Pseudocode

- From algorithms to pseudocode
  - **Pseudocode**: a set of human-readable statements for delineating the steps of an algorithm
  - Pseudocode is not a program you can run, but provides more detail than an algorithm, easing the transition from idea to program code
  - Used to help brainstorm before you code
  - Easy to write and help determine logic of a problem
  - Non-technical team members can understand
- Pseudocode guidelines
  - Some keywords and style conventions are derived from programming languages
  - In pseudocode, you work with **variables**, which hold values that might change during program execution, and computations

Pseudocode Examples: Output

- Python
  - **print(“Game Over”)**
- Pseudocode
  - **Display ‘Game Over’**

Pseudocode Examples: Retrieve Information

- Python
  - **Name = input(“Enter name:”)**
- Pseudocode
  - **Prompt for name**

    Or
  - **Get name**

Pseudocode Examples: Calculations

- Python
  - **avg = point / num_students**
- Pseudocode
  - **DIVIDE num_students by points**

Pseudocode Examples: Assigning Variables

- Python
  - **cart_items = 0**
- Pseudocode
  - **SET cart_items to 0**

Pseudocode Examples: IF Statements

- Python
  - **if attendance == True:**
    - **attendance_points += 1**
  - **Else:**
    - **print(“You missed class”)**

- Pseudocode
  - **IF attendance is True THEN**
    - **Add 1 to attendance_points**
  - **Else**
    - **Display “You missed class”**
  - **EndIf**

Pseudocode Examples: IF Statements

- Python: Sample restaurant rating code

```
rating = int(input("Enter rating:"))
if rating >= 7:
    print("Great Work!")
else:
    print("Let's try again")
```

Pseudocode Examples: IF Statements

- Pseudocode: Sample restaurant rating code

  - **PROMPT for rating**
  - **Get the rating and make it a number**
  - **IF rating is greater than or equal to 7**
    - **Display “Great Work!”**
  - **Else**
    - **Display “Let’s try again”**
  - **EndiF**

Pseudocode

- Pseudocode guidelines (continued)
  - Steps for writing pseudocode:
    - Write down the algorithm
    - Identify and name variables
    - Write statements that declare and initialize variables
    - Work through each step of the algorithm, writing detailed pseudocode steps
    - Identify computations and write them as mathematical expressions, using variables
    - Identify decision structures and write them in `if..then` and `otherwise` format
    - Identify repetition structures and write them using `repeat x times` or `repeat until`
    - Use indents for multiline decision and repetition structures
    - Use blank lines as whitespace to set off decision and repetition structures

Pseudocode Example

```
1
2  BEGIN
3
4    NUMERIC nNum1,nNum2
5    DISPLAY "ENTER THE FIRST NUMBER : "
6    INPUT nNum1
7
8    DISPLAY "ENTER THE SECOND NUMBER : "
9    INPUT nNum2
10
11   IF nNum1 > nNum2
12     DISPLAY nNum1 + " is larger than "+ nNum2
13   ELSE
14    DISPLAY nNum2 + " is larger than " + nNum1
15
16 END
17
```

1. `BEGIN` is a marker that indicates the start of the program.
2. `NUMERIC nNum1,nNum2` declares two variables, nNum1 and nNum2, as numeric data types.
3. `DISPLAY "ENTER THE FIRST NUMBER : "` prints the string "ENTER THE FIRST NUMBER :" to the screen.
4. `INPUT nNum1` waits for the user to enter a value, which is then stored in the variable nNum1.
5. `DISPLAY "ENTER THE SECOND NUMBER : "` prints the string "ENTER THE SECOND NUMBER :" to the screen.
6. `INPUT nNum2` waits for the user to enter a value, which is then stored in the variable nNum2.
7. `IF nNum1 > nNum2` checks if the value stored in nNum1 is greater than the value stored in nNum2.
8. If nNum1 > nNum2 is true, then the next line is executed, `DISPLAY nNum1 + " is larger than "+ nNum2` which will print the value of nNum1 and a string that says " is larger than " and value of nNum2.
9. If the value of nNum1 is not greater than nNum2, the next line is executed: `DISPLAY nNum2 + " is larger than " + nNum1` which will print the value of nNum2 and a string that says " is larger than " and value of nNum1.
10. `END` is a marker that indicates the end of the program.

