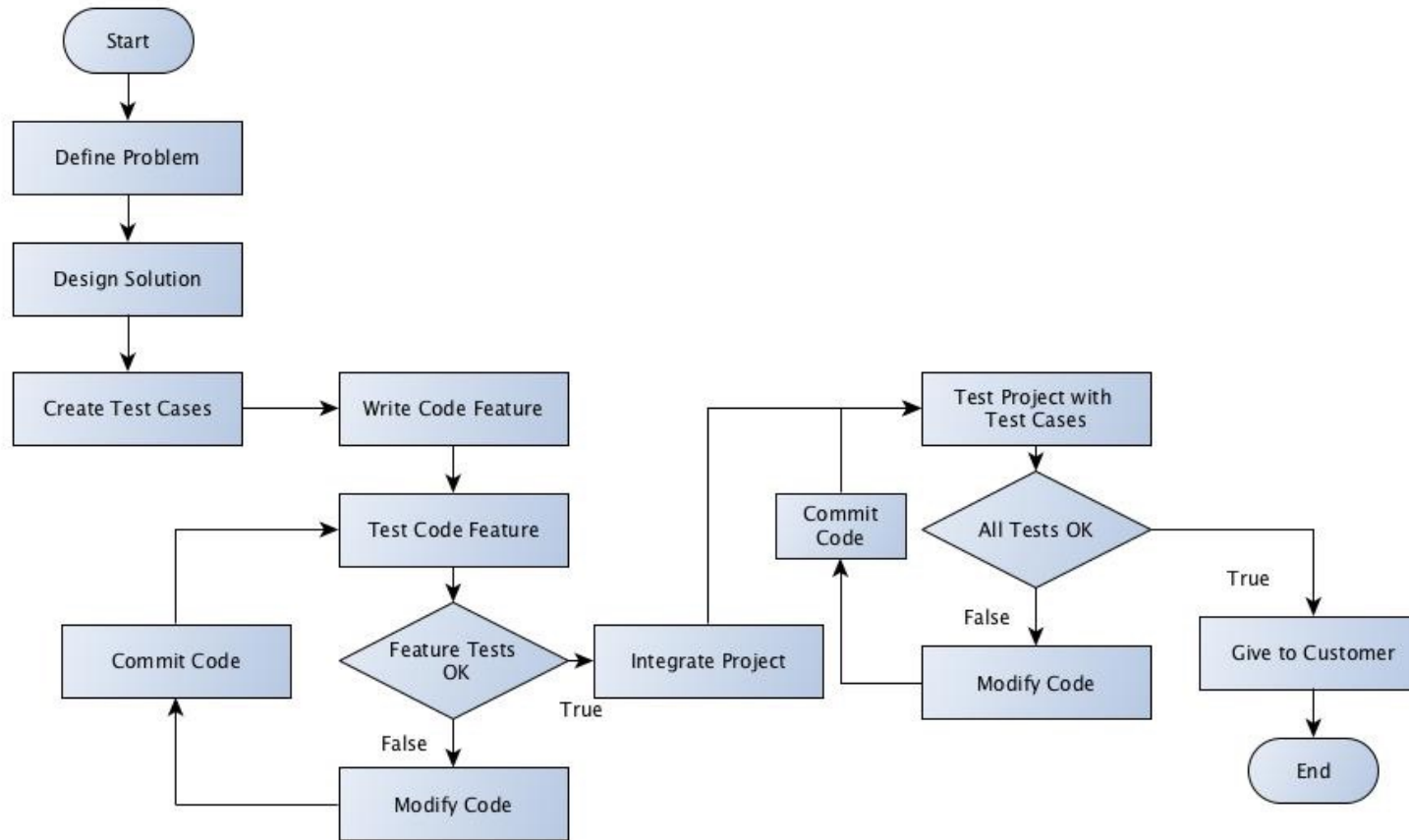
Flowcharts (Problem 6)

Not just for algorithms!

You can express a work process in a flowchart!

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.






