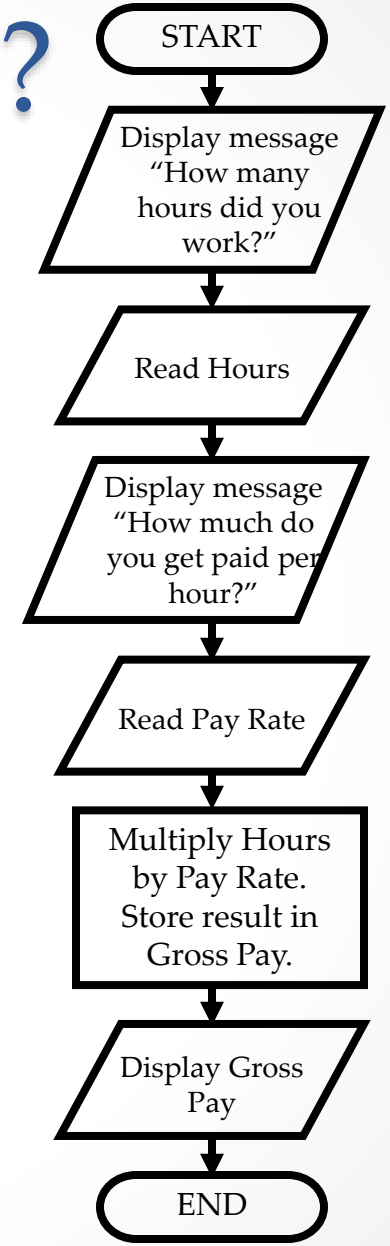
Introduction to Computers
Lab 8
First Year (2017– 2018)

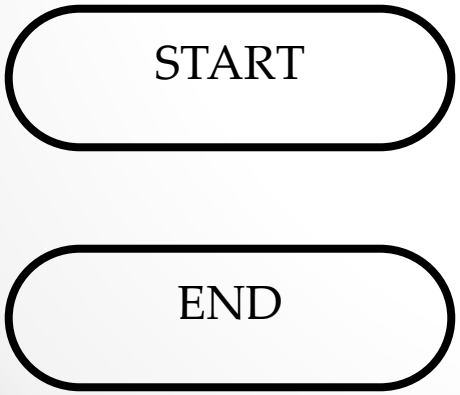
What is a Flowchart?

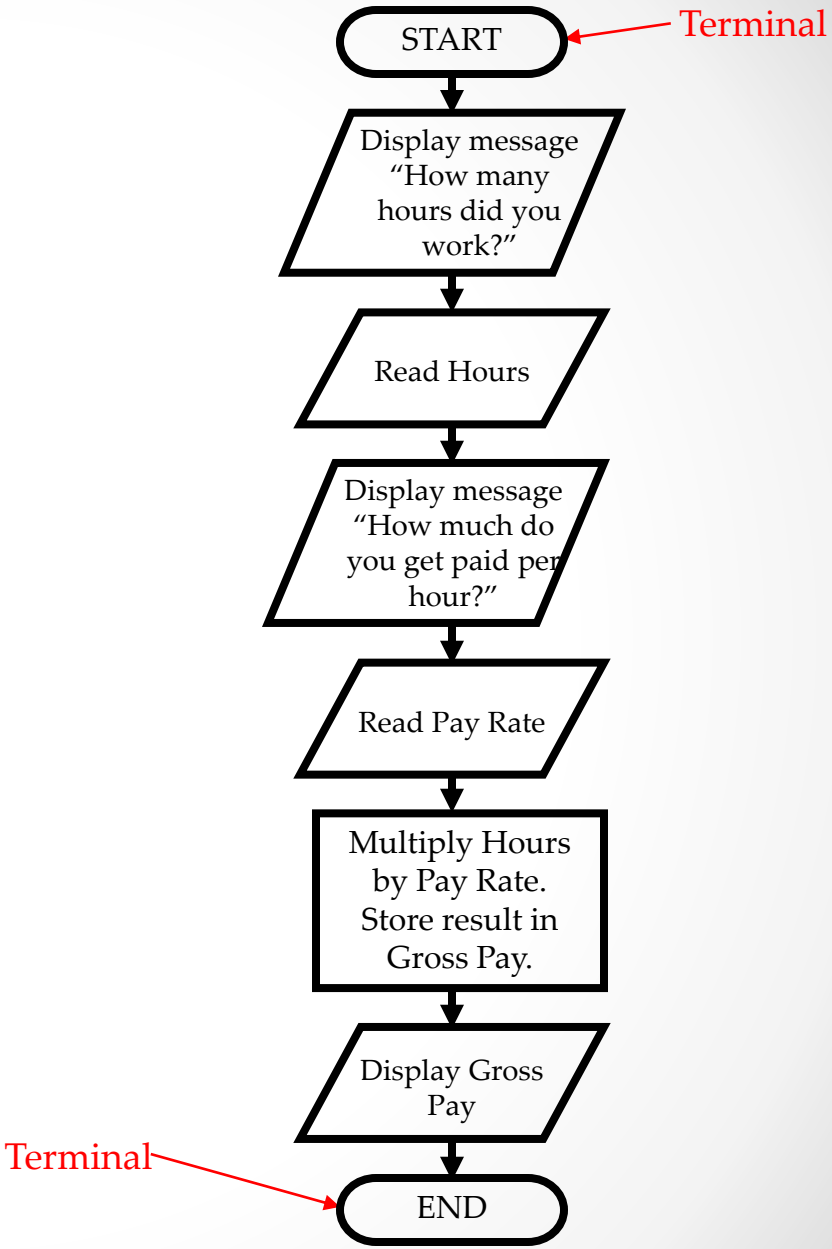
- A flowchart is a diagram that depicts the "flow" of a program.
- The figure shown here is a flowchart for the pay-calculating.



Basic Flowchart Symbols

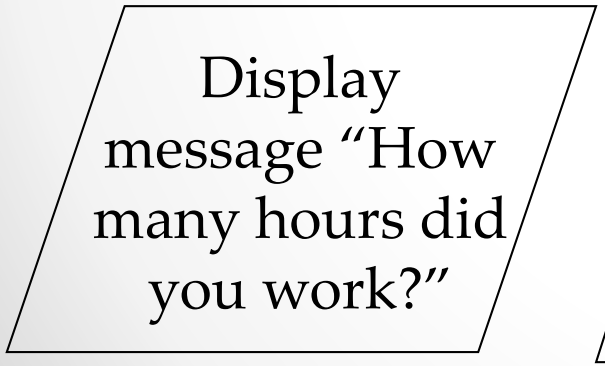
- Terminals
  - represented by rounded rectangles
  - indicate a starting or ending point

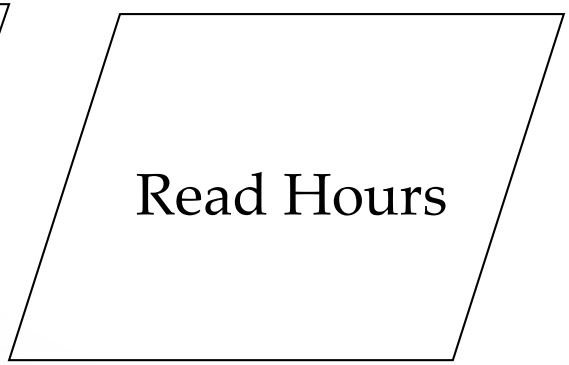


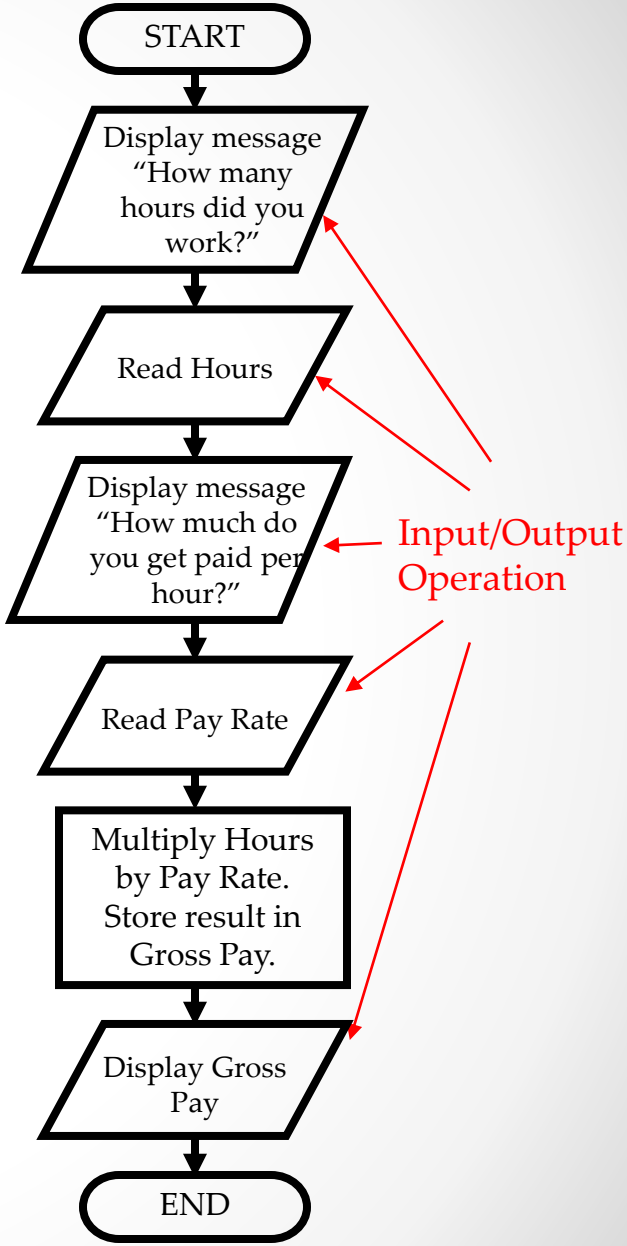


Basic Flowchart Symbols

- Input / Output Operations
  - represented by parallelograms
  - indicate an input or output operation