Pseudocode Example

```
1
2 begin
3     numeric nNum1,nNum2,nSum
4     display "ENTER THE FIRST NUMBER : "
5     accept nNum1
6     display "ENTER THE SECOND NUMBER : "
7     accept nNum2
8     compute nSum=nNum1+nNum2
9     display "SUM OF THESE NUMBER : " nSum
10 end
11
```

1. `BEGIN` is a marker that indicates the start of the program.
2. `NUMERIC nNum1,nNum2,nSum` declares three variables, nNum1, nNum2 and nSum, as numeric data types.
3. `DISPLAY "ENTER THE FIRST NUMBER : "` prints the string "ENTER THE FIRST NUMBER :" to the screen.
4. `ACCEPT nNum1` waits for the user to enter a value, which is then stored in the variable nNum1.
5. `DISPLAY "ENTER THE SECOND NUMBER : "` prints the string "ENTER THE SECOND NUMBER :" to the screen.
6. `ACCEPT nNum2` waits for the user to enter a value, which is then stored in the variable nNum2.
7. `COMPUTE nSum=nNum1+nNum2` calculates the sum of the two numbers stored in nNum1 and nNum2, and assigns the result to the variable nSum.
8. `DISPLAY "SUM OF THESE NUMBER : " nSum` prints the string "SUM OF THESE NUMBER :" and the value of the nSum variable.
9. `END` is a marker that indicates the end of the program.

