ALGORITHM

ALGORITHM: Introduction

An algorithm is a procedure or method that solves instances of a problem or problem as whole.

In simple words, we can say “set of instructions or rules in a uniform language that solves the problem”

Instruction in algorithm,

Add A, B & store in C
    Or
C is equal to A plus B

ALGORITHM: Introduction

Qualities of a good algorithm

- Input and output should be defined precisely.
- Each step in an algorithm should be clear and unambiguous.
- Algorithms should be most effective among many different ways to solve a problem.
- An algorithm shouldn't include computer code. Instead, the algorithm should be written in such a way that it can be used in different programming languages.

ALGORITHM: Formal Definition

Definition: An algorithm is a sequence of unambiguous instructions for solving a problem.

For an algorithm to be an acceptable solution to a problem, it must also be <u>effective</u>.

That is, it must give a solution in a ‘reasonable’ amount of time

ALGORITHM: Properties

Basic properties are:

Finite: the algorithm must eventually terminate

Complete: Always give a solution when one exists

Correct (sound): Always give a correct solution

ALGORITHM: Designing

A general approach to designing algorithms is as follows

- Understanding the problem, assess its difficulty
- Choose an approach (e.g., exact/approximate, deterministic/ probabilistic)
- Choose a strategy
- Prove
  - Termination
  - Completeness
  - Correctness/soundness
- Evaluate complexity
- Implement and test it
- Compare to other known approach and algorithms

ALGORITHM: Example (MAX)

When designing an algorithm, we usually give a formal statement about the problem to solve.

Problem

Given: a set A={a1,a2,…,an} of integers

Question: find the maximum integer A_i

ALGORITHM: Example (MAX)

A straightforward idea is

1. Start
2. Input 'n', no of elements in list.
3. Input set of integer values in A={a1,a2,…,an}
4. Simply assign a1 to a new variable max (assuming first value as max)
5. check max with all elements of list, if any value in list is greater than max, update stored max to that value
6. Output max is the maximum value
7. Stop

ALGORITHM: Example (Factorial)

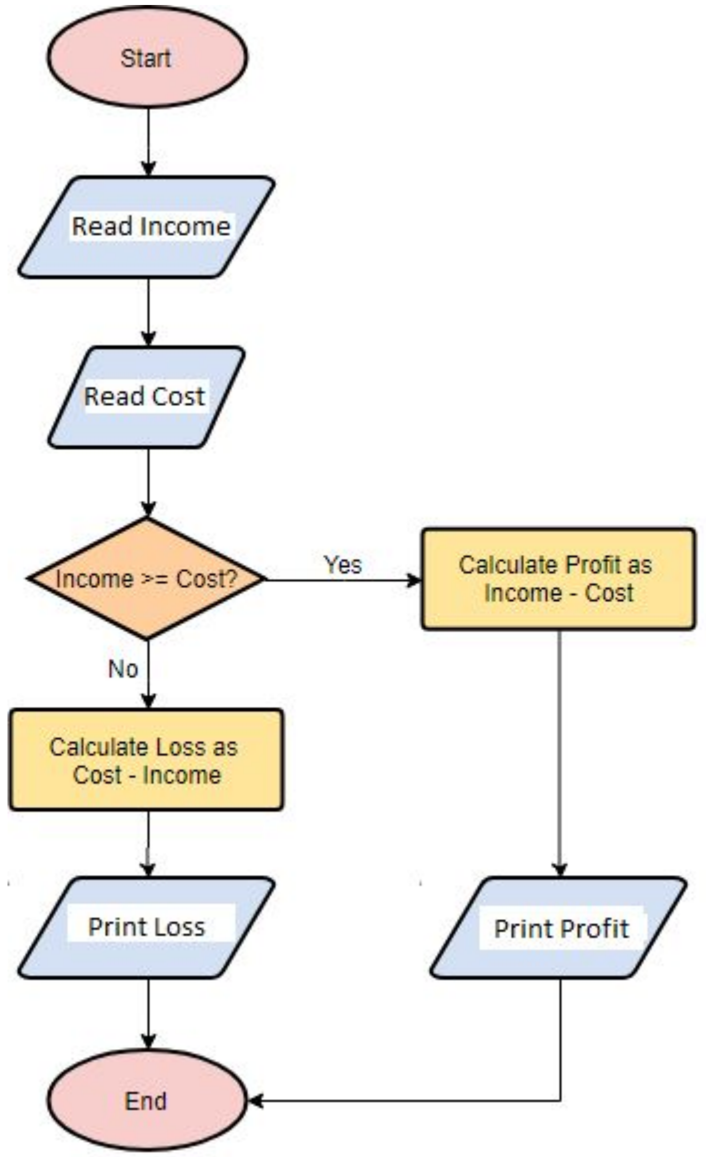
Algorithm : a

```
    Step 1 : Start
    Start 2 : Read n
    Start 3 : Initialize counter variable i to 1 and fact to 1
    Start 4 : if i <= n go to step 5 otherwise goto step 7
    Start 5 : calculate fact = fact * i
    Start 6 : increment counter variable i and goto step 4
    Start 7 : Write fact.
    Start 8 : Stop
```

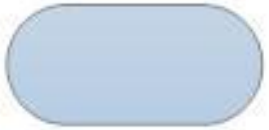
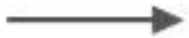
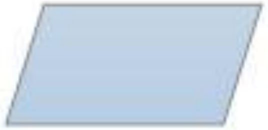
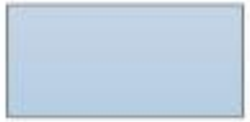
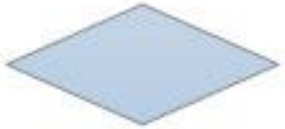
Flowchart

What is a Flowchart?