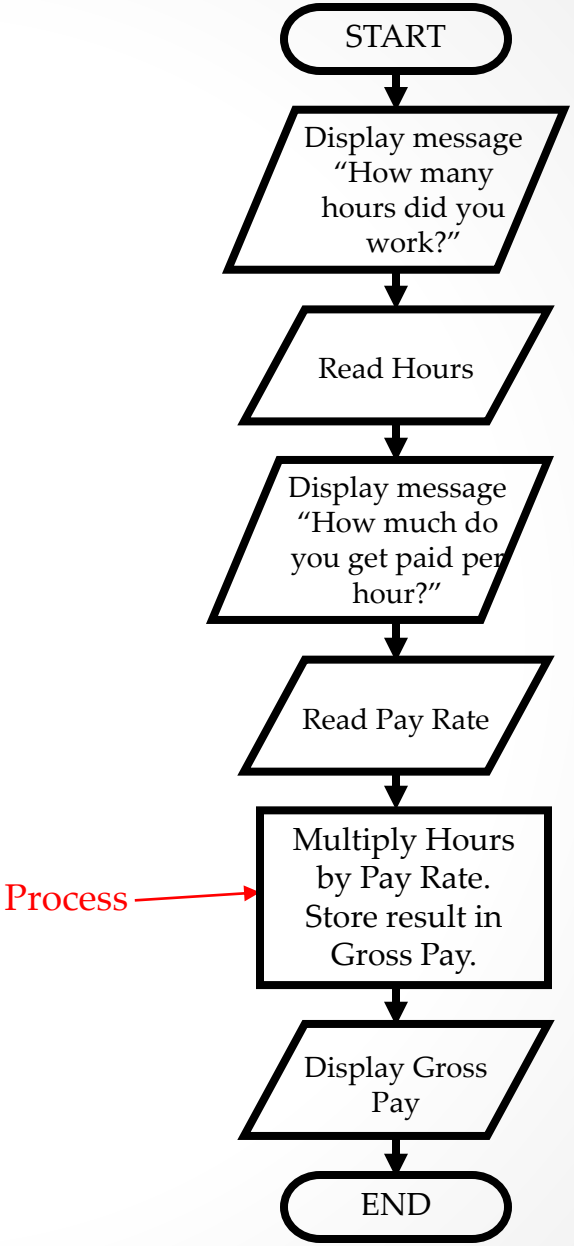




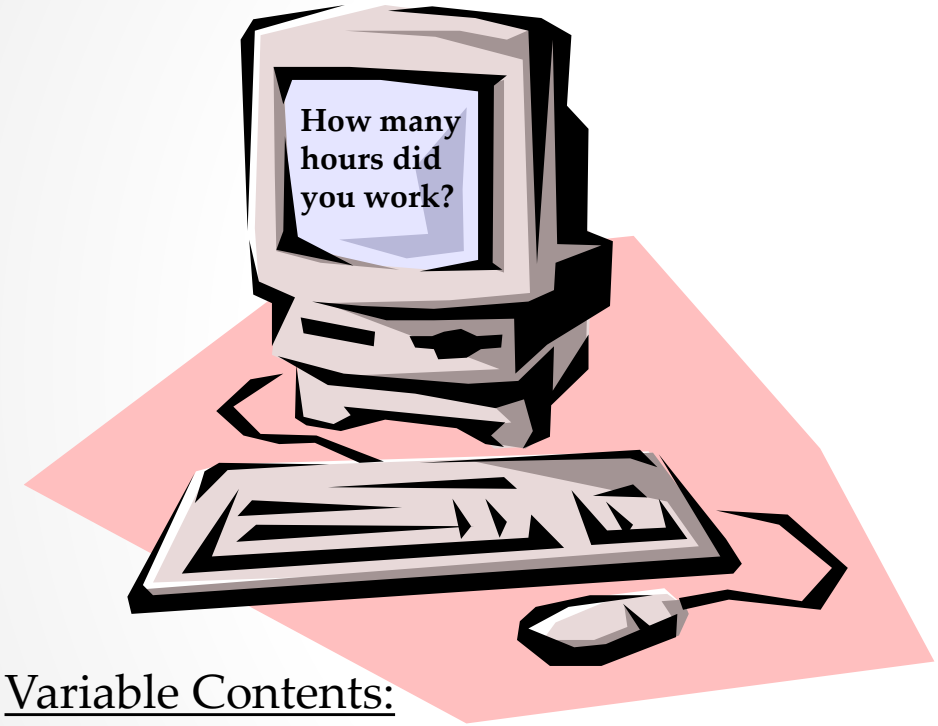
Basic Flowchart Symbols

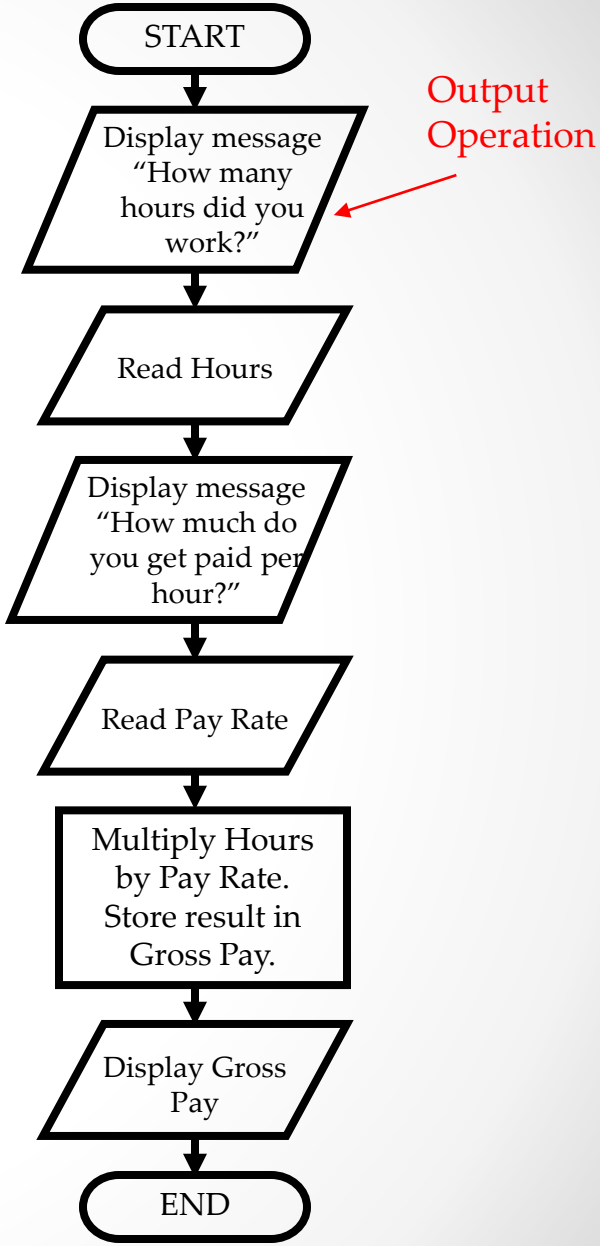
- Processes
  - represented by rectangles
  - indicates a process such as a mathematical computation or variable assignment

Multiply Hours by Pay Rate. Store result in Gross Pay.



Stepping Through the Flowchart





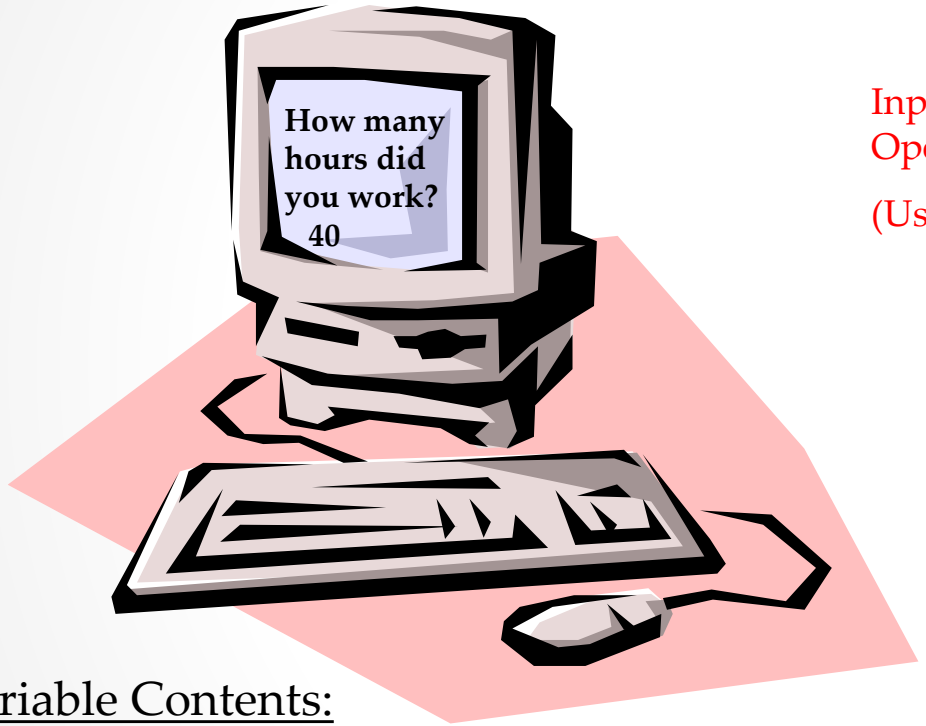
<u>Variable Contents:</u>

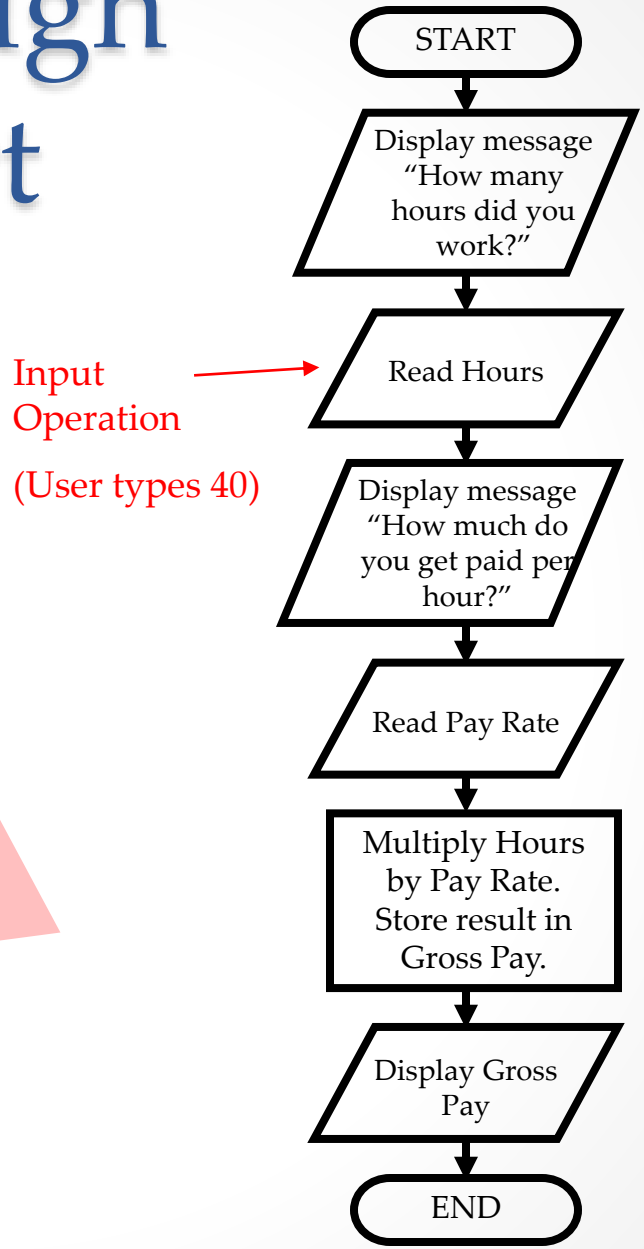
Hours: ?

Pay Rate: ?

Gross Pay: ?

Stepping Through the Flowchart

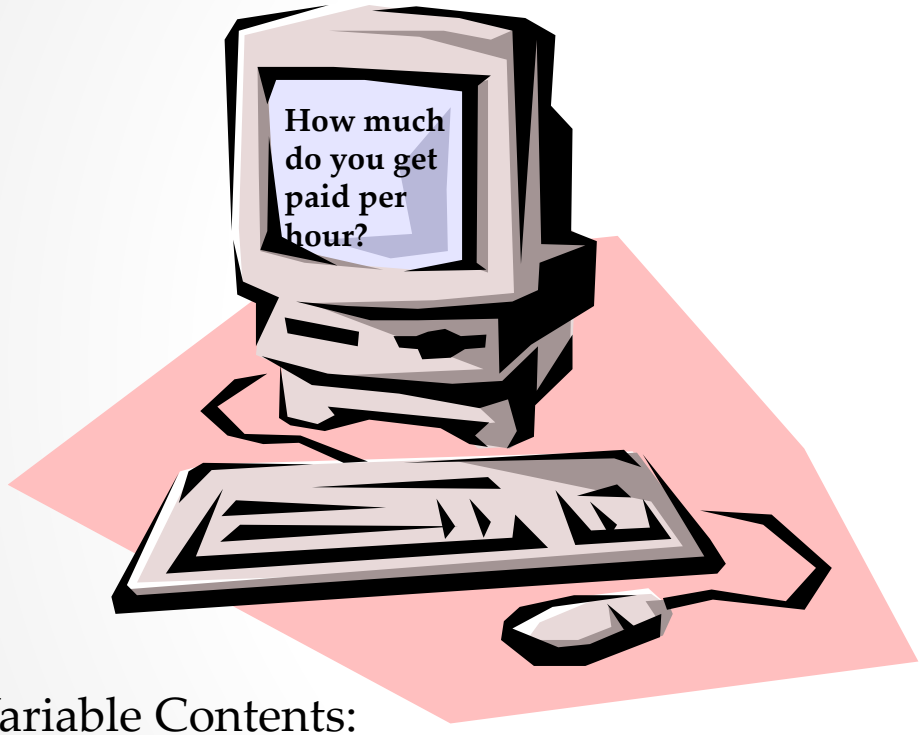


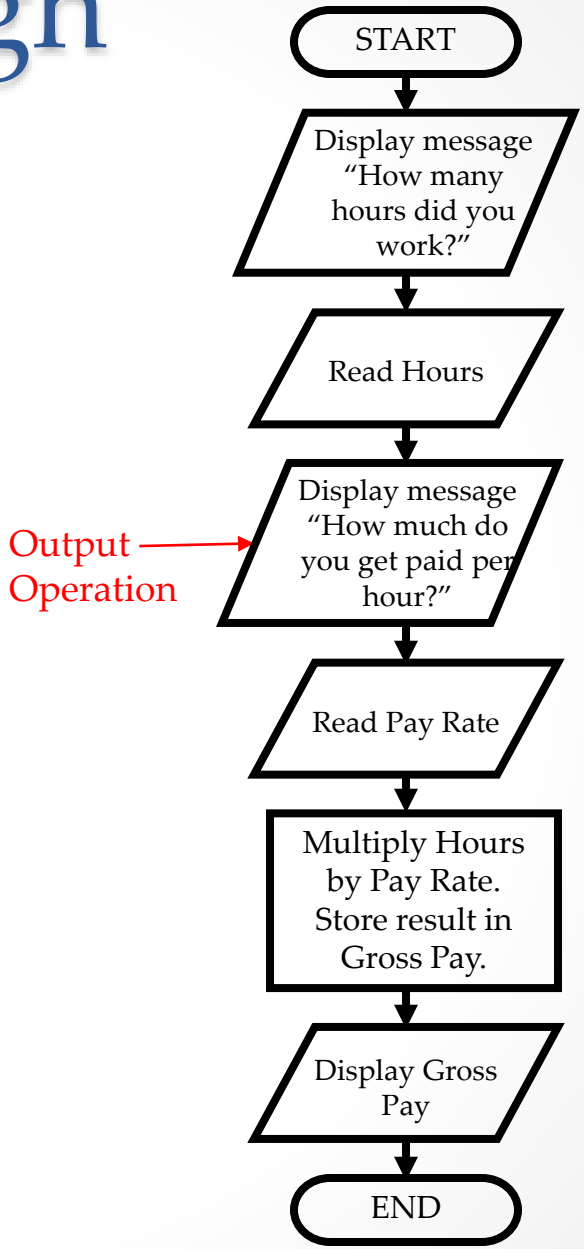


<u>Variable Contents:</u>

Hours: 40
Pay Rate: ?
Gross Pay: ?