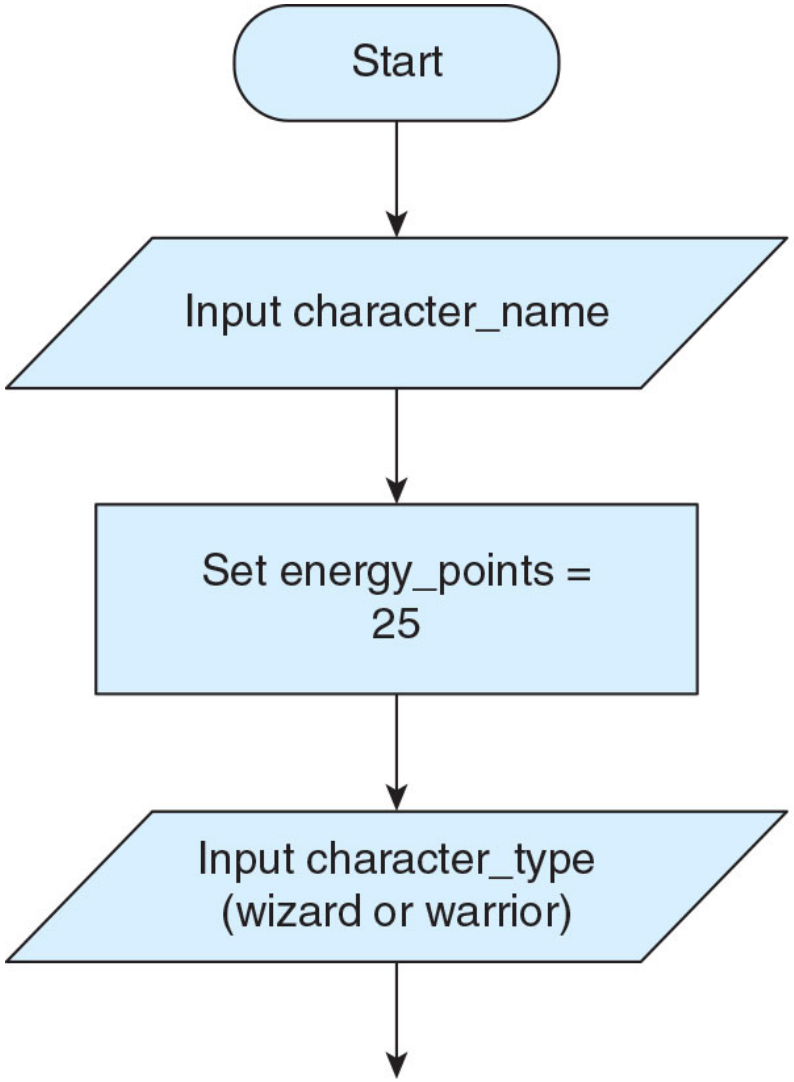
Flowcharts

- Flowchart basics
  - **Flowchart**: a diagram that represents the sequence and flow of steps in an algorithm
  - May be used instead of or along with pseudocode
  - Use a standard set of shapes that are connected by flowline arrows
- Drawing flowcharts
  - Flowcharts begin and end with a **terminator shape**
  - The first shape contains the word “Start”
  - Additional shapes are stacked vertically to indicate sequential program flow
  - Decision control structures use a diamond shape and branching arrows for decisions