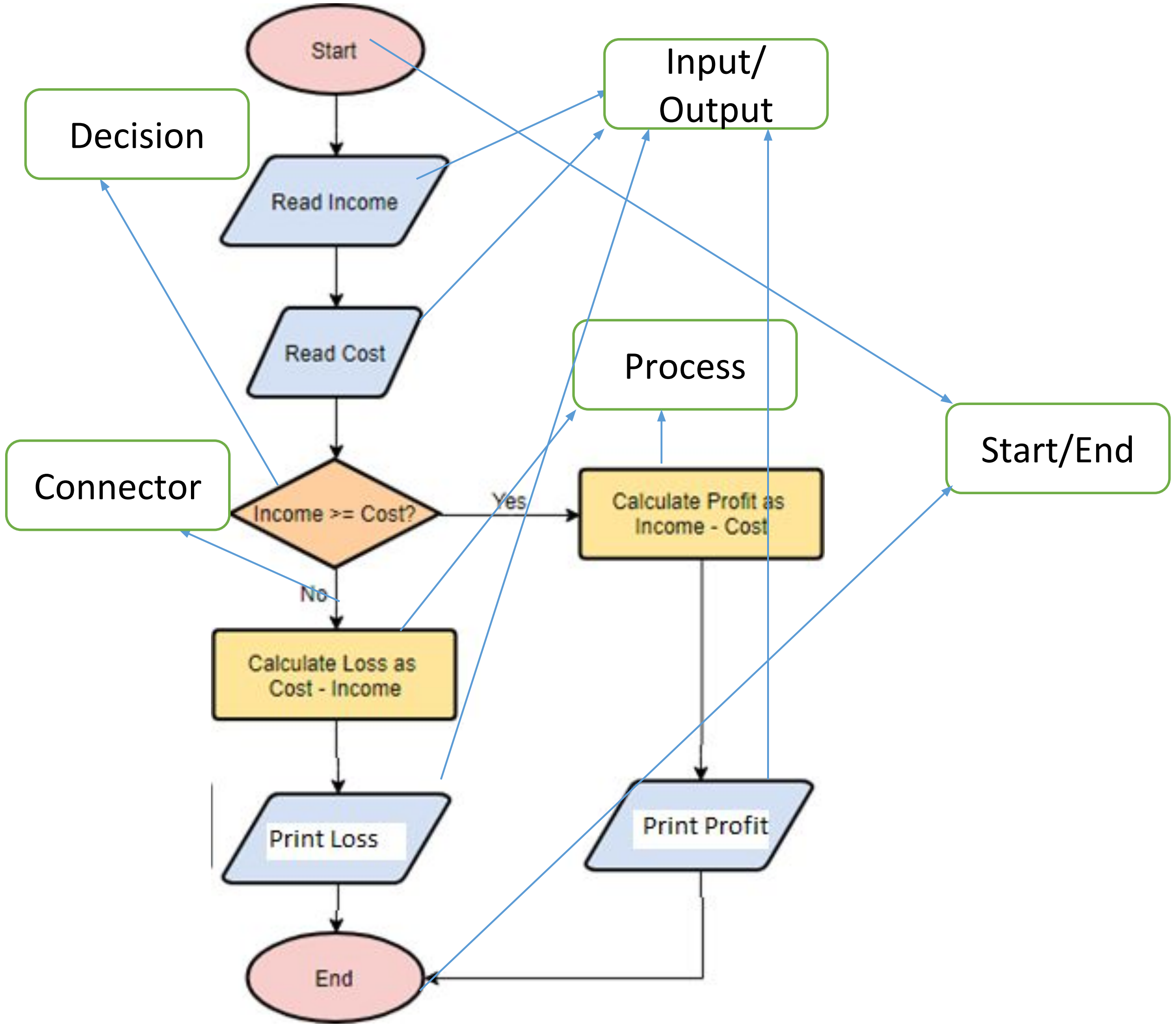
- A flowchart is a graphical representation that depicts the "flow" of steps to solve a problem.
- The figure shown the flowchart for the profit & loss.