Stepping Through the Flowchart

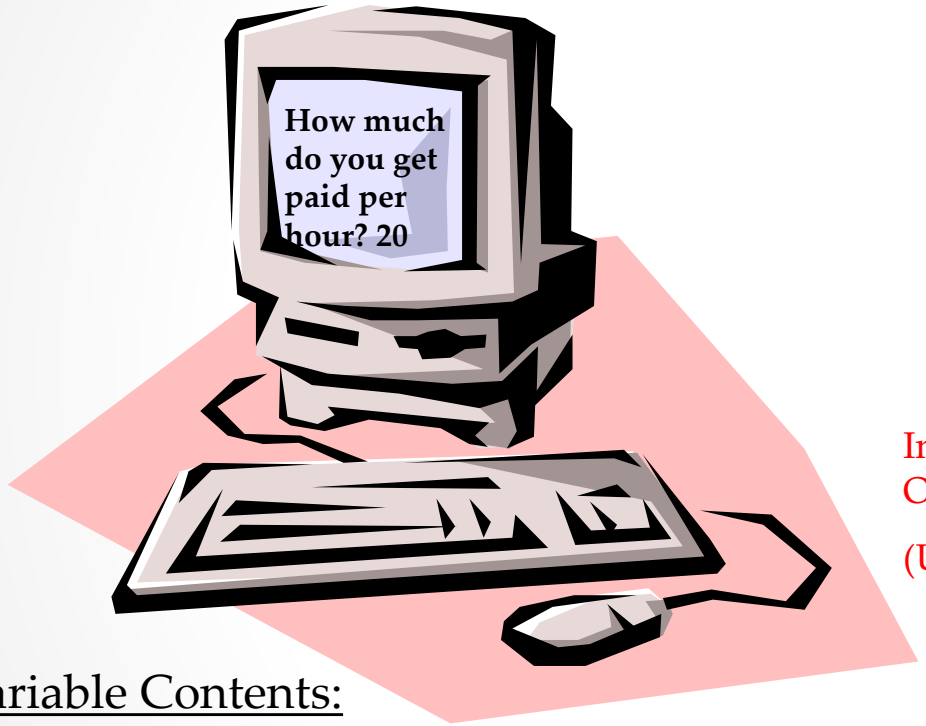


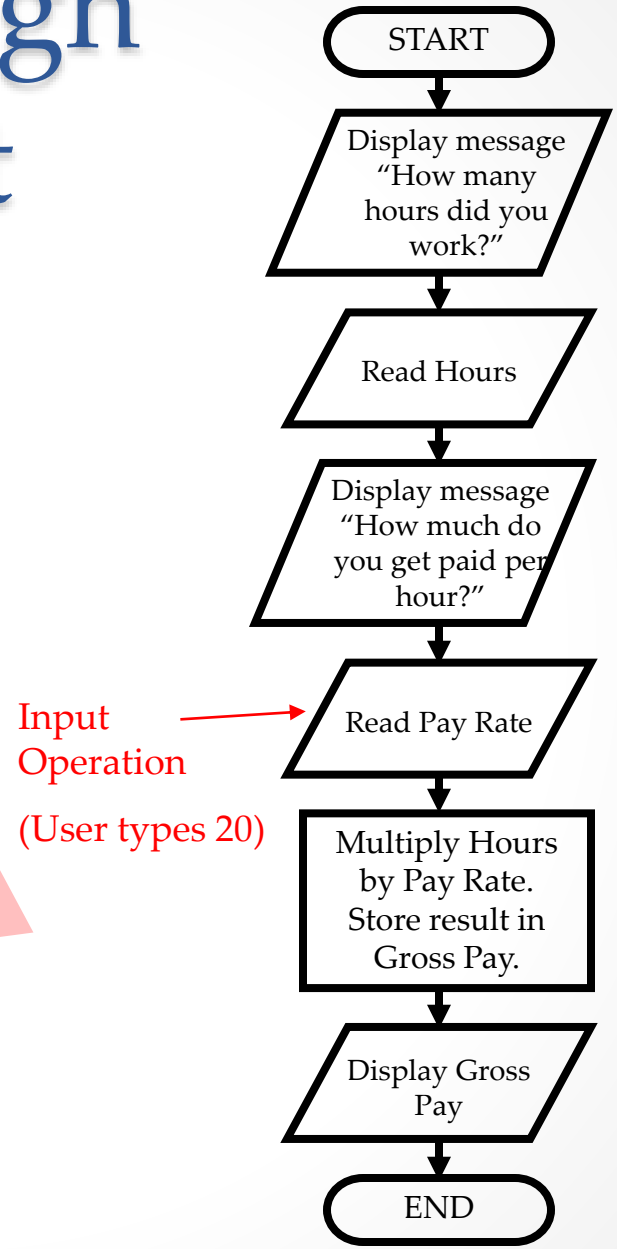


<u>Variable Contents:</u>

Hours: 40
Pay Rate: ?
Gross Pay: ?

Stepping Through the Flowchart

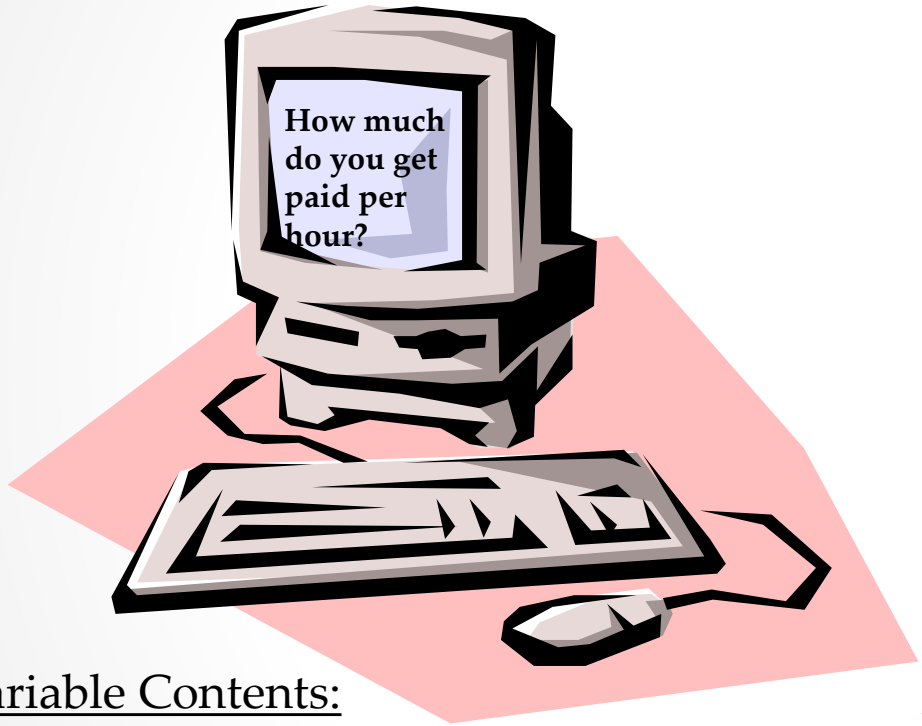


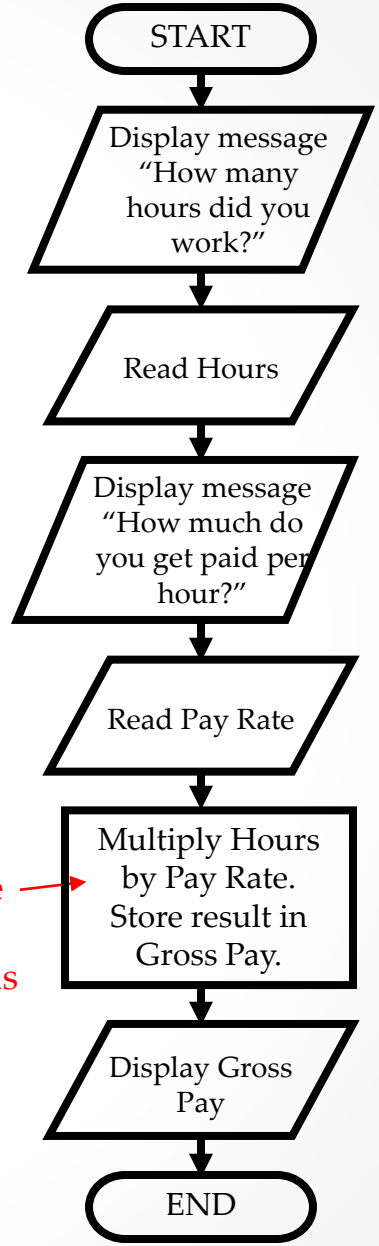


<u>Variable Contents:</u>

Hours: 40
Pay Rate: 20
Gross Pay: ?

Stepping Through the Flowchart



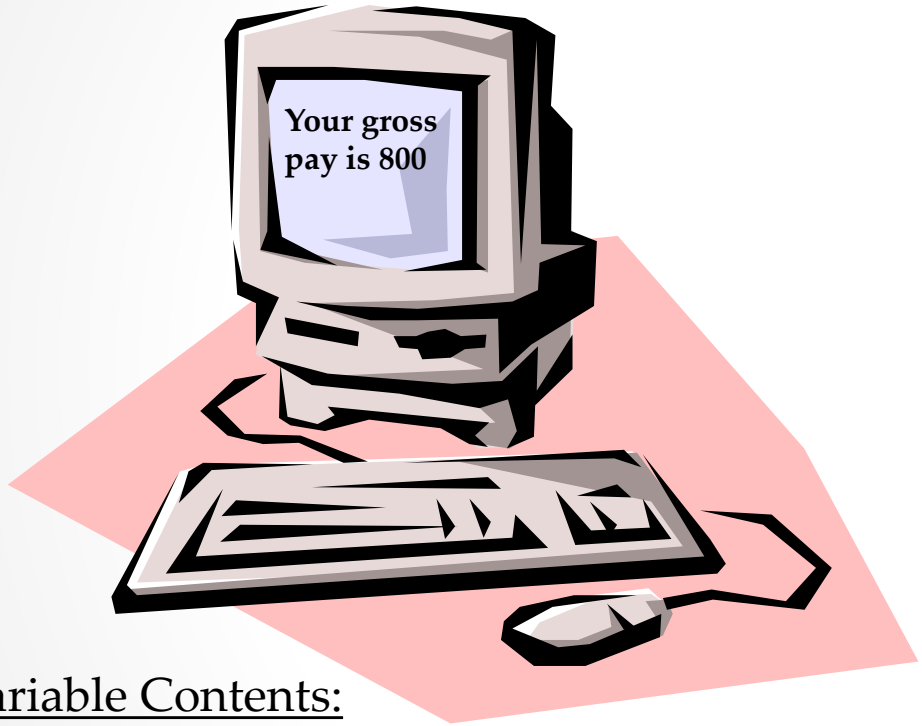


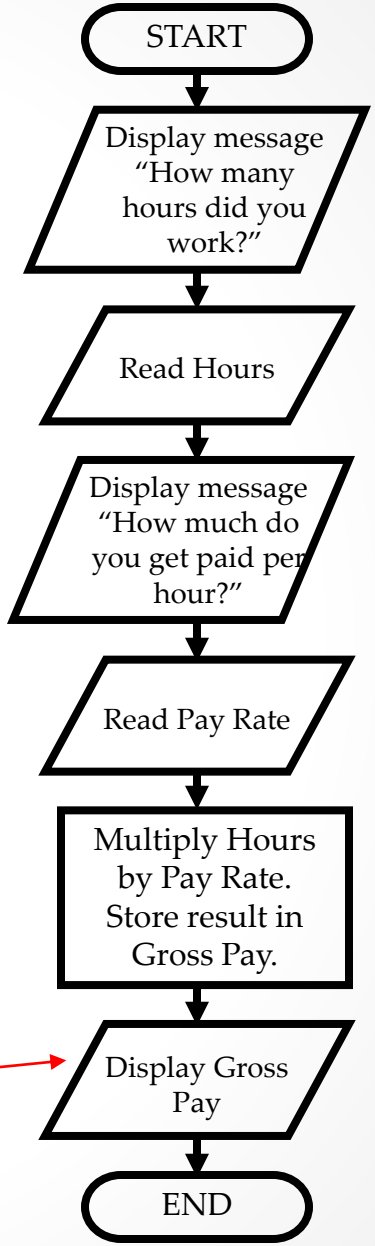
Process: The product of 40 times 20 is stored in Gross Pay