Software Development Process

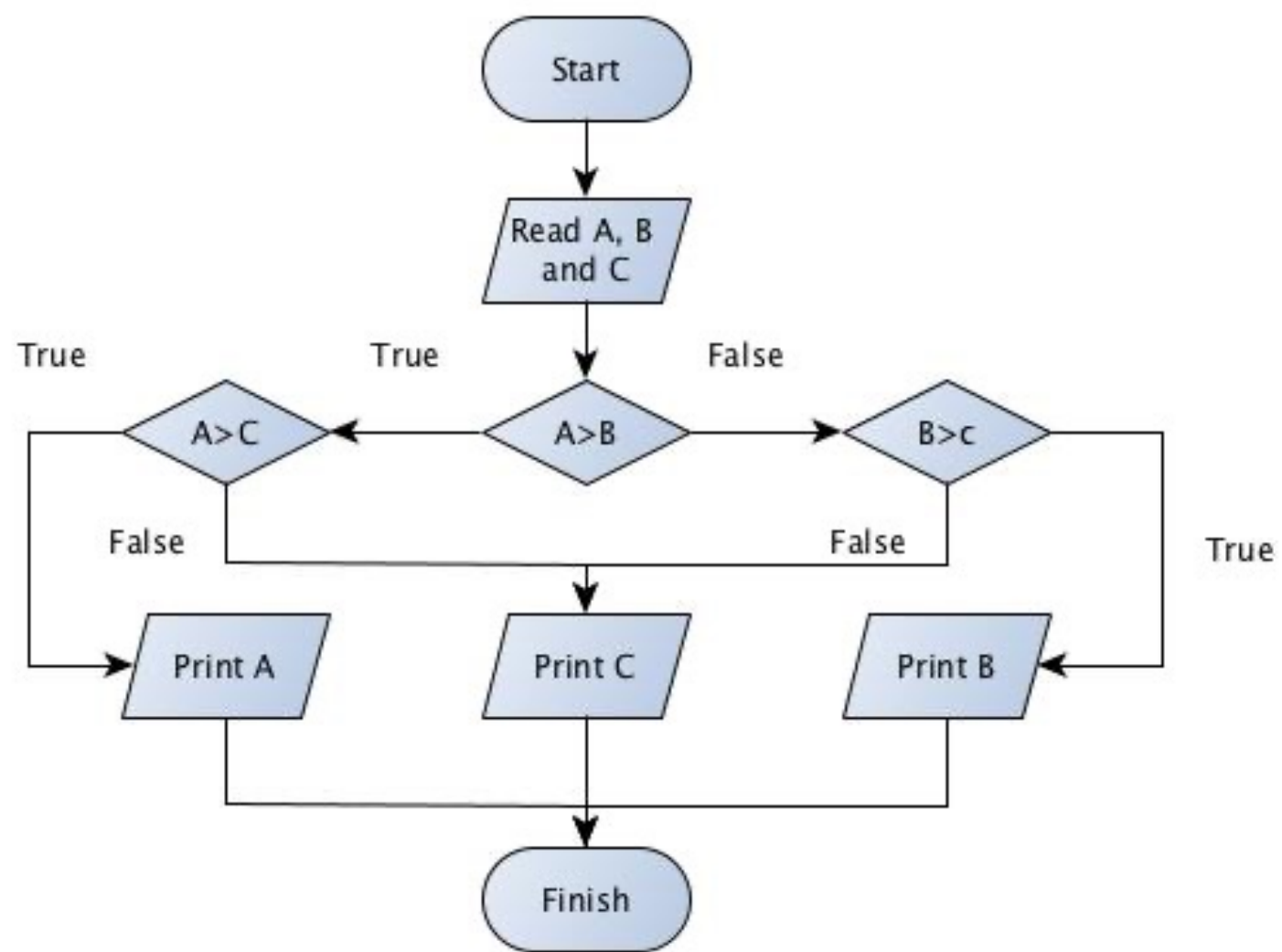


Flowcharts (Problem 7)