Basic 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

Basic Flowchart Symbols

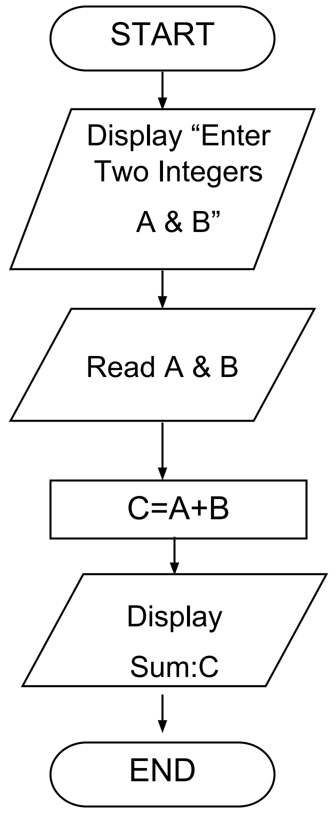


Four Flowchart Structures

- Sequence
- Decision
- Repetition
- Case

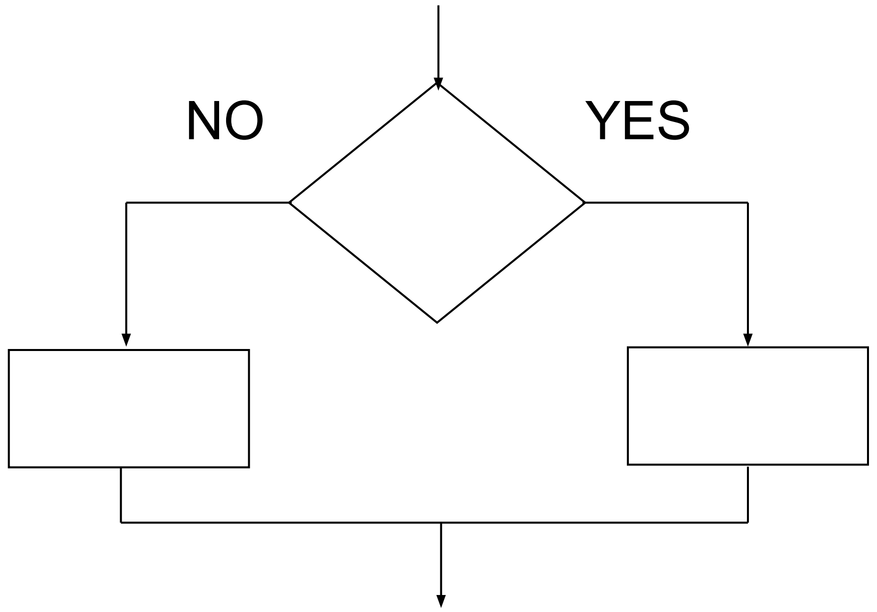
Sequence Structure

- a series of actions are performed in sequence
- The sum of two numbers is an example sequence flowchart.



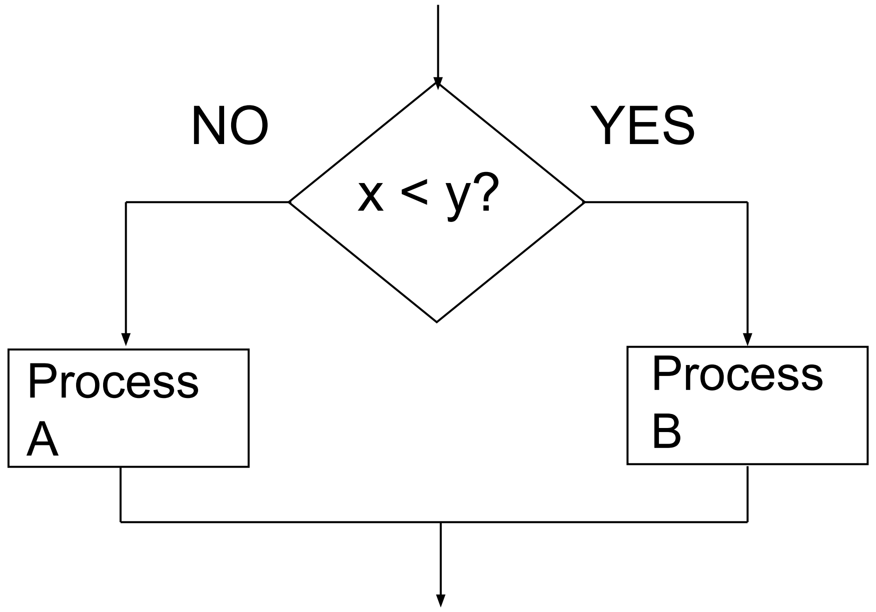
Decision Structure

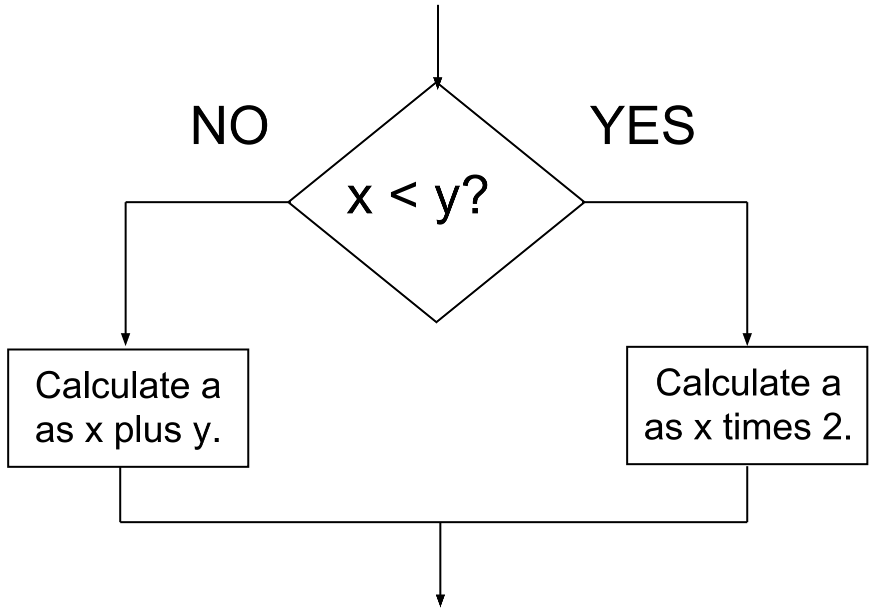
- One of two possible actions is taken, depending on a condition.
- A new symbol, the diamond, indicates a yes/no question. If the answer to the question is yes, the flow follows one path. If the answer is no, the flow follows another path



Decision Structure

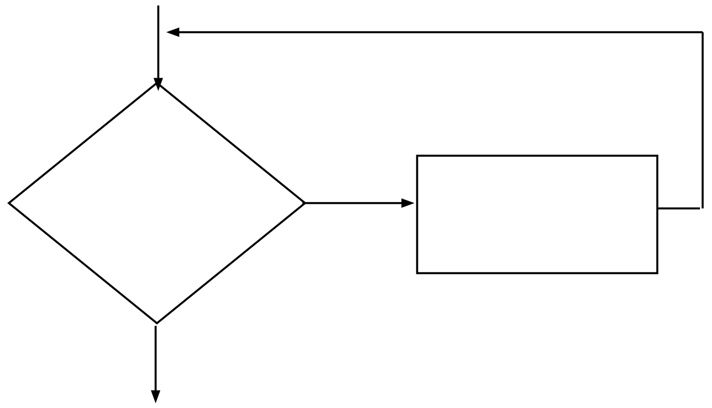
- In the flowchart segment below, the question “is x < y?” is asked. If the answer is no, then process A is performed. If the answer is yes, then process B is performed.
- The flowchart segment below shows how a decision structure is expressed as an if/else statement.