<u>Variable Contents:</u>

Hours: 40
Pay Rate: 20
Gross Pay: 800

Stepping Through the Flowchart





Output Operation

<u>Variable Contents:</u>

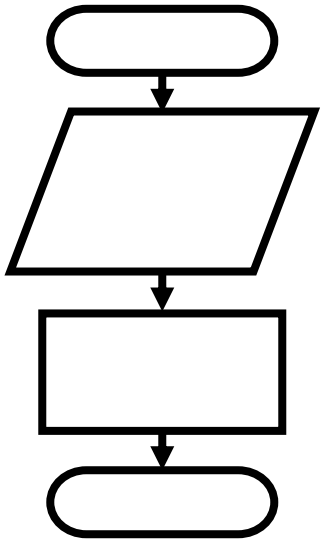
Hours: 40
Pay Rate: 20
Gross Pay: 800

Four Flowchart Structures

- Sequence
- Decision
- Repetition
- Case

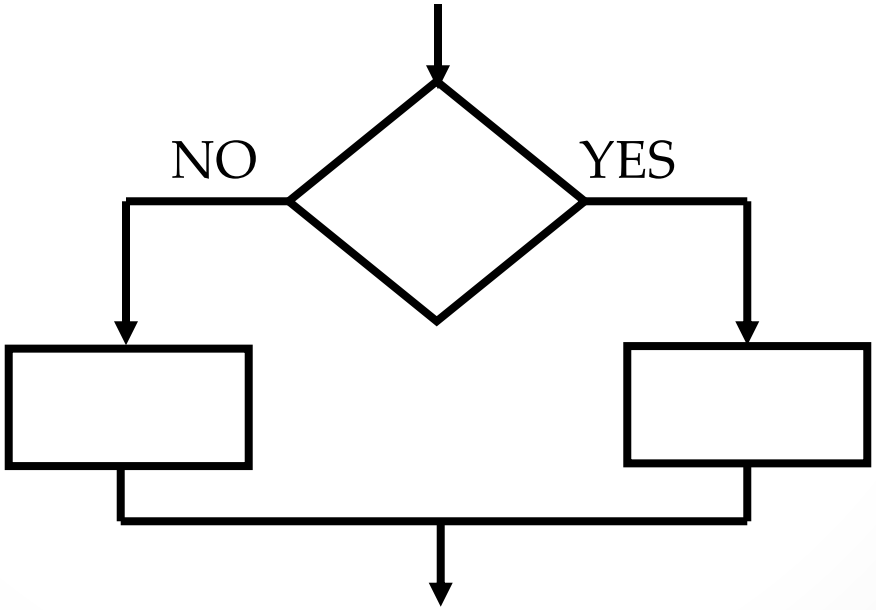
Sequence Structure

- A series of actions are performed in sequence
- The pay-calculating example was a sequence flowchart.



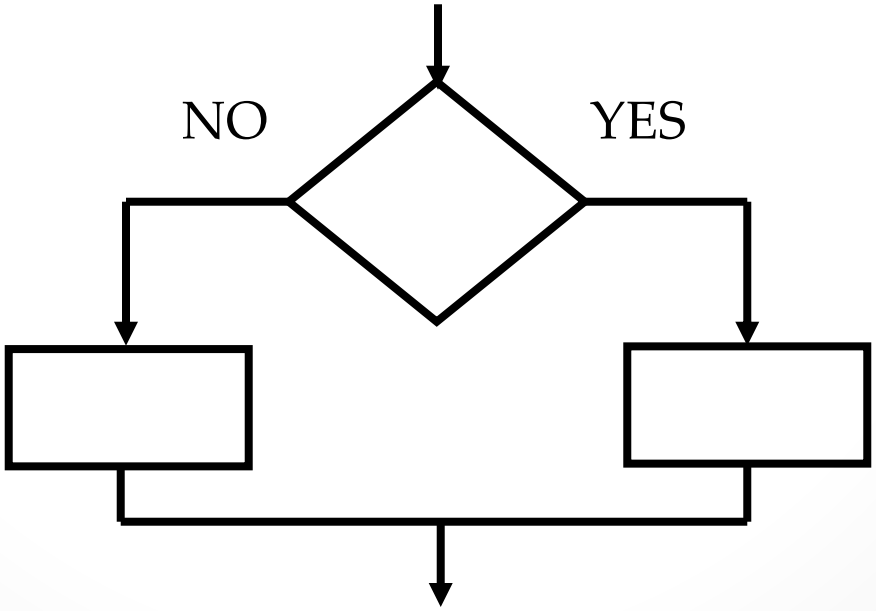
Decision Structure

- One of two possible actions is taken, depending on a condition.



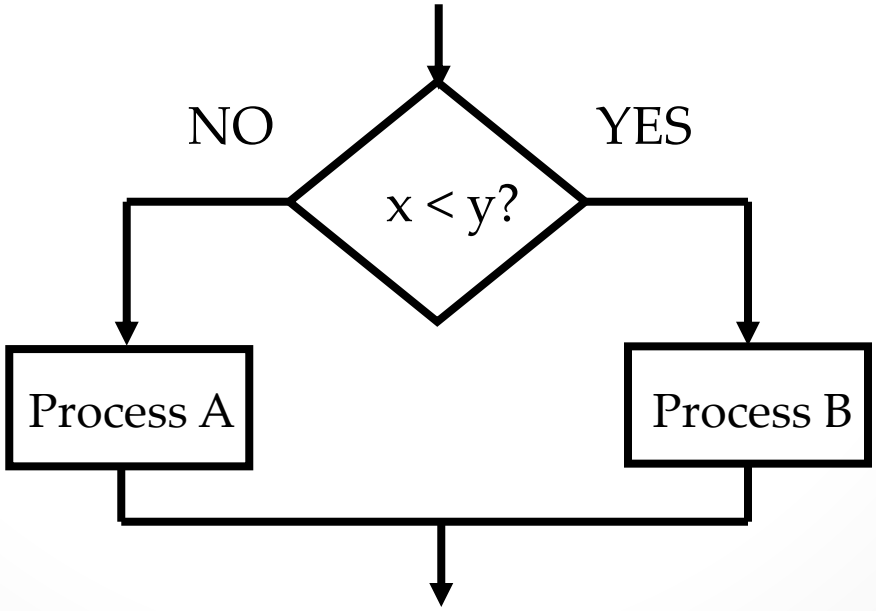
Decision Structure

- 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



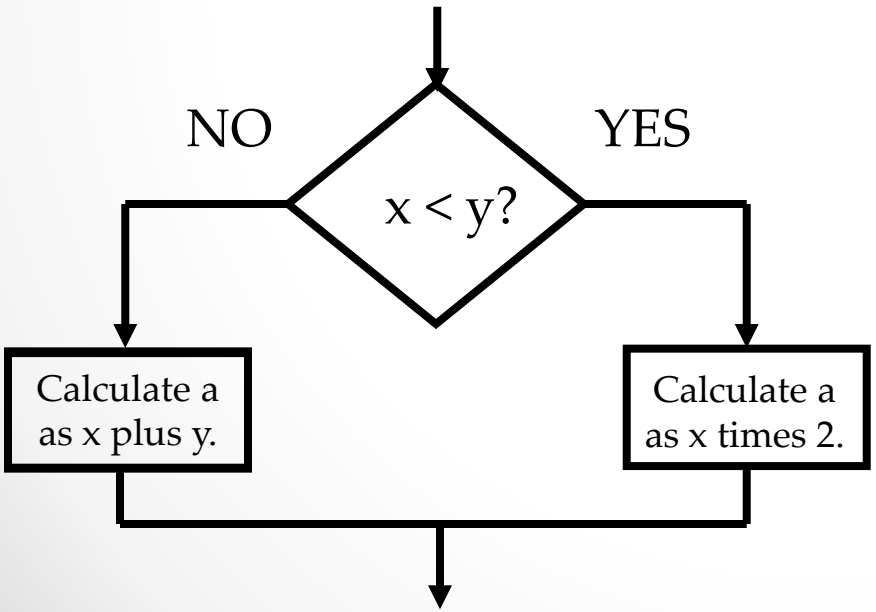
Decision Structure

- 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.



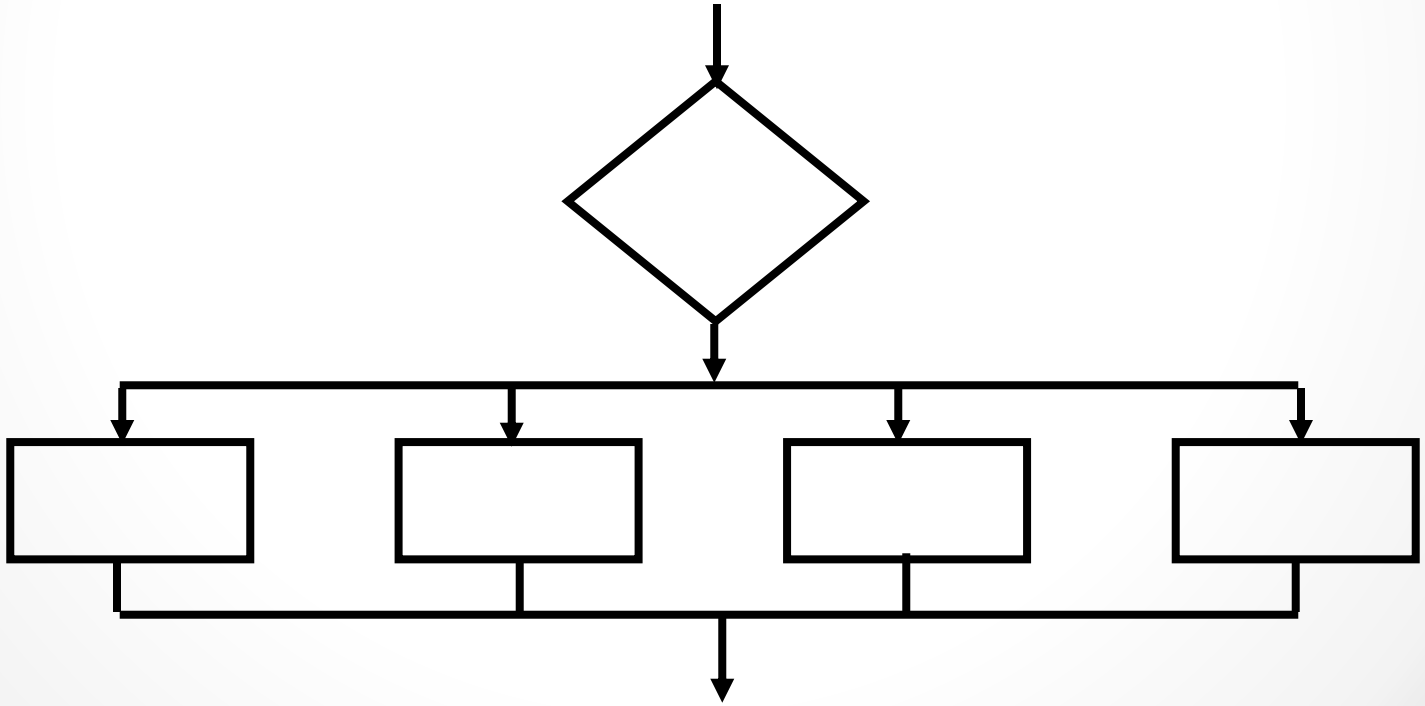
Decision Structure

- The flowchart segment below shows how a decision structure is expressed in C++ as an if/else statement.