  - Repetition control structures are represented using arrows that point upward to an earlier point in the sequence

Flowcharts

Shape	Name	Purpose
	Terminator	Represents the start or end of the algorithm
	Process	Indicates a mathematical or logical operation
	Decision	Represents a decision point that branches to different sets of steps
	Data	Represents data input or output
	Connector	Indicates a connection between two separate sections of a flowchart

Flowcharts



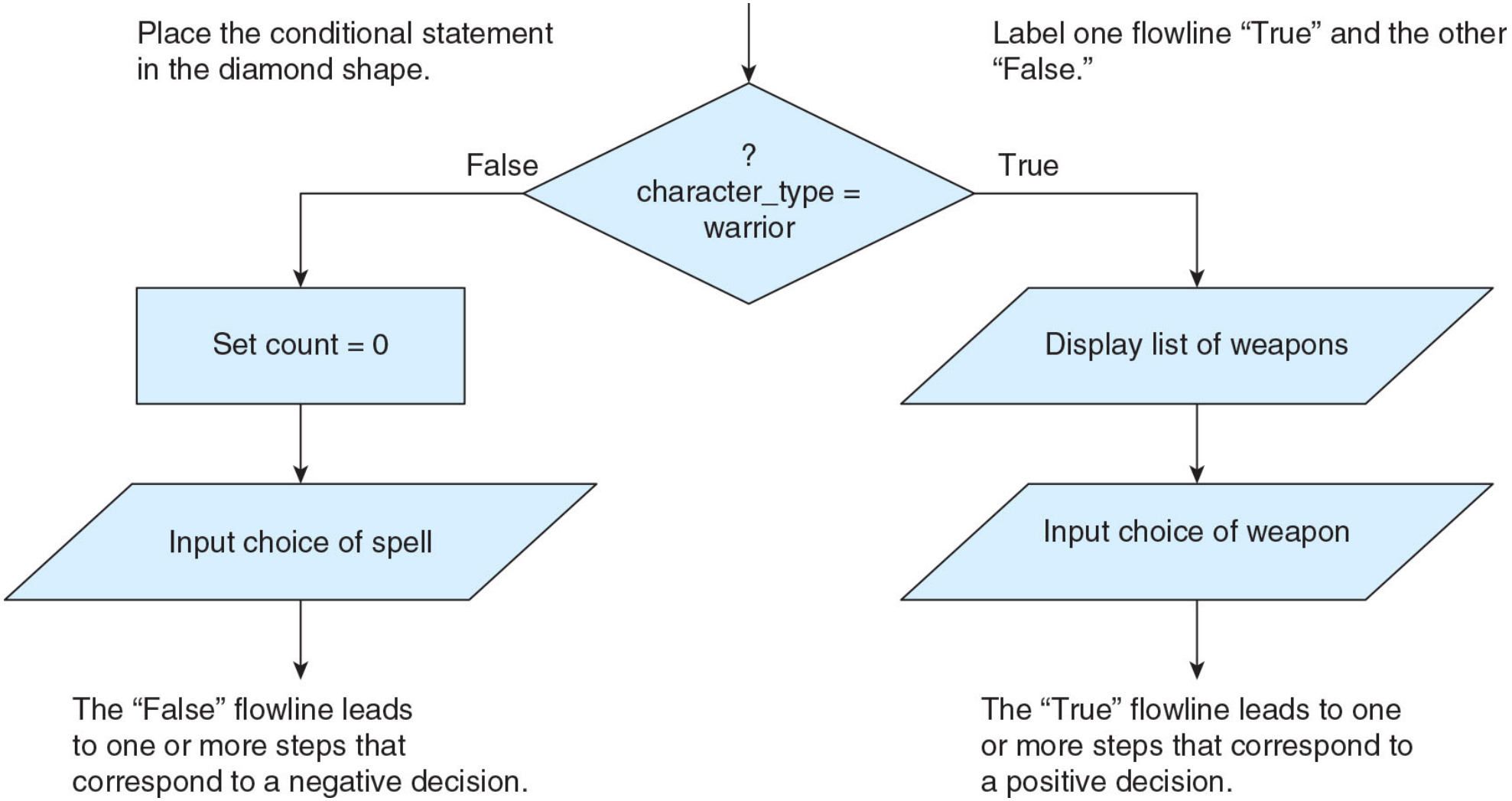
Begin the flowchart with a terminating symbol containing the word “Start.”