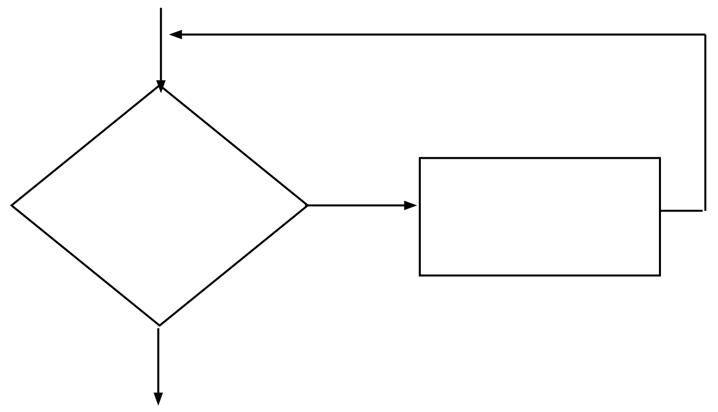
Repetition Structure

- A repetition structure represents part of the program that repeats. This type of structure is commonly known as a loop.



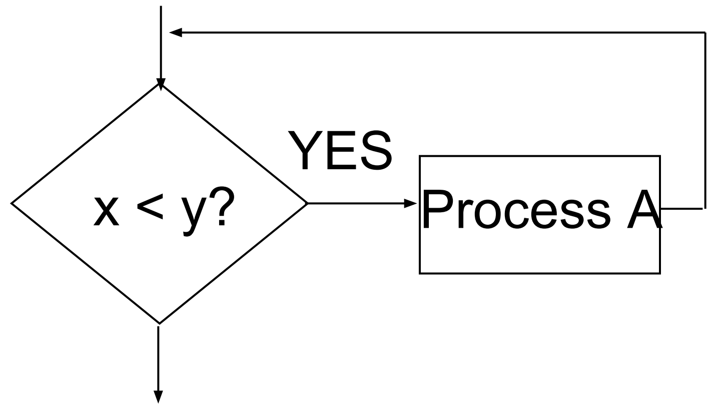
Repetition Structure

- Notice the use of the diamond symbol. A loop tests a condition, and if the condition exists, it performs an action. Then it tests the condition again. If the condition still exists, the action is repeated. This continues until the condition no longer exists.



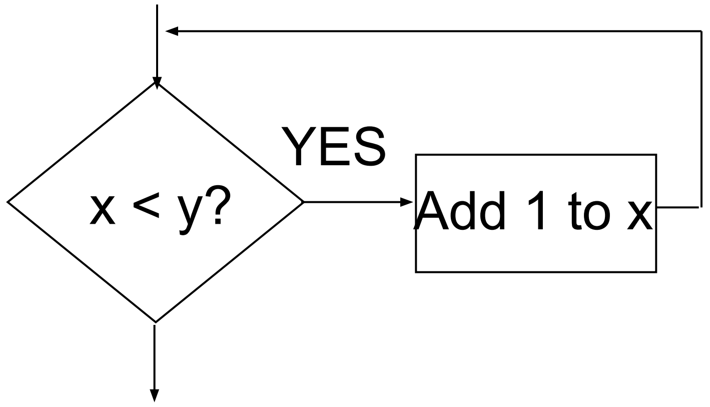
Repetition Structure

- In the flowchart segment, the question “is x < y?” is asked. If the answer is yes, then Process A is performed. The question “is x < y?” is asked again. Process A is repeated as long as x is less than y. When x is no longer less than y, the repetition stops and the structure is exited.



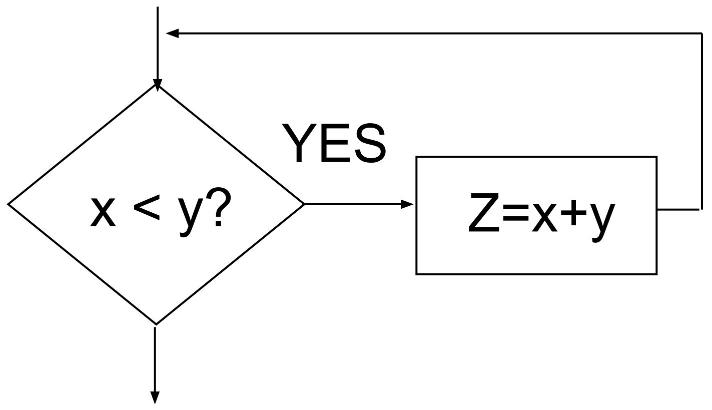
Repetition Structure

- The flowchart segment below shows a repetition structure expressed as a while loop.



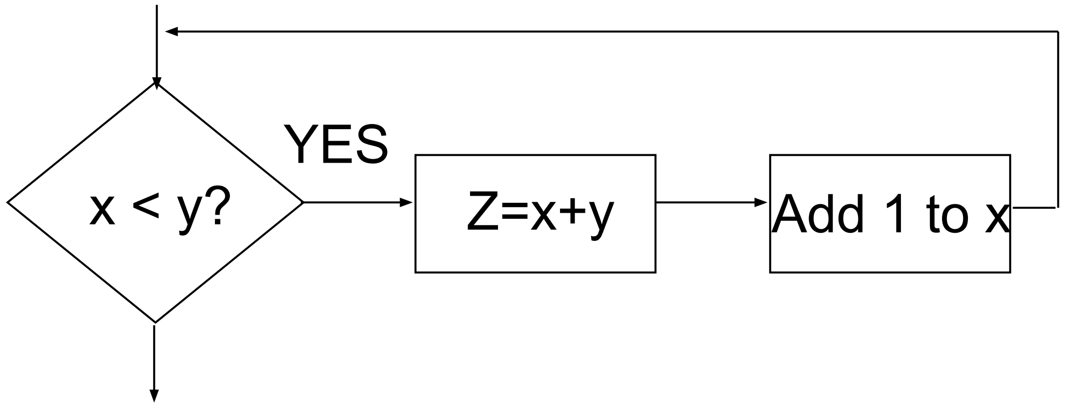
Controlling a Repetition Structure

- The action performed by a repetition structure must eventually cause the loop to terminate. Otherwise, an infinite loop is created.
- In this flowchart segment, x is never changed. Once the loop starts, it will never end.
- QUESTION: How can this flowchart be modified so it is no longer an infinite loop?



