## Example 1 – Adding a series of numbers

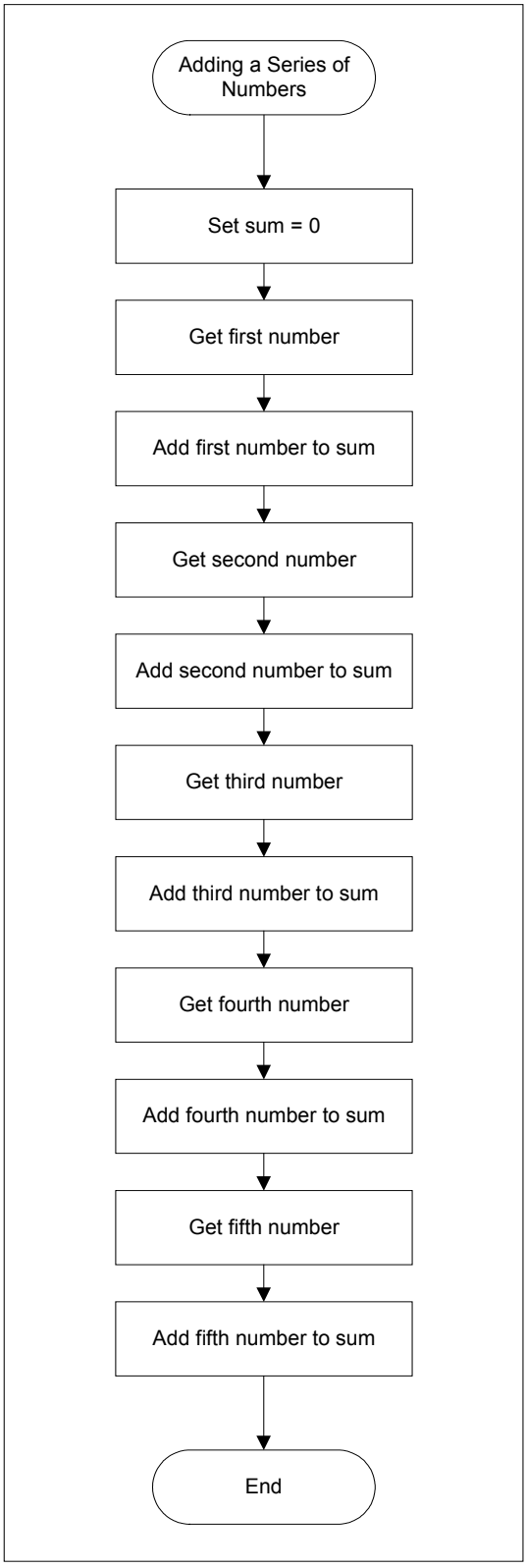
Revisiting Example 1 from the section on Algorithm Development…

Add the following set of numbers:
17, 20, 1, 22, 18

Solution steps:

1. start with the sum equal to zero
2. take the first number
3. add the first number to the sum
4. get the second number
5. add the second number to the sum
6. get the third number
7. add the third number to the sum
8. get the fourth number
9. add the fourth number to the sum
10. get the last number
11. add the last number to get the final sum

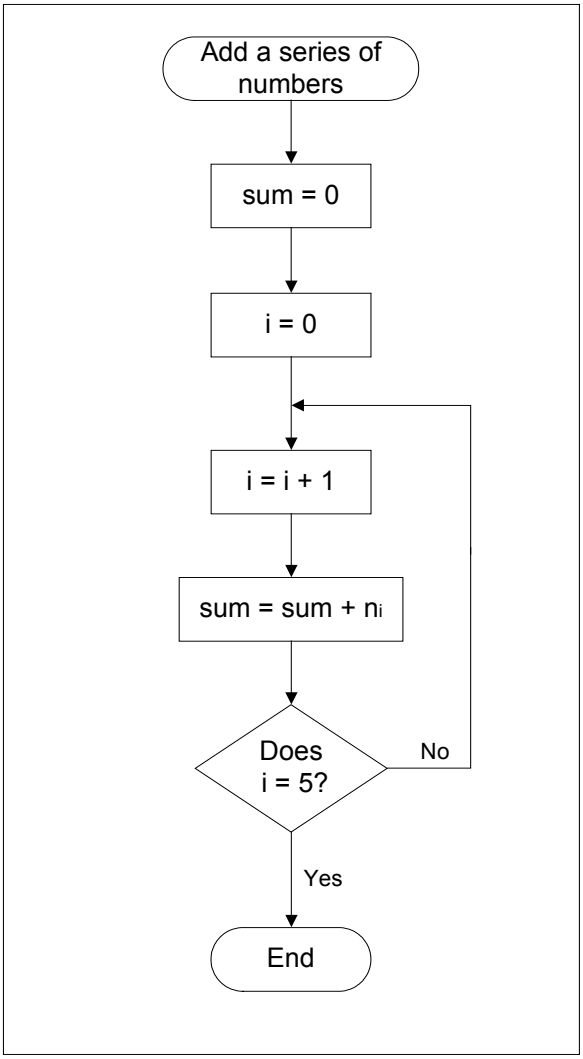
The flowchart for the above algorithm is shown at the right. Notice that the Terminal symbol at the top of the flowchart contains a description for the flowchart. The Terminal symbol at the bottom denotes the ending point of the algorithm. The Process steps show what steps must be performed. The flow lines that connect the Process symbols show the flow through the algorithm.



Now let's look at the second algorithm developed for adding the series of numbers.

Solution steps:

1. sum = 0
2. i = 0
3. i = i + 1
4. sum = sum + $n_i$
5. repeat steps 3 and 4 until i = 5




The flowchart for this algorithm is shown at the right. Again, notice that the Terminal symbol at the top of the flowchart contains a description for the flowchart. The Terminal symbol at the bottom denotes the ending point of the algorithm as well. The Process steps show what steps must be performed and the flow lines that connect the Process symbols show the flow through the algorithm. Notice the Decision symbol located at the bottom of the flow chart. The decision checks to see if the value of i is equal to five (see step number 5 in the algorithm). If the value of i is not equal to five, then the program flow goes back to step number 3, repeats steps 3 & 4, and then re-tests the value of i. This process repeats until the value of i equals five. Once the value of i is equal to five, the algorithm ends.

The process of repeating steps 3 & 4 through the use of a decision is a part of the foundation for structured programming. This gives the program power and flexibility. Simply changing the test value used in the Decision can change the algorithm to add any number of numbers. In order to change the first algorithm to add additional numbers, more steps must be added to the bottom of the sequence increasing the program size without any flexibility.