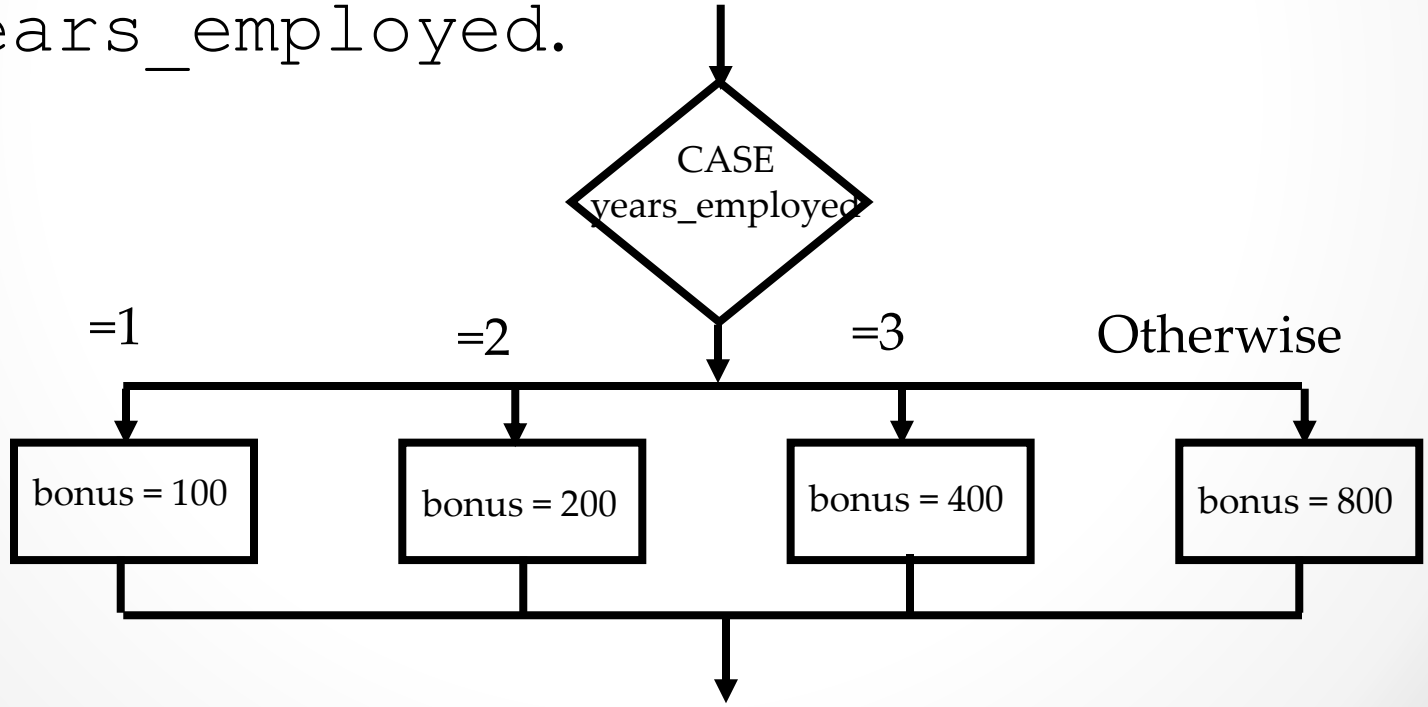
Case Structure

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

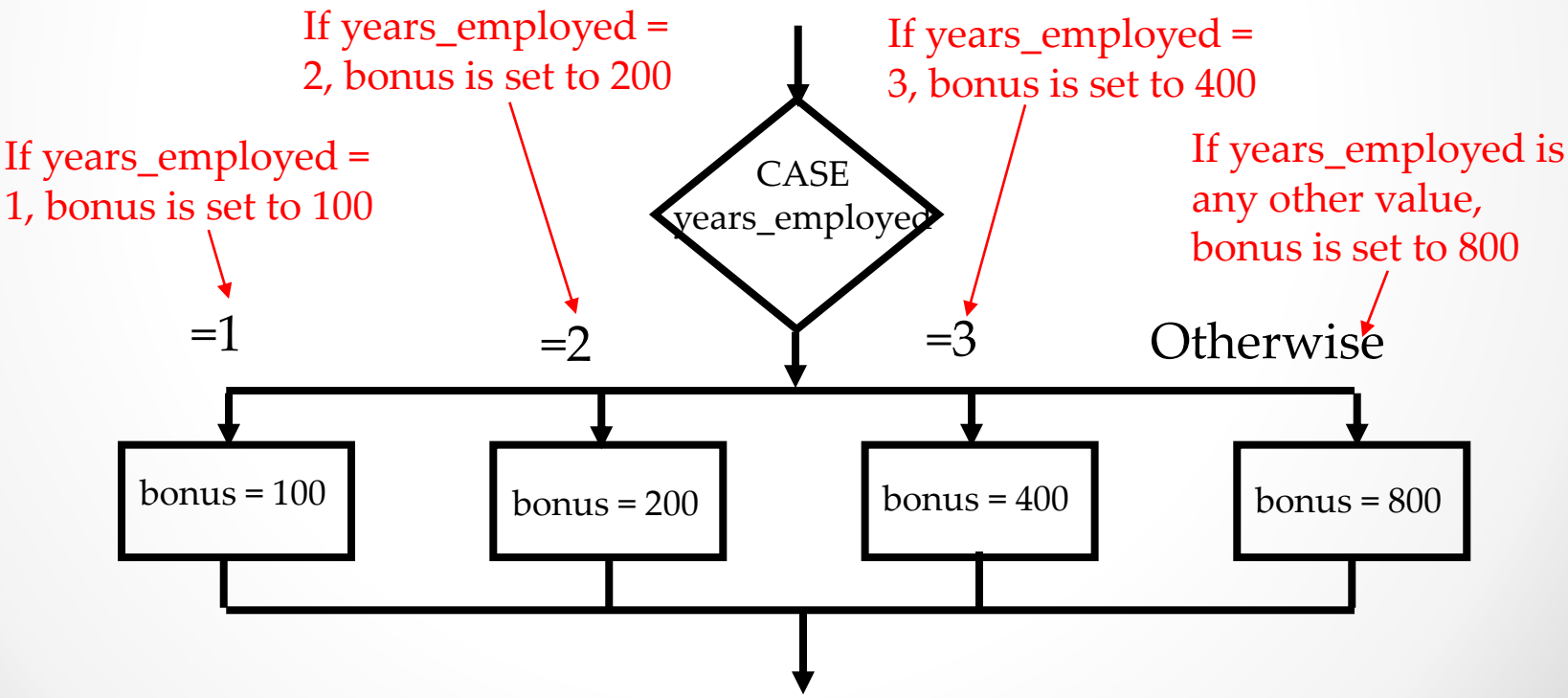


Case Structure

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

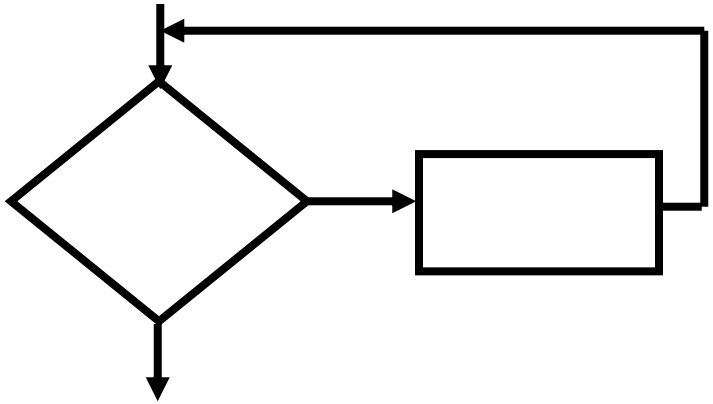


Case Structure



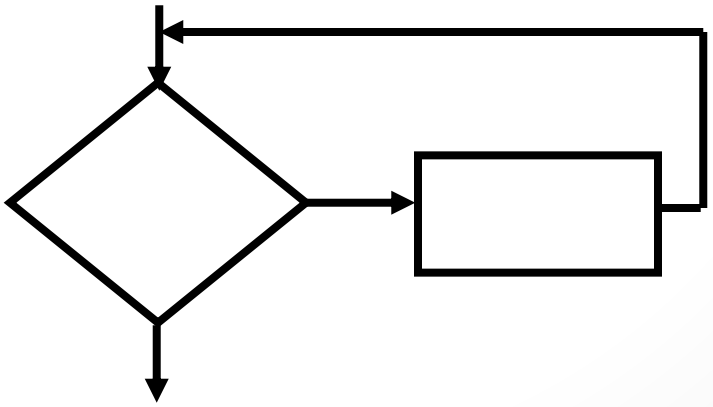
Repetition Structure

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



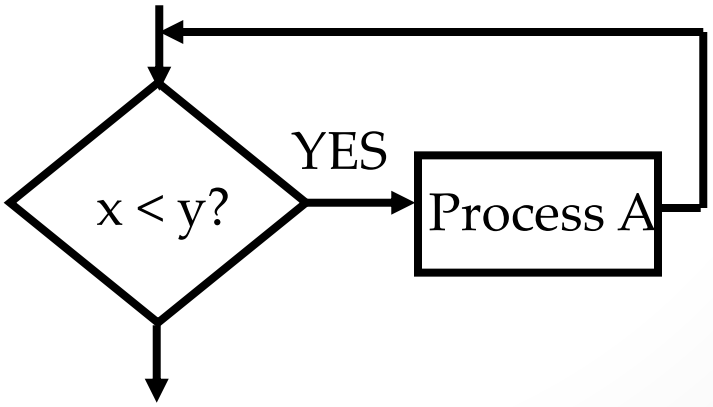
Repetition Structure

- 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.



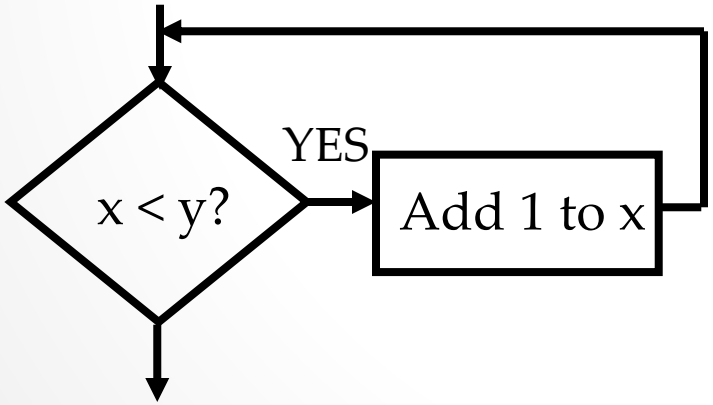
Repetition Structure

- 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.



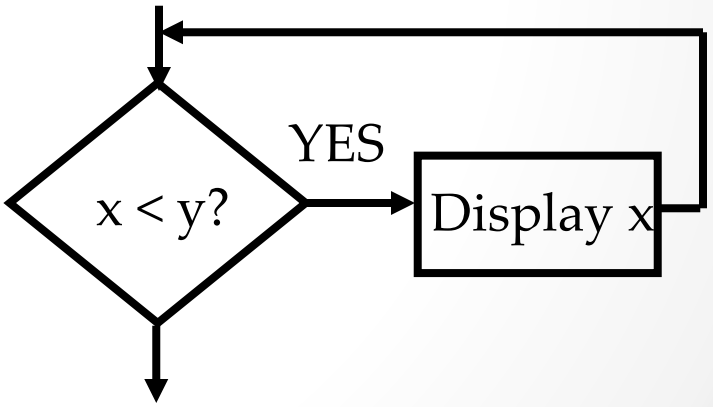
Repetition Structure

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



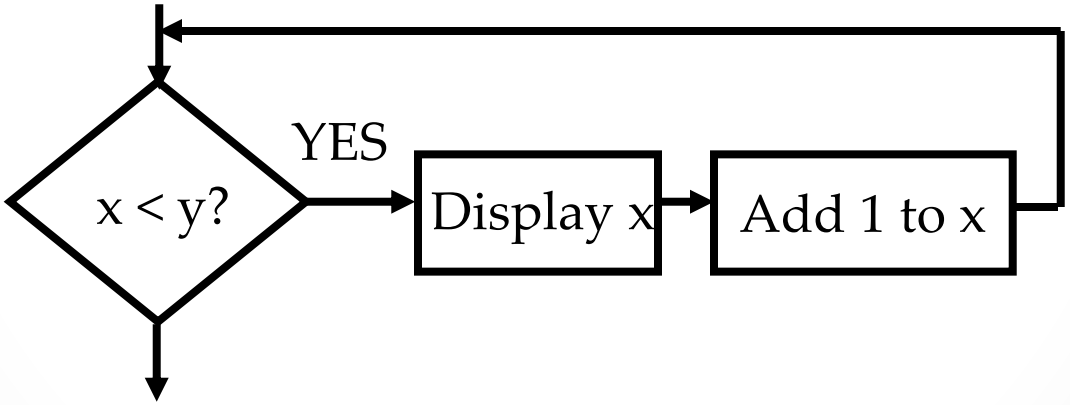
Controlling a Repetition Structure

- 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?