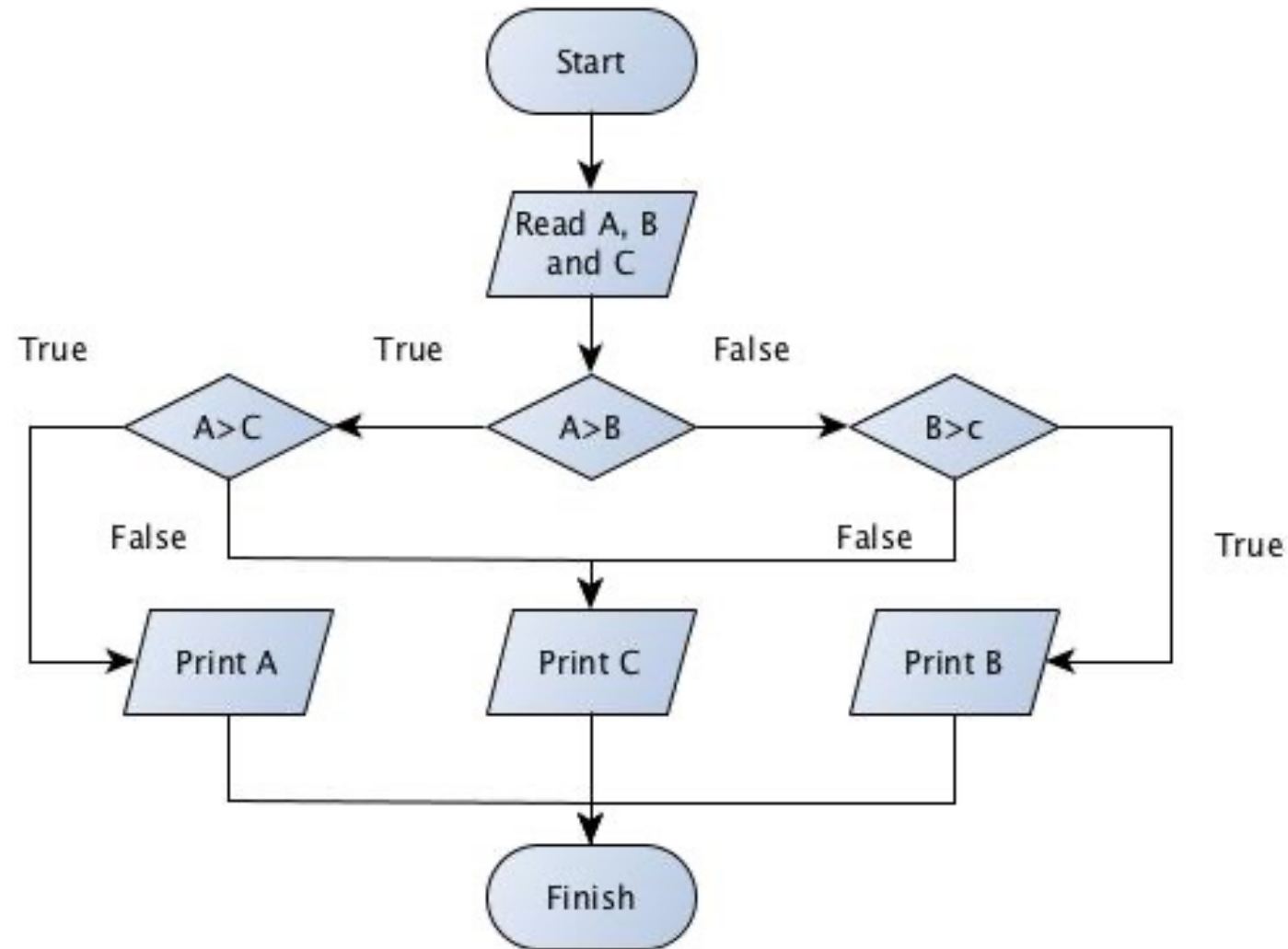
- So let's say we want to express the following algorithm to print out the biggest of three numbers: :

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.