Use a parallelogram for steps that collect input from the user.

Use a rectangle for calculations.

Connect shapes with flow arrows.

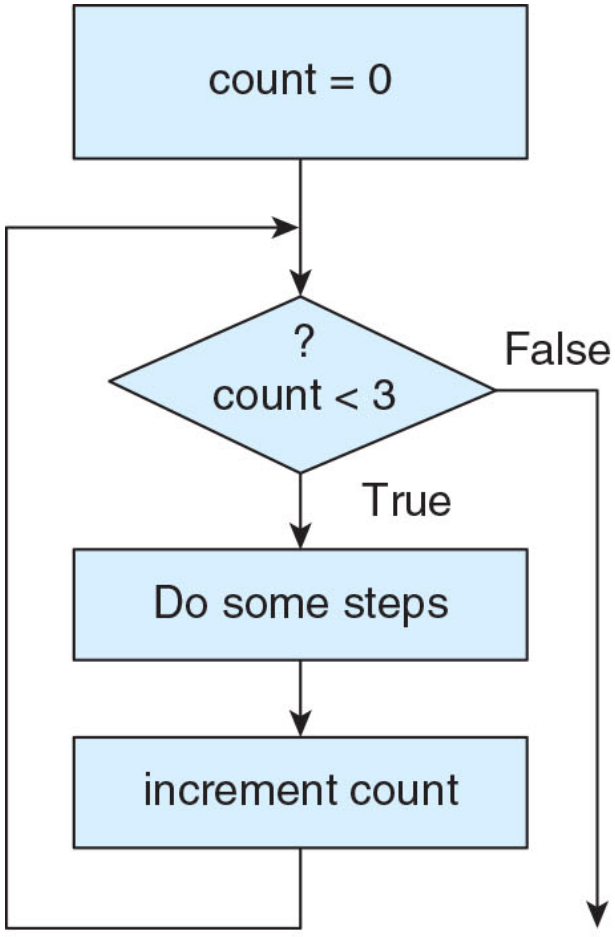
Flowcharts



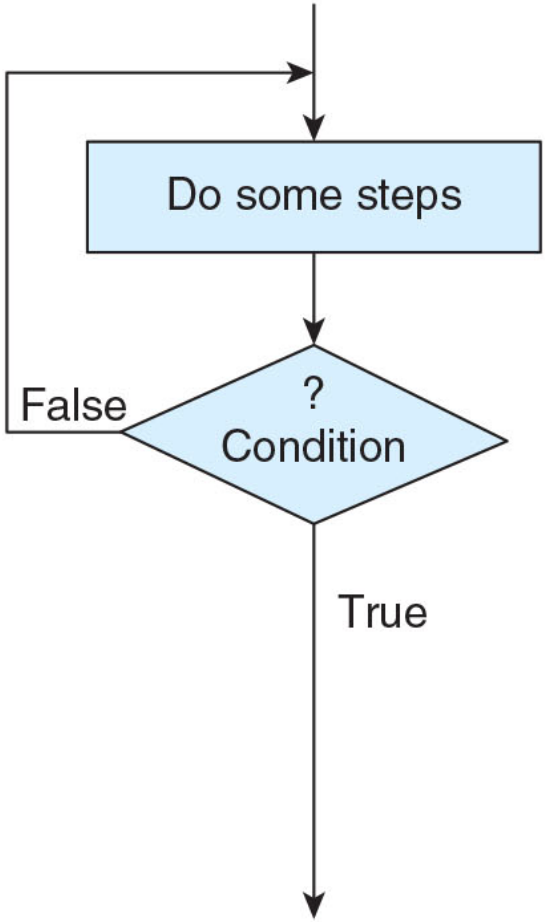
Flowcharts

- Flowchart tools
  - You can use diagramming software or an online diagramming app to create flowcharts
  - Apps allow you to drag and drop shapes, and typically to drag connecting arrows between shapes
  - Check your completed flowchart:
    - Each shape excluding the Start and End terminators should have at least one input and one output
    - All flow arrows should point in the correct direction to indicate the sequence of steps

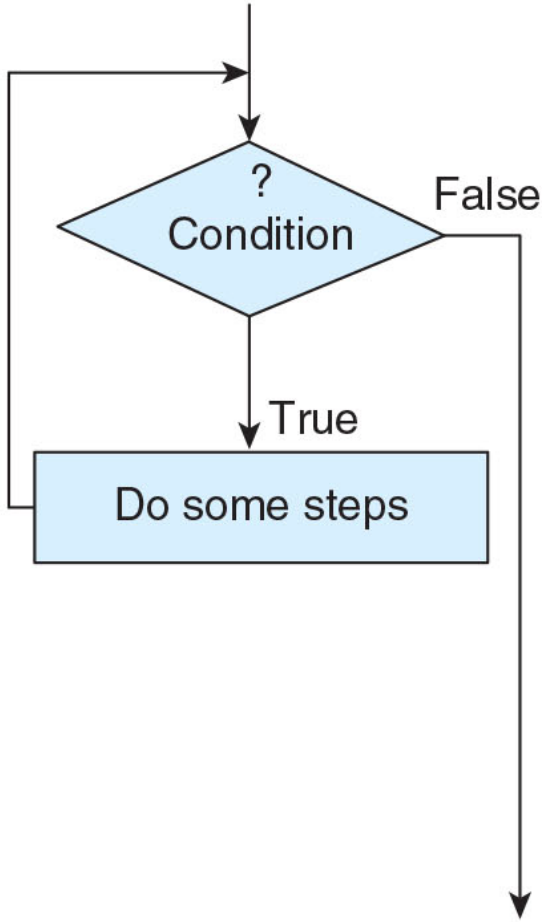
Flowcharts: repetition control structures