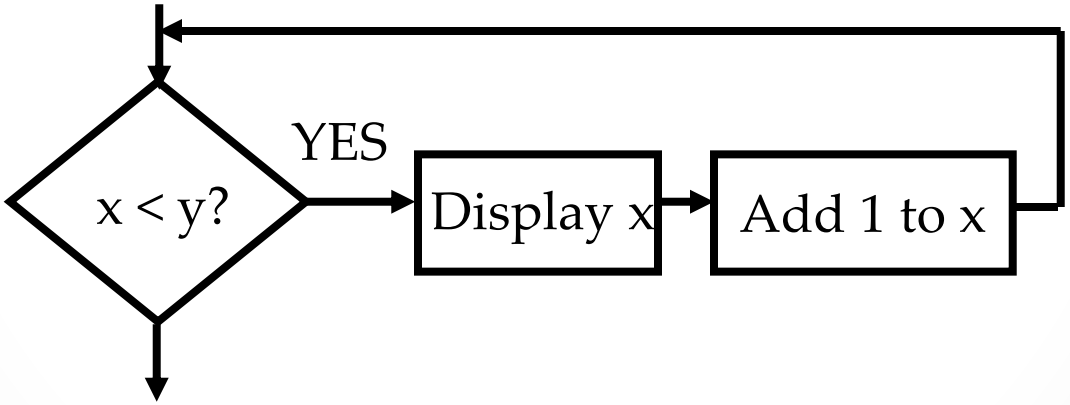
Controlling a Repetition Structure

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



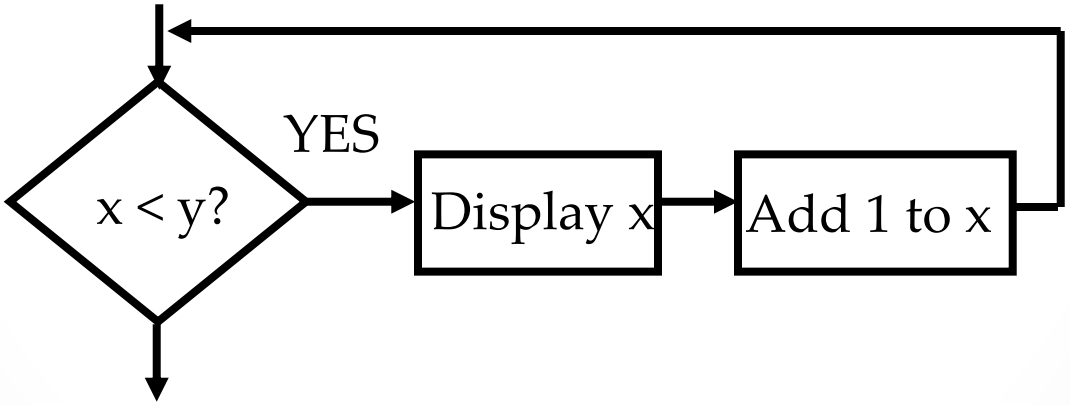
A Pre-Test Repetition Structure

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



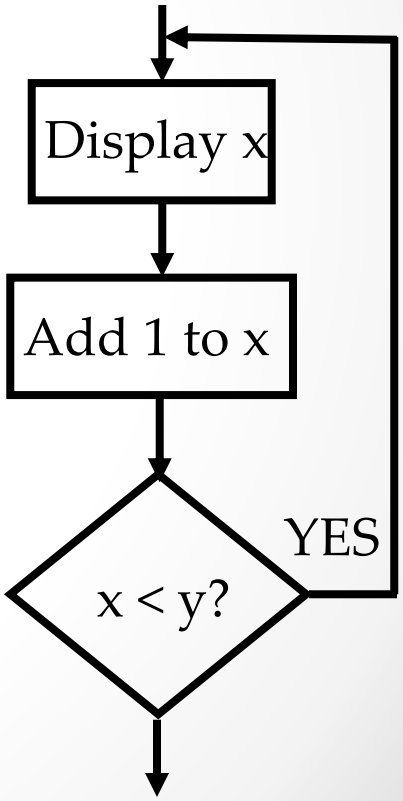
A Pre-Test Repetition Structure

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



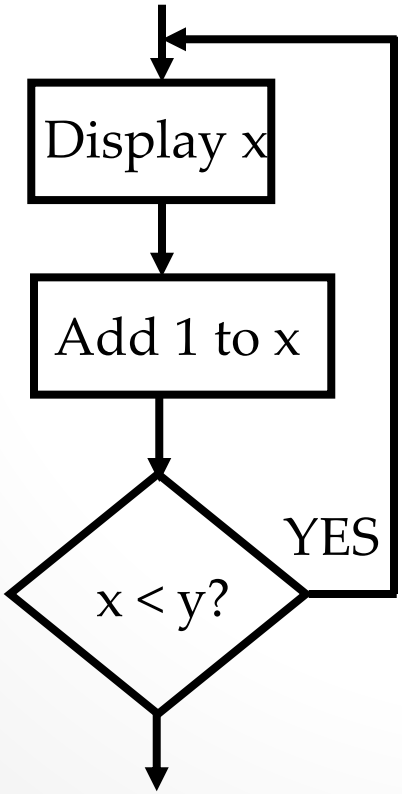
A Post-Test Repetition Structure

- 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.

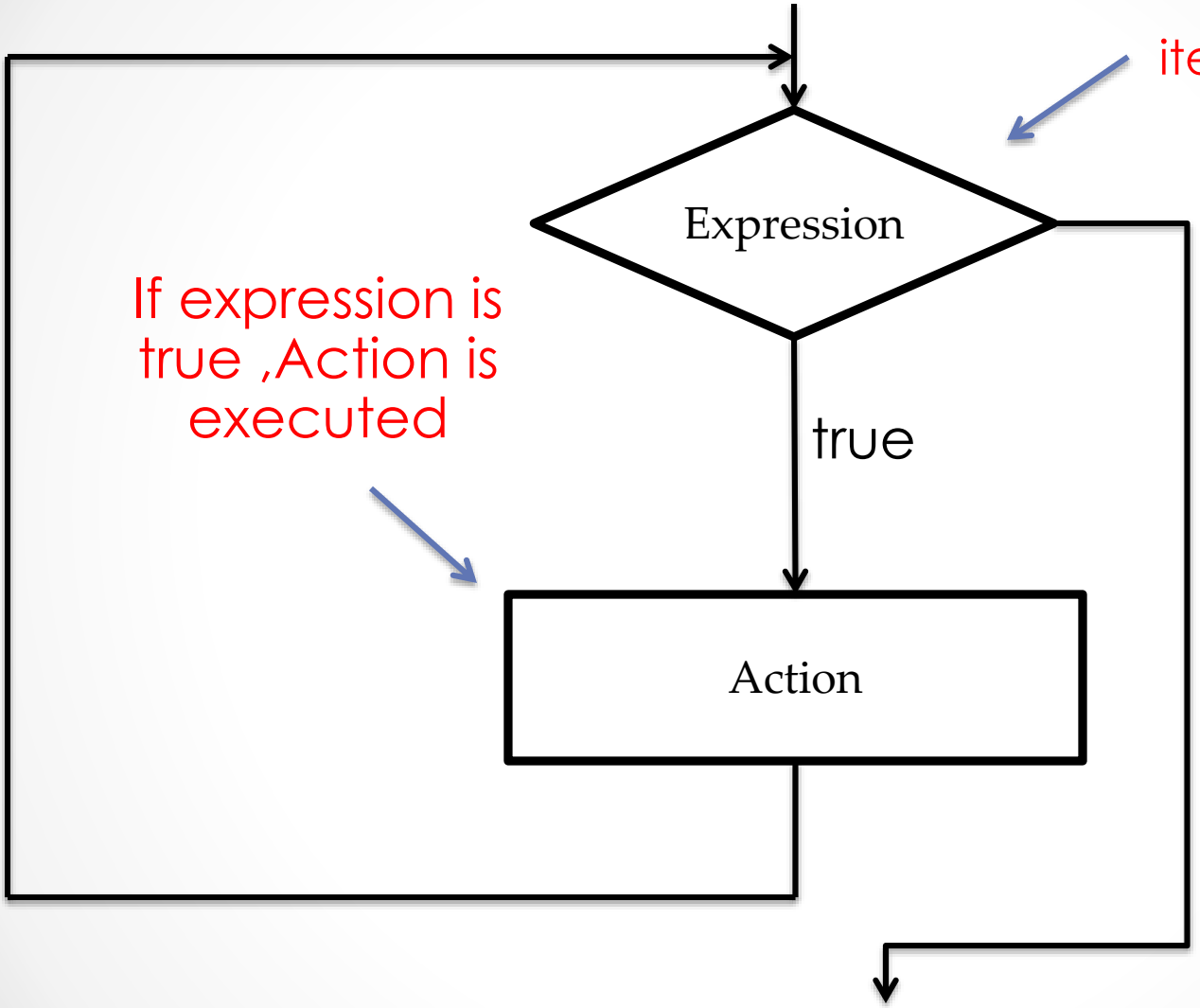


A Post-Test Repetition Structure

- The flowchart segment below shows a post-test repetition structure expressed in C++ as a do-while loop.



WHILE Loop



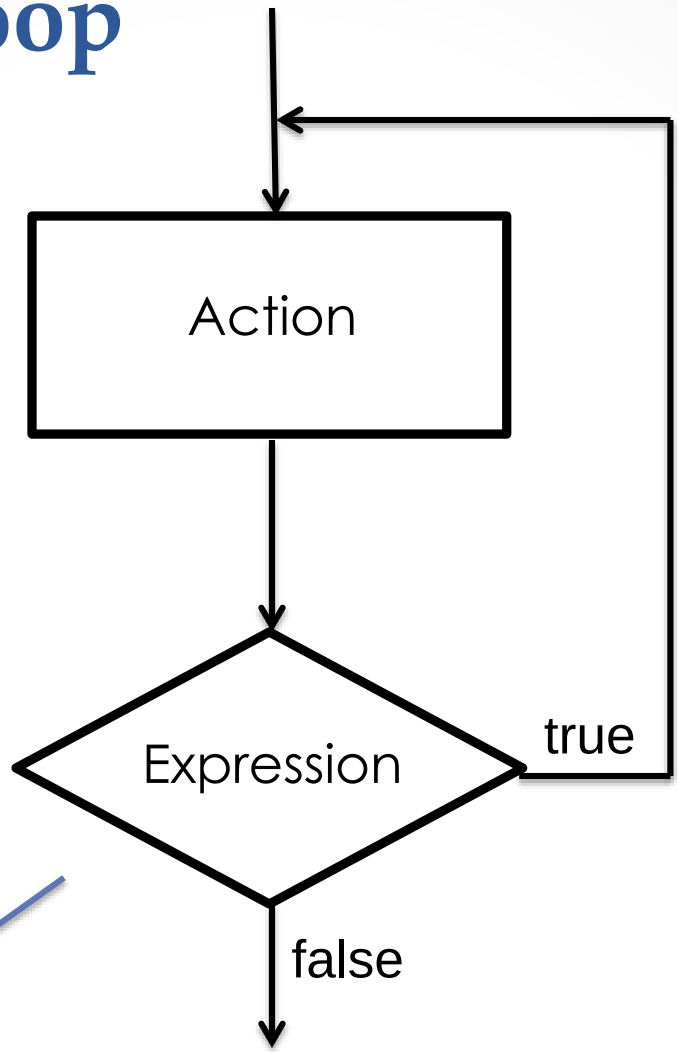
Expression is evaluated at the start of each iteration of the loop

If expression is true ,Action is executed

If expression is false, program execution continues with next statement outside the loop