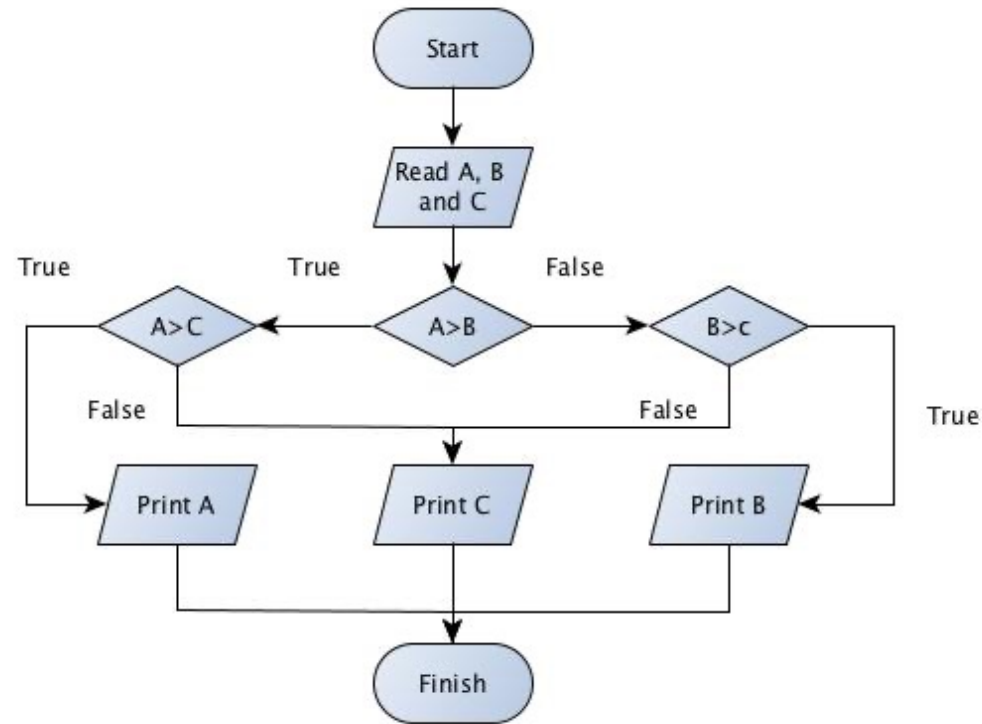
Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.



What happens if you put in $A=B=C=1$?



Can you translate this flowchart to pseudocode?