for..next



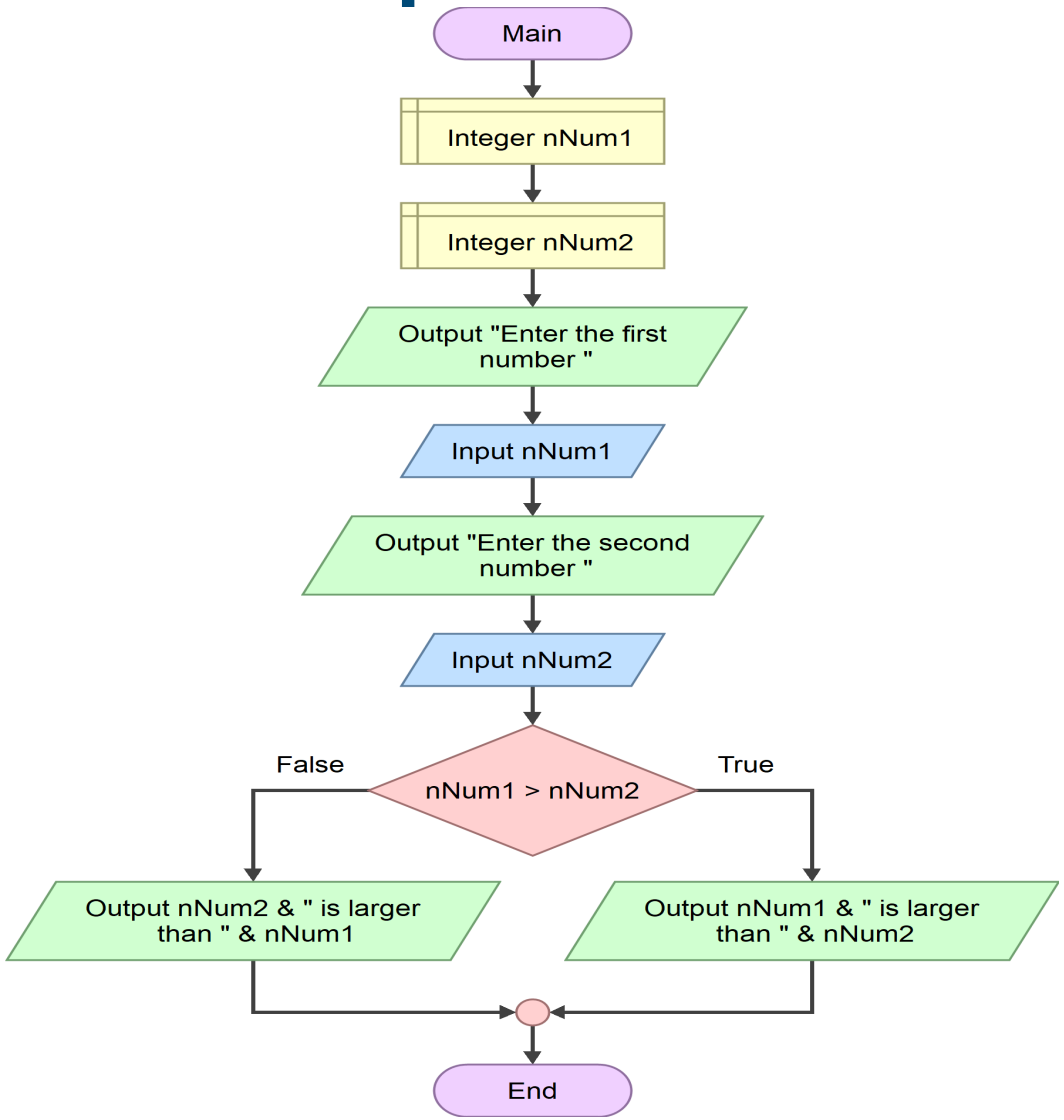
repeat..until



while..do

Pseudocode/Flowchart Example

Shape	Name	Purpose
	Terminator	Represents the start or end of the algorithm
	Process	Indicates a mathematical or logical operation
	Decision	Represents a decision point that branches to different sets of steps
	Data	Represents data input or output
	Connector	Indicates a connection between two separate sections of a flowchart



NUMERIC nNum1,nNum2 declares two variables, nNum1 and nNum2, as numeric data types

Pseudocode/Flowchart Example

```
BEGIN

  NUMERIC nNum1,nNum2
  DISPLAY "ENTER THE FIRST NUMBER : "
  INPUT nNum1

  DISPLAY "ENTER THE SECOND NUMBER : "
  INPUT nNum2

  IF nNum1 > nNum2
    DISPLAY nNum1 + " is larger than "+ nNum2
  ELSE
   DISPLAY nNum2 + " is larger than " + nNum1

END
```