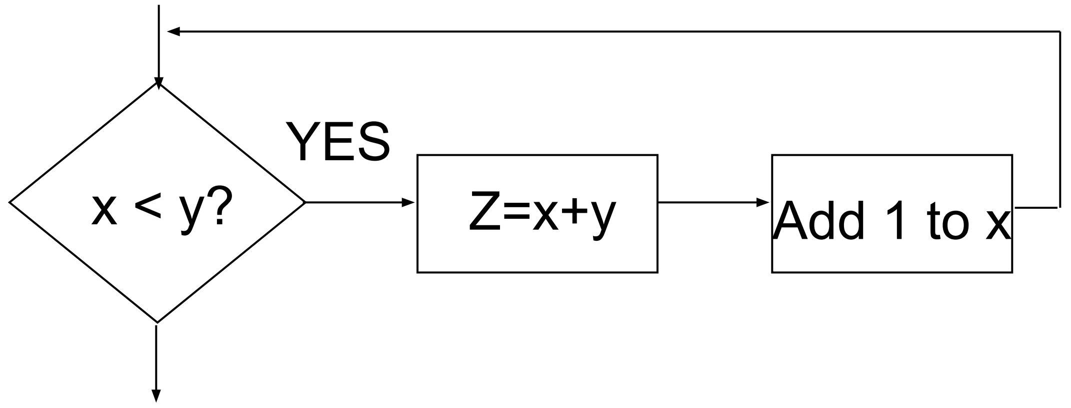
Controlling a Repetition Structure

- ANSWER: By adding an action within the repetition that changes the value of x.



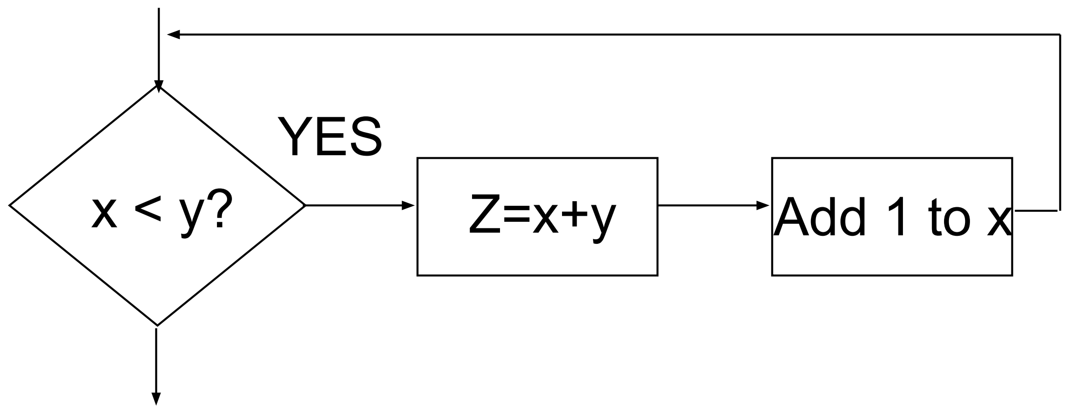
Pre-Test Repetition Structure

- This type of structure is known as a pre-test repetition structure. The condition is tested *BEFORE* any actions are performed.



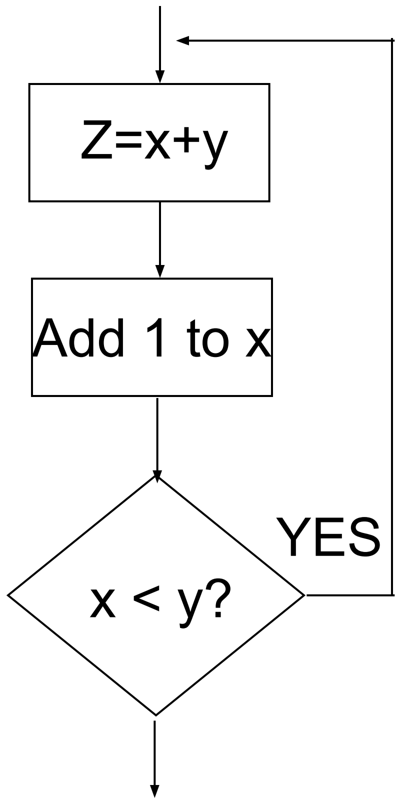
Pre-Test Repetition Structure

- In a pre-test repetition structure, if the condition does not exist, the loop will never begin.