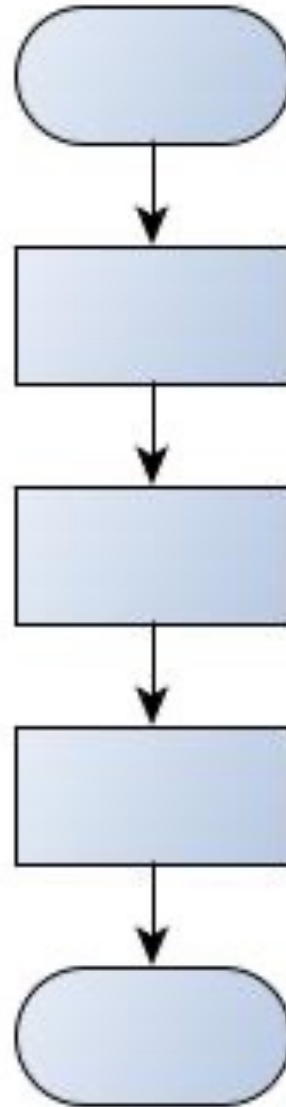


Structured flowcharts

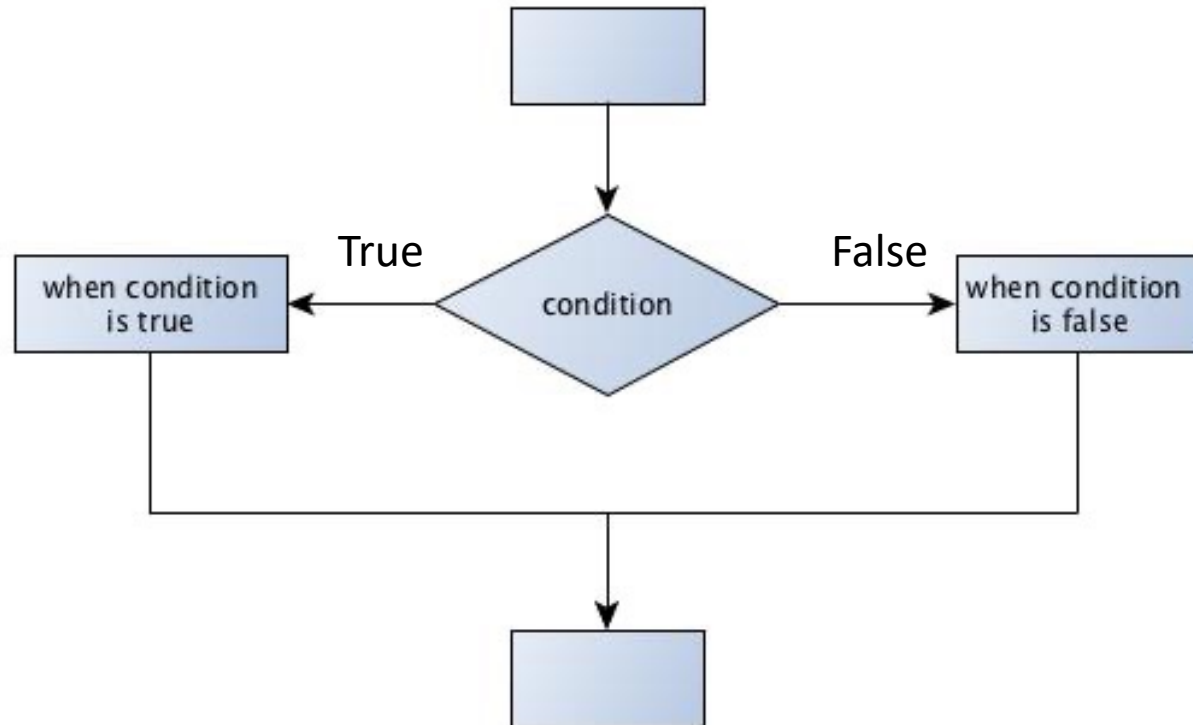
We can make our flowcharts easier to translate and less error prone if we use the basic structures we have been using