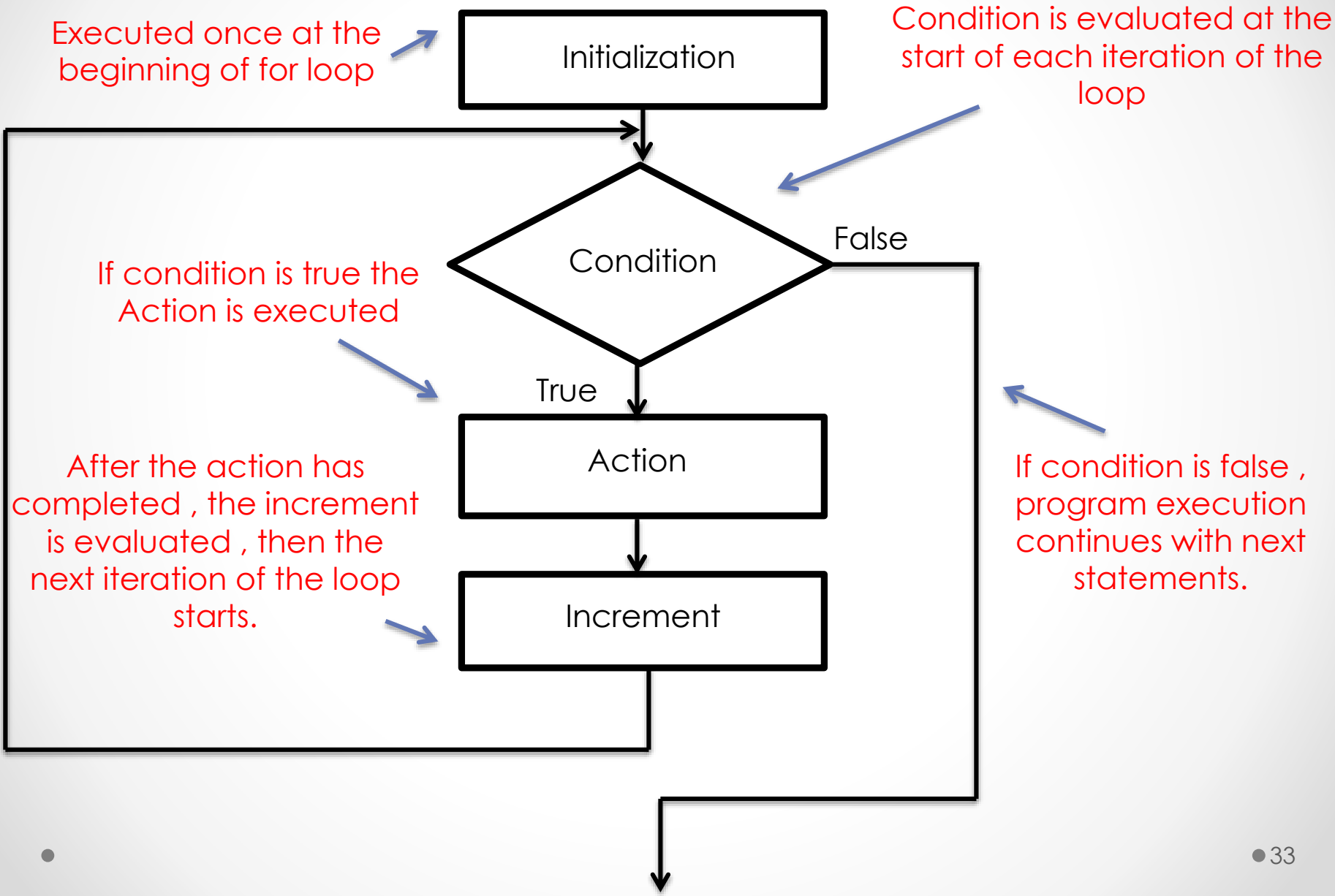
DO WHILE Loop

Execute *Action first*



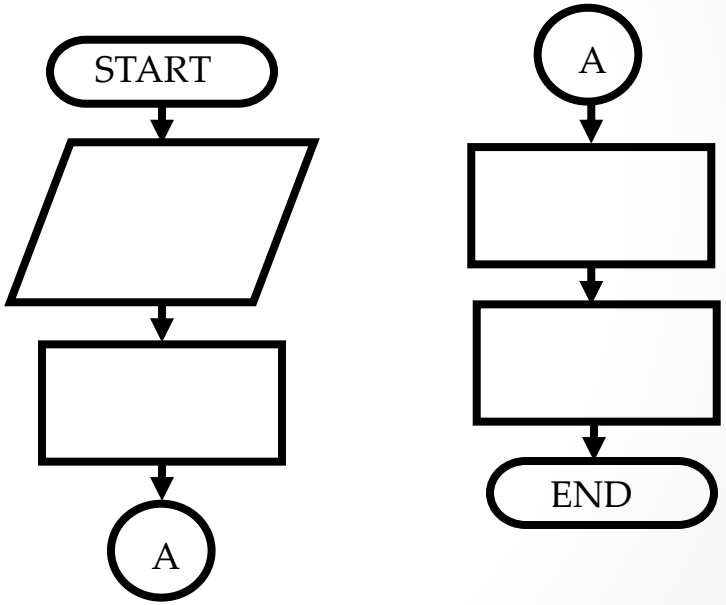
Expression is evaluated at the End , If *Expression* is true then execute *Action* again , Repeat this process until *Expression* evaluates to false

FOR Loop


Executed once at the beginning of for loop
Initialization
Condition is evaluated at the start of each iteration of the loop
Condition
False
If condition is true the Action is executed
True
Action
After the action has completed , the increment is evaluated , then the next iteration of the loop starts.
Increment
If condition is false , program execution continues with next statements.

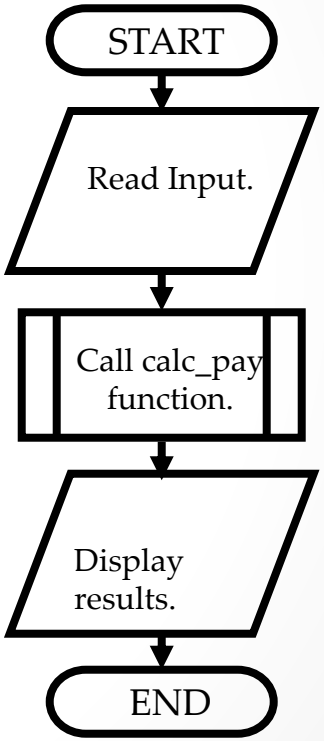
Connectors

- The "A" connector indicates that the second flowchart segment begins where the first segment ends.



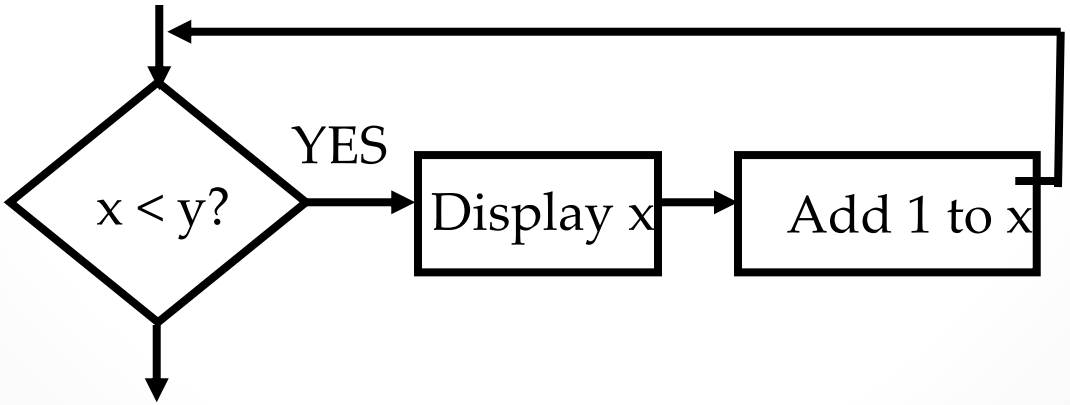
Subroutines (Modules)

- A program module (such as a function in C++) is represented by a special symbol.
- The position of the module symbol indicates the point the module is executed.
- A separate flowchart can be constructed for the module.



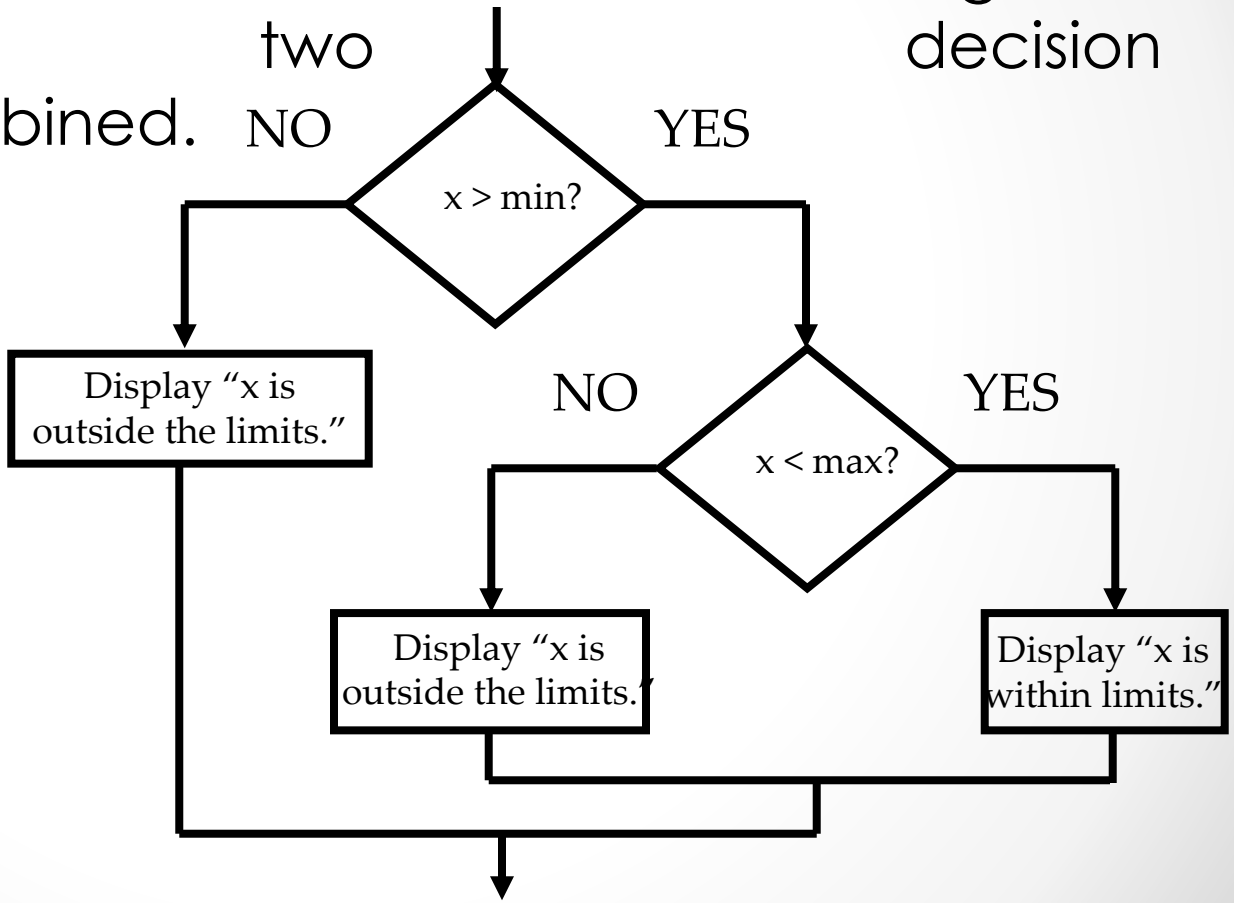
Combining Structures

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



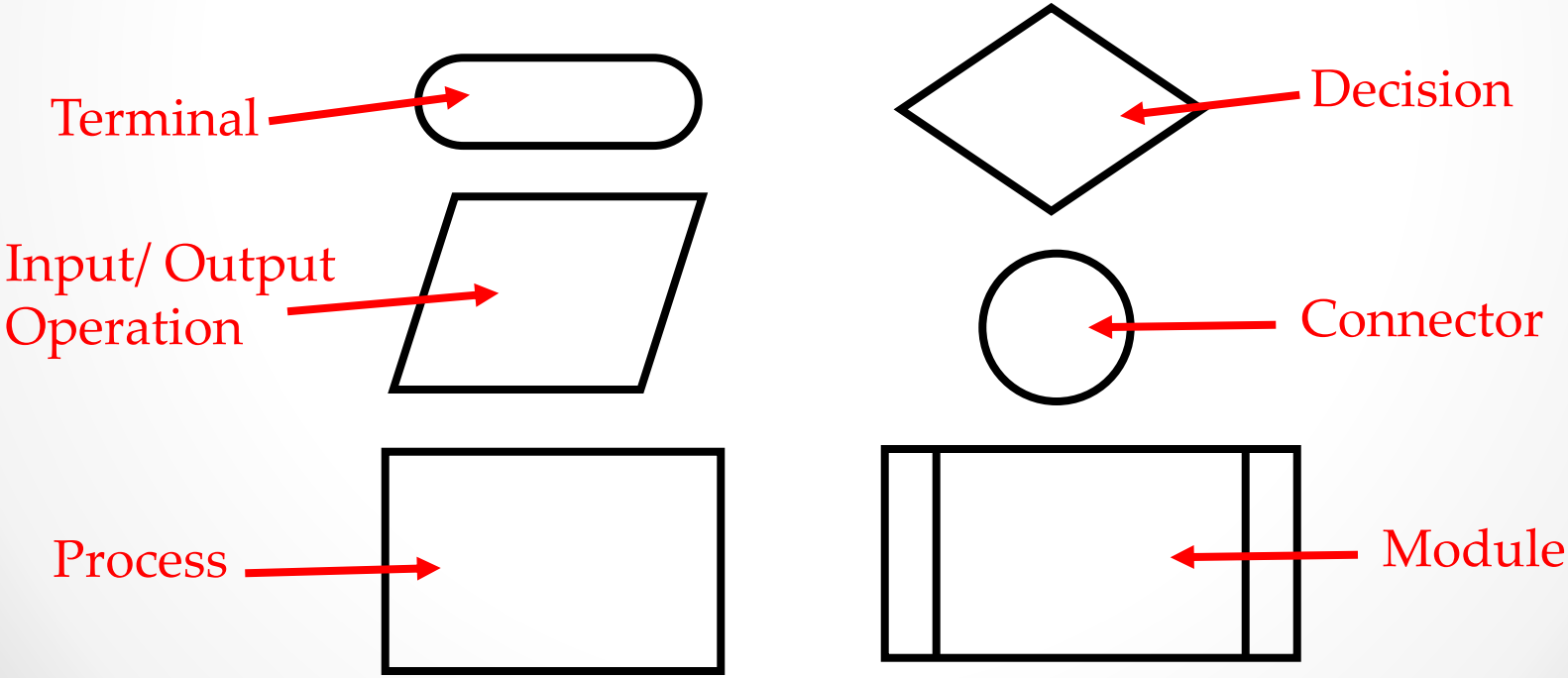
Combining Structures

- This flowchart shows two decision structures combined.



Review

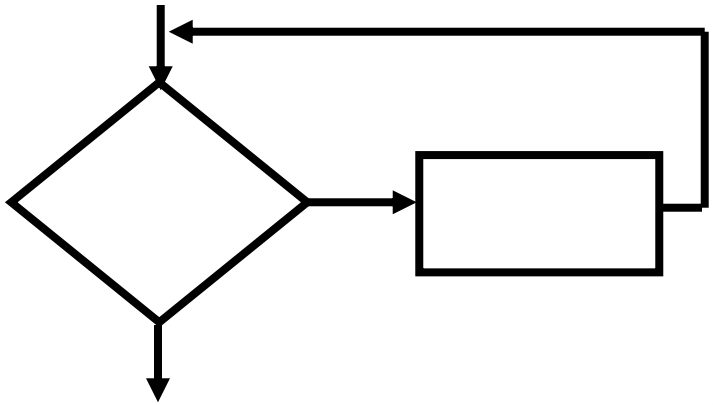
- What do each of the following symbols represent?



Review

- What type of structure is this?