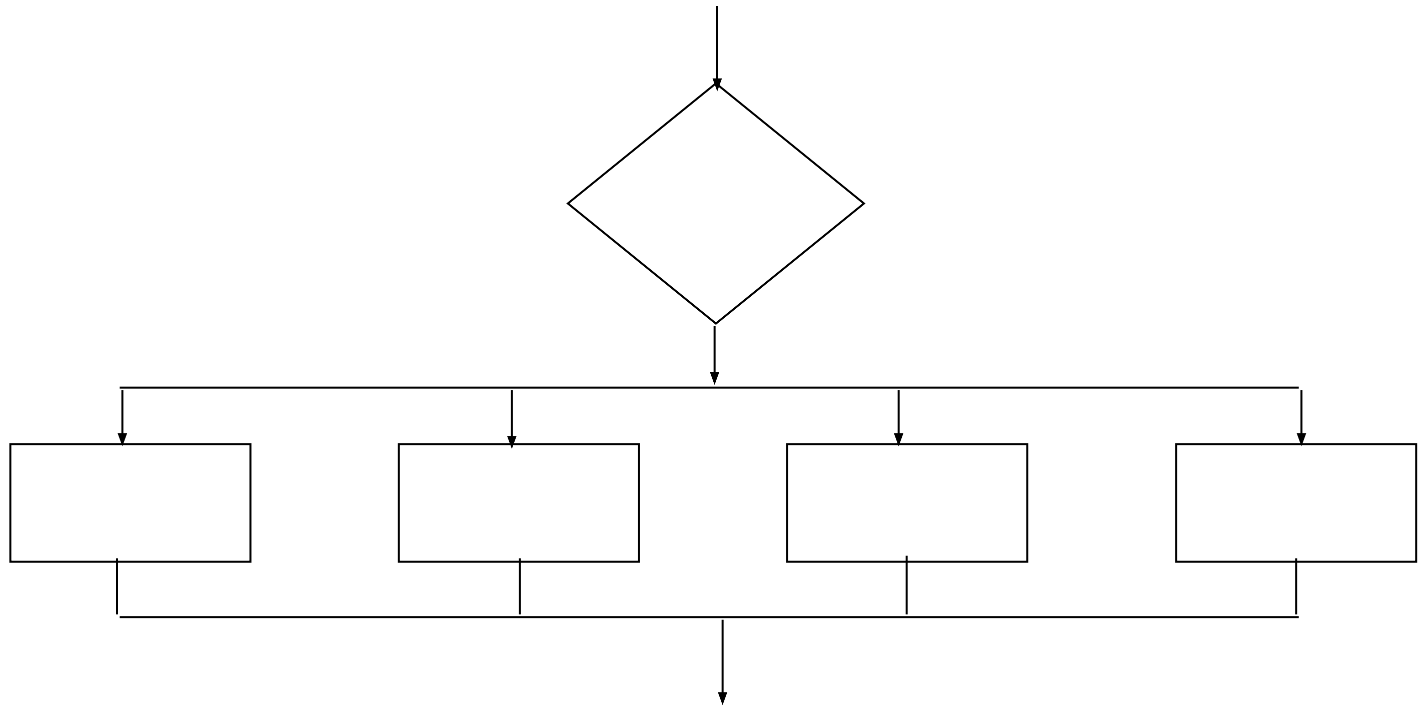
Post-Test Repetition Structure

- This flowchart segment shows a post-test repetition structure.
- The condition is tested *AFTER* the actions are performed.
- A post-test repetition structure always performs its actions at least once.



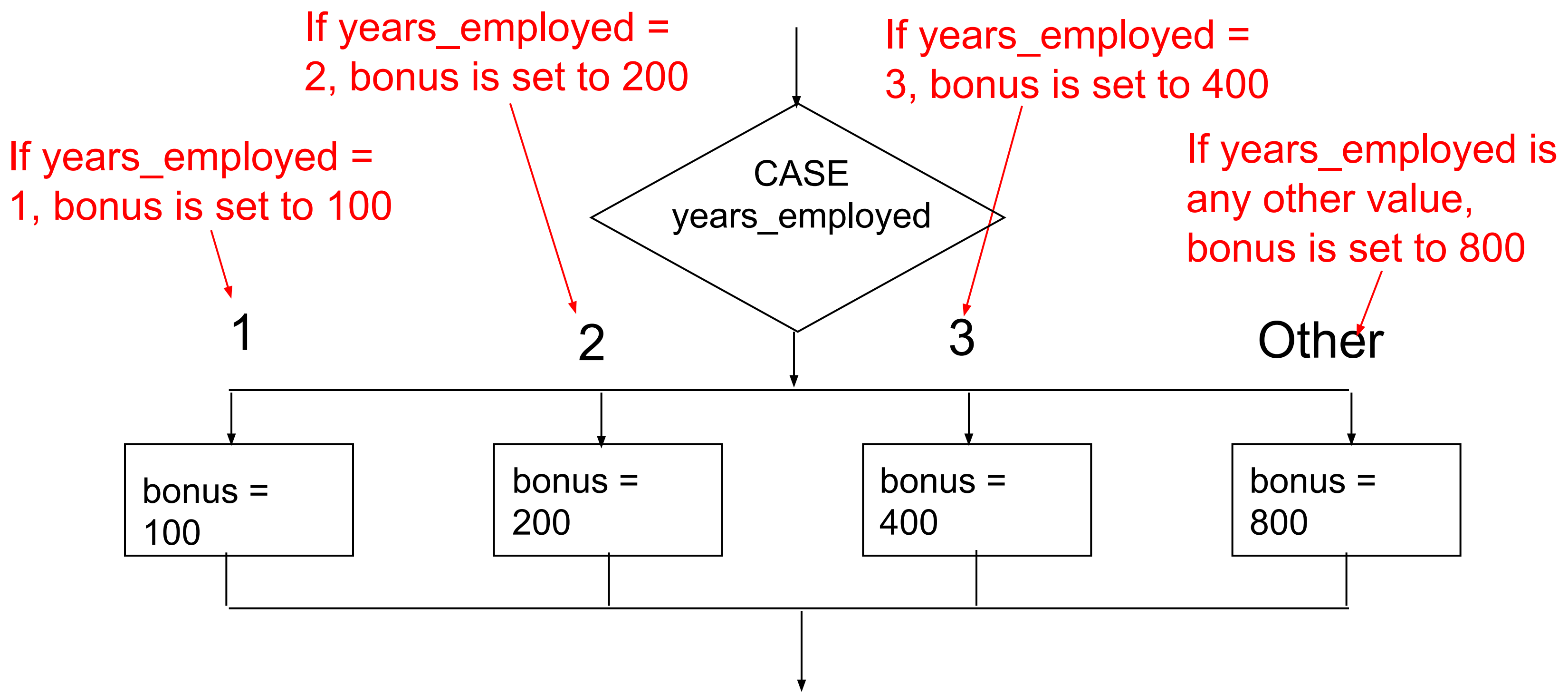
Case Structure

- One of several possible actions is taken, depending on the contents of a variable.