- Sequence
- Decision
- Loop

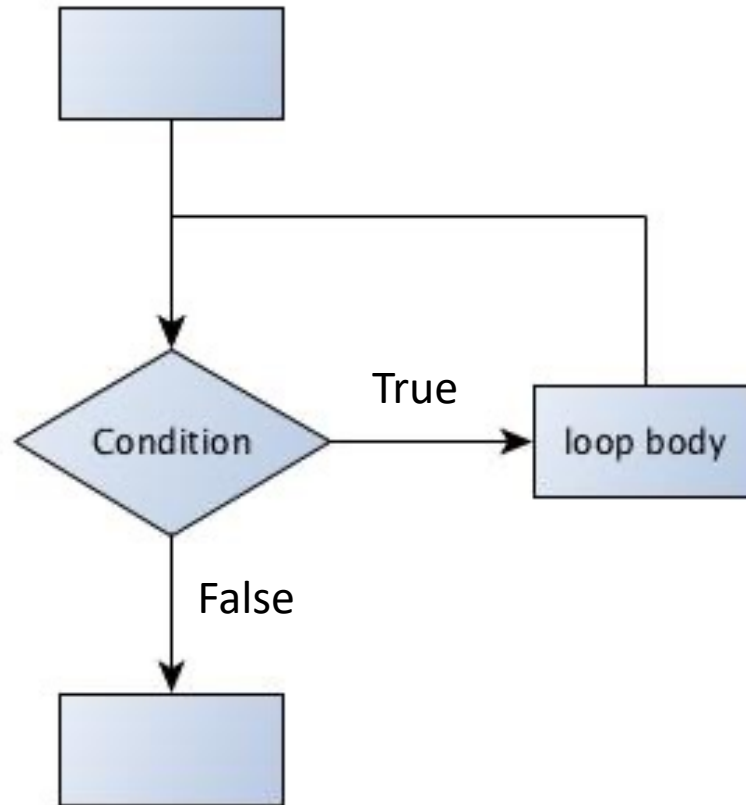
Sequence



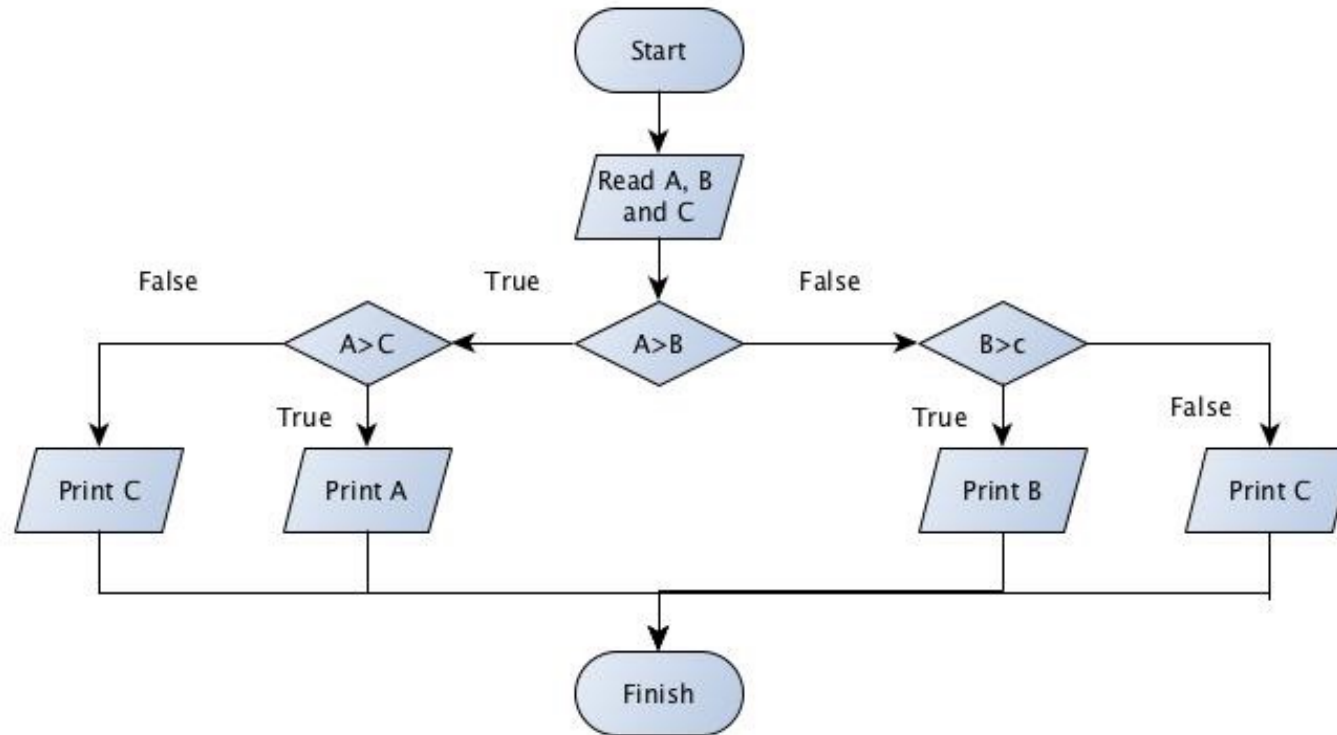
If



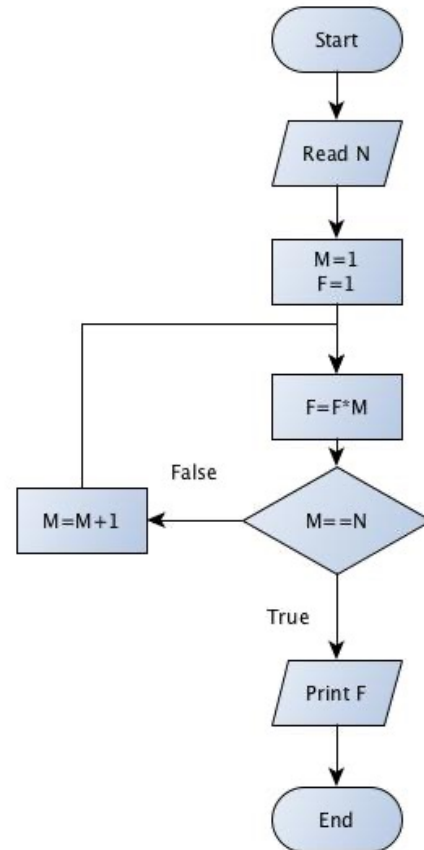
Loop



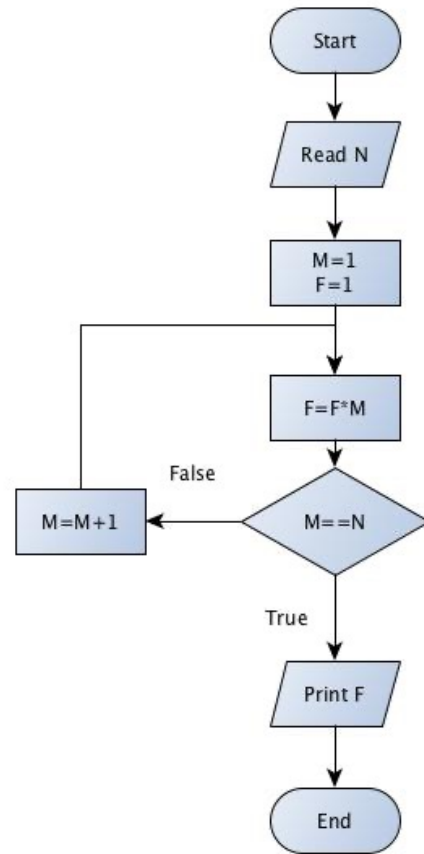
Structured Version



Our factorial example – can this be translated?



Example of a Flowchart



```
N = int(input('please enter a positive integer.\n'))