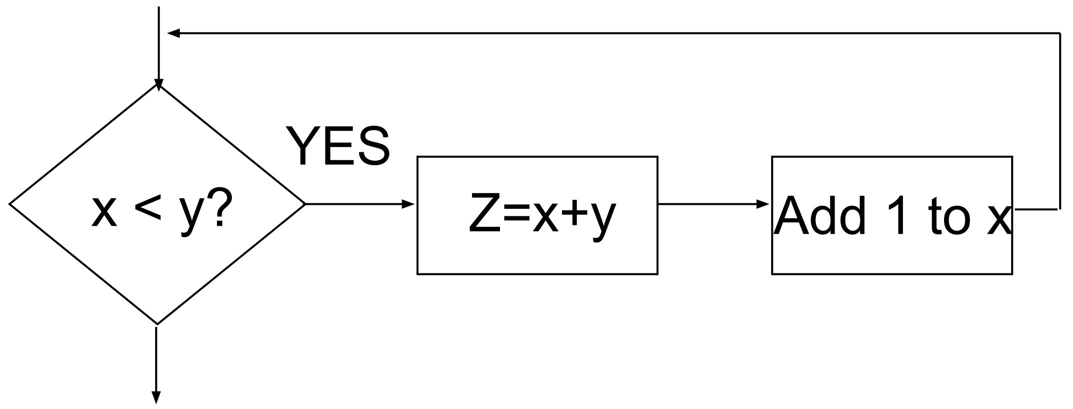
Case Structure

The structure below indicates actions to perform depending on the value in years_employed.


If years_employed = 2, bonus is set to 200
If years_employed = 3, bonus is set to 400
If years_employed = 1, bonus is set to 100
CASE years_employed
If years_employed is any other value, bonus is set to 800
1
2
3
Other
bonus = 100
bonus = 200
bonus = 400
bonus = 800

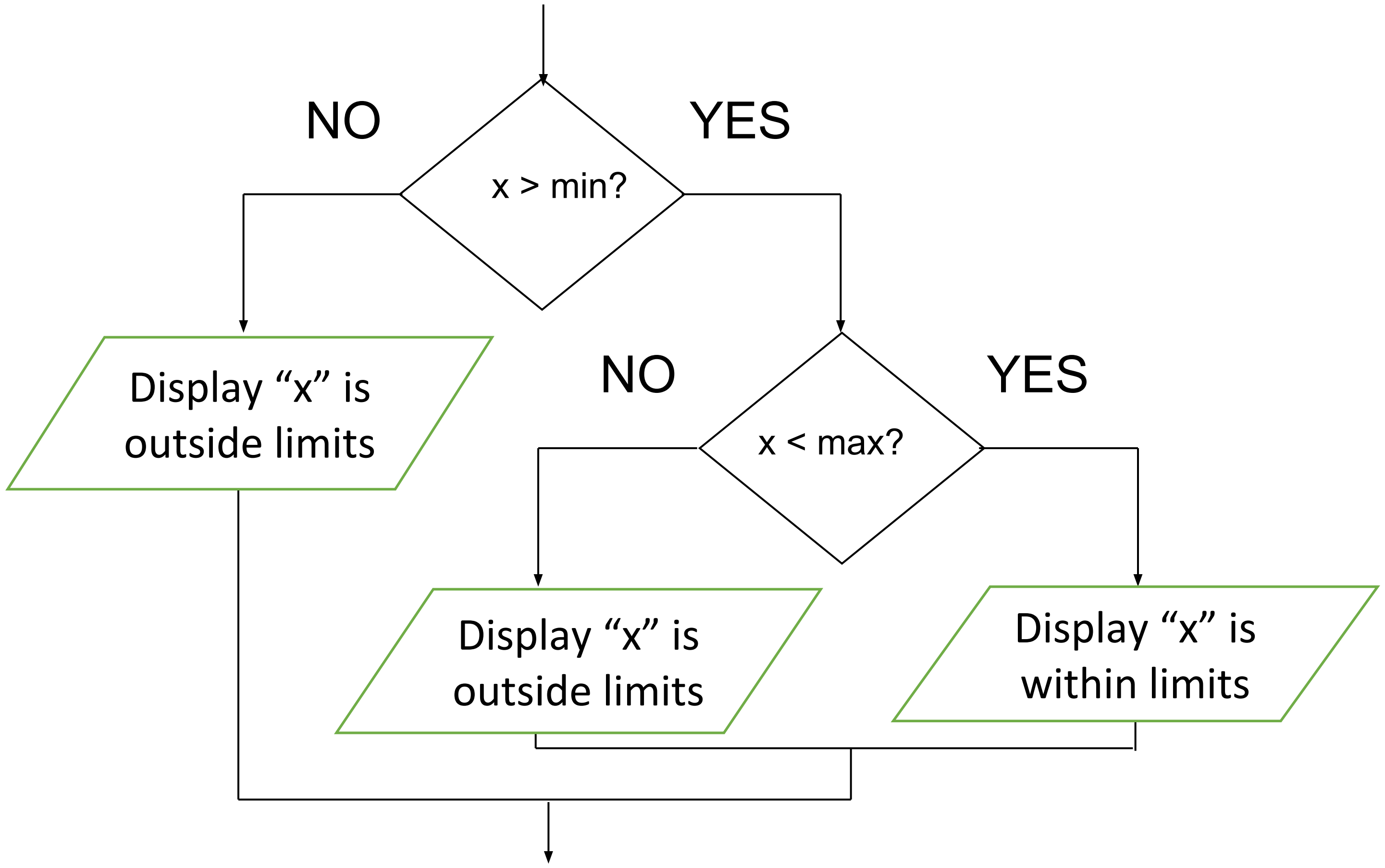
Combining Structures

- Structures are commonly combined to create more complex algorithms.
- The flowchart segment below combines a decision structure with a sequence structure.