```

```
M=1
```

```
F = 1
```

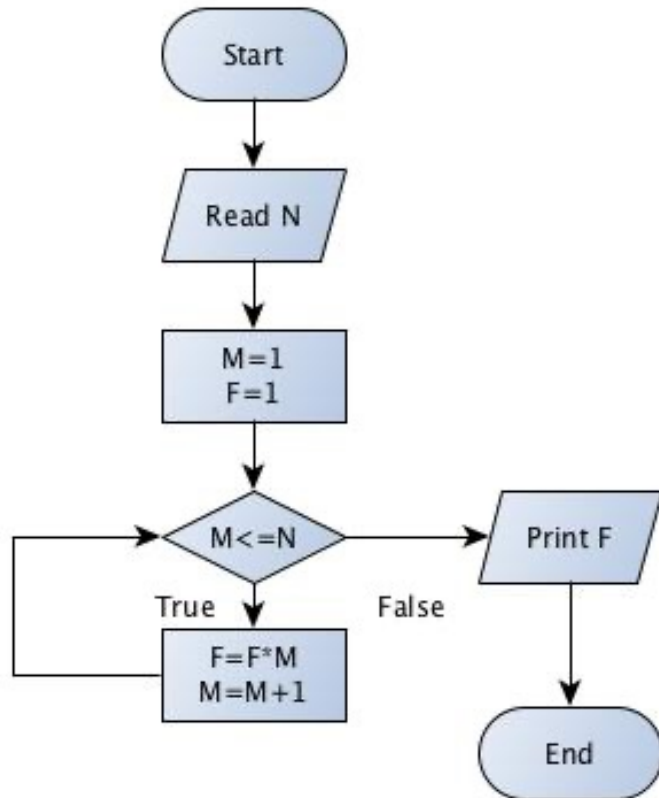
```
while M<= N:
```

```
    F = F * M
```

```
    M = M+1
```

```
print('factorial = ', F, '\n')
```


Structured Version



```
N = int(input('please enter a positive integer.\n'))
M=1
F = 1
while M<= N:
    F = F *M
    M=M+1
print('factorial = ', F, '\n')
```