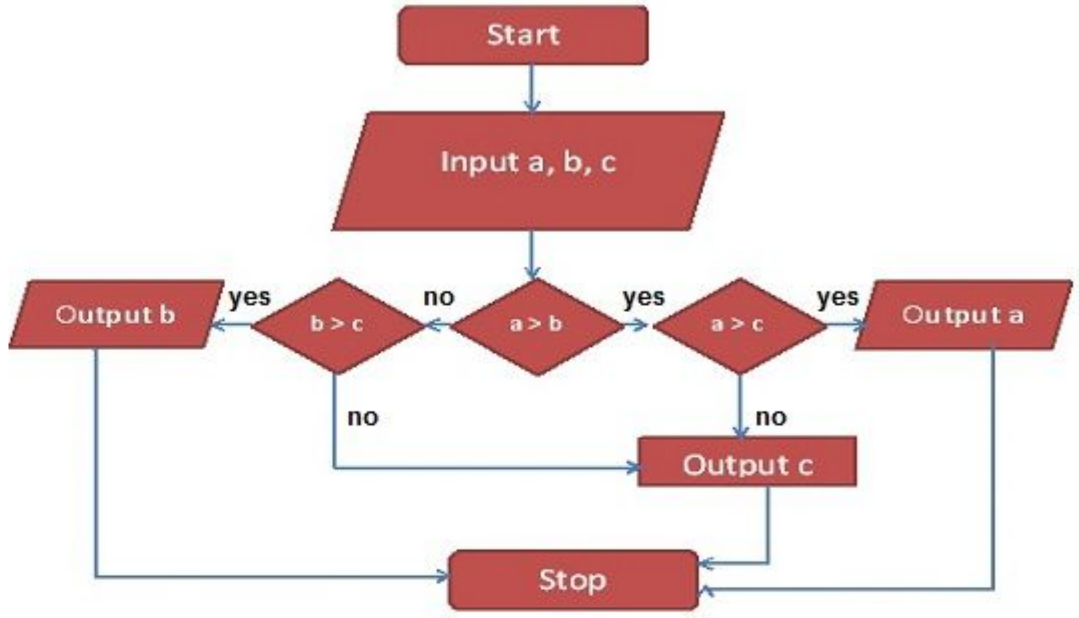
Combining Structures

- This flowchart segment shows two decision structures combined.


NO
x > min?
YES
Display "x" is outside limits
NO
x < max?
YES
Display "x" is outside limits
Display "x" is within limits

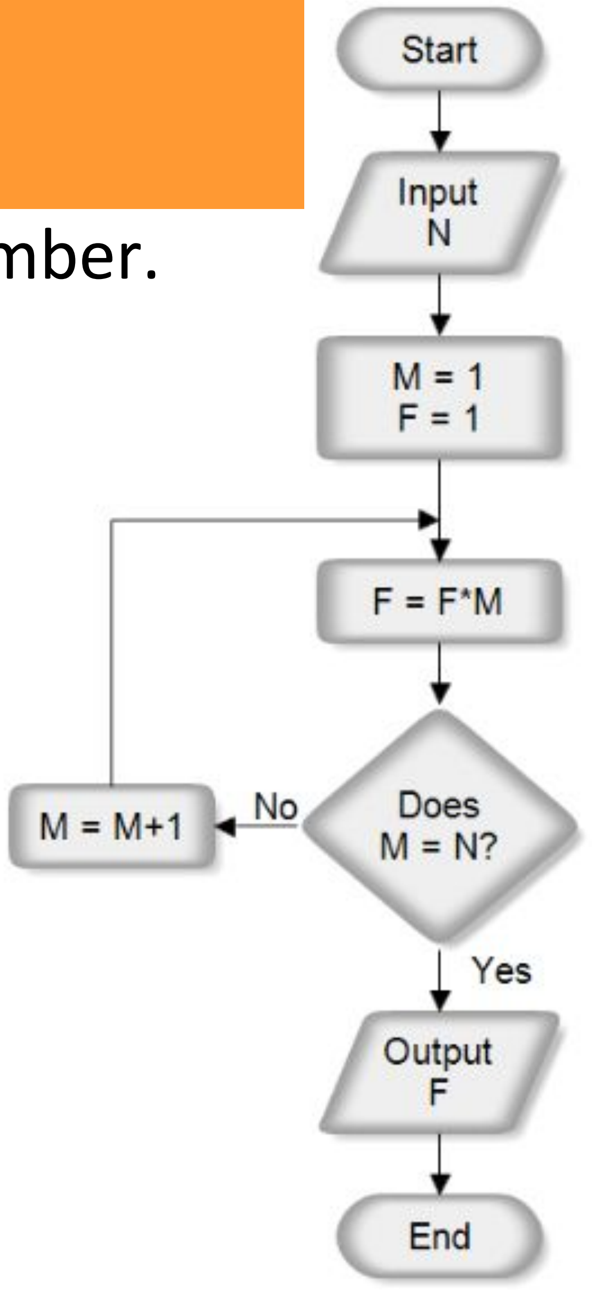
Example

- Flowchart to find largest among three numbers



Combining Structures

- Flowchart to find the factorial of a number.
- F=1*2*3*4……………………………….*N



Flowchart: Advantages

It's usually more effective to visualize something graphically that it is to describe it with words.

The following are some of the more salient reasons to use flowcharts.

.Process Documentation / Training Materials

.Workflow Management and Continuous Improvement

.Programming

.Troubleshooting ” problem solve” Guides

.Regulatory and Quality Management Requirements