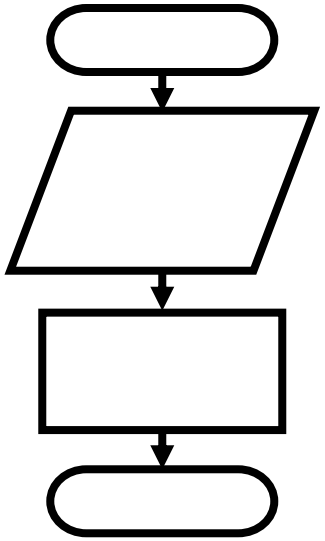
Repetition



Review

- What type of structure is this?

Sequence

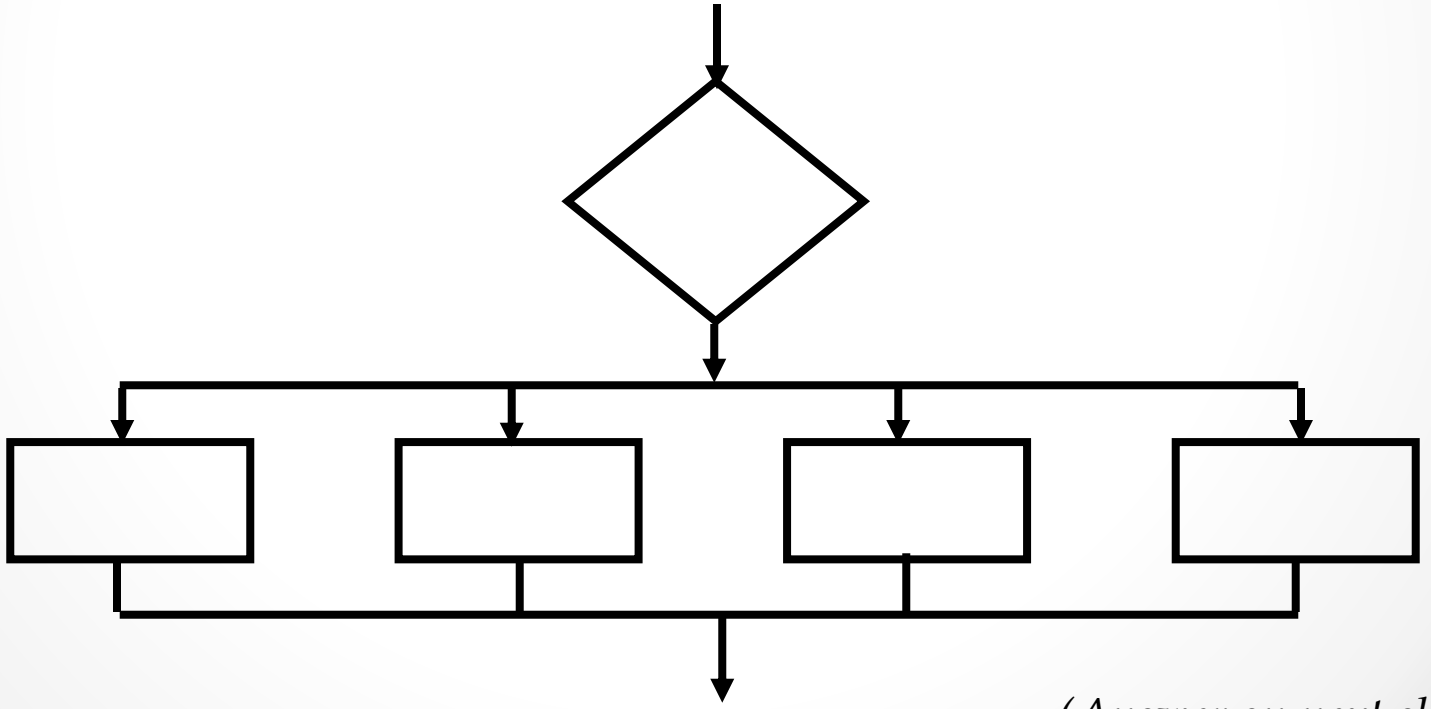


(Answer on next slide)

Review

- What type of structure is this?

Case

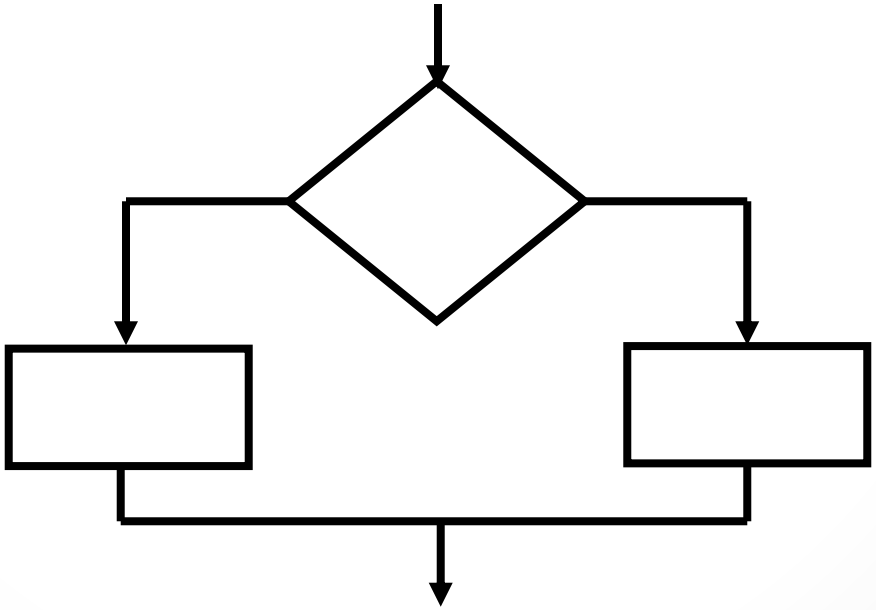


(Answer on next slide)

Review

- What type of structure is this?

Decision



(Answer on next slide)

Working with Fields

Calculations

+	add
-	subtract
*	multiply
/	divide
** or ^	exponentiation
( )	grouping
%	modulo

Selection

>	greater than
<	less than
=	equal to
>=	greater than or equal to
<=	less than or equal to
< >	not equal to

Logical Operators

NOT (!)
AND (&&)
OR (||)

- Logical operators are used to **combine several conditions** into one compound condition
- The outcome of condition is True or False

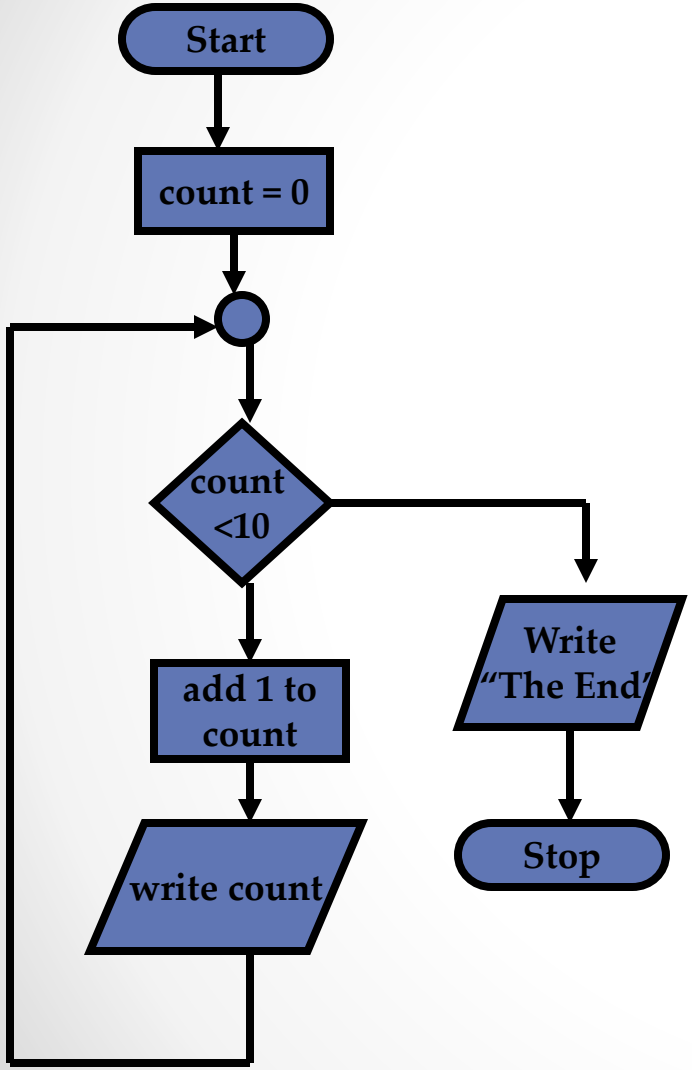
- Operators above are listed in order of precedence
  Not is evaluated first, then AND, then OR
- The relational operators have higher precedence that the logical operators
- It is usually best to include the extra parentheses

The Looping Structure

In flowcharting, one of the more confusing things is to separate selection from looping. This is because each structure use the diamond as their control symbol.

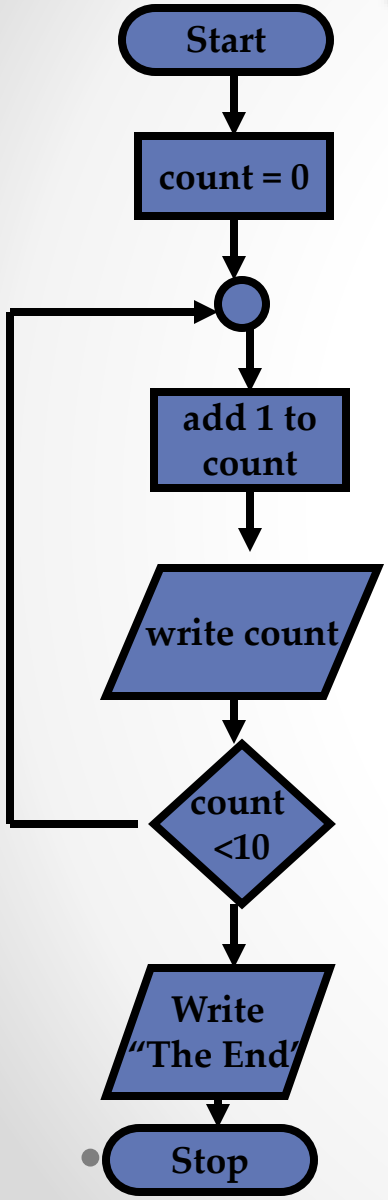
Practice makes perfect ☺

WHILE Loop



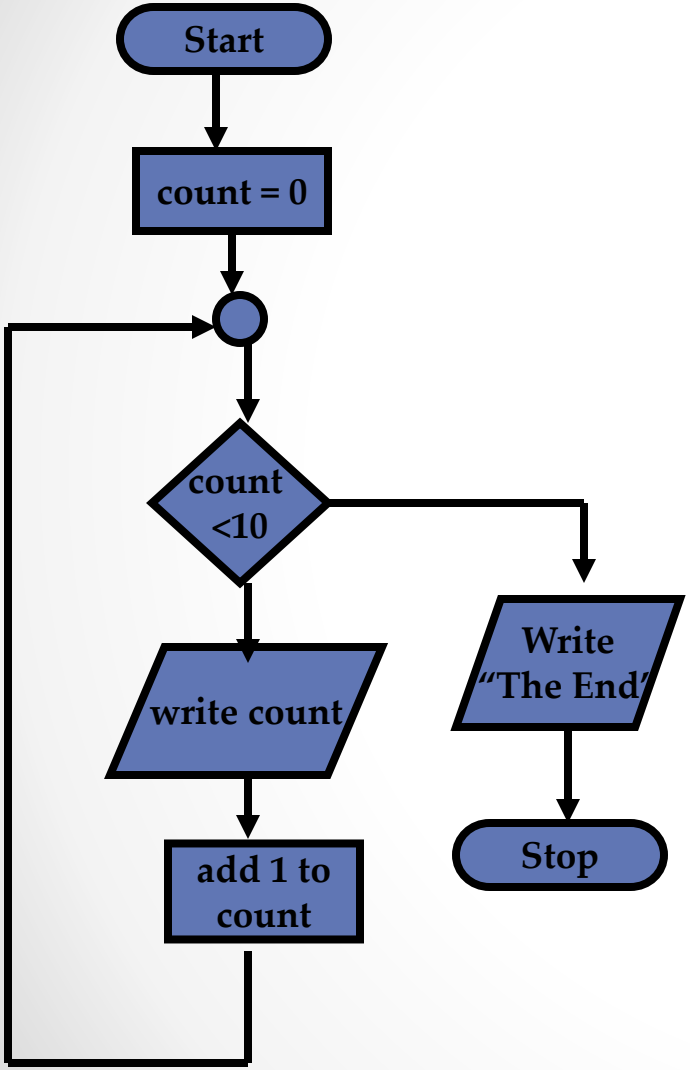
Program calculates and displays the sum of integers as long as it is less than 10.

DO WHILE Loop



Program calculates and displays the sum of integers as long as it is less than 10.

FOR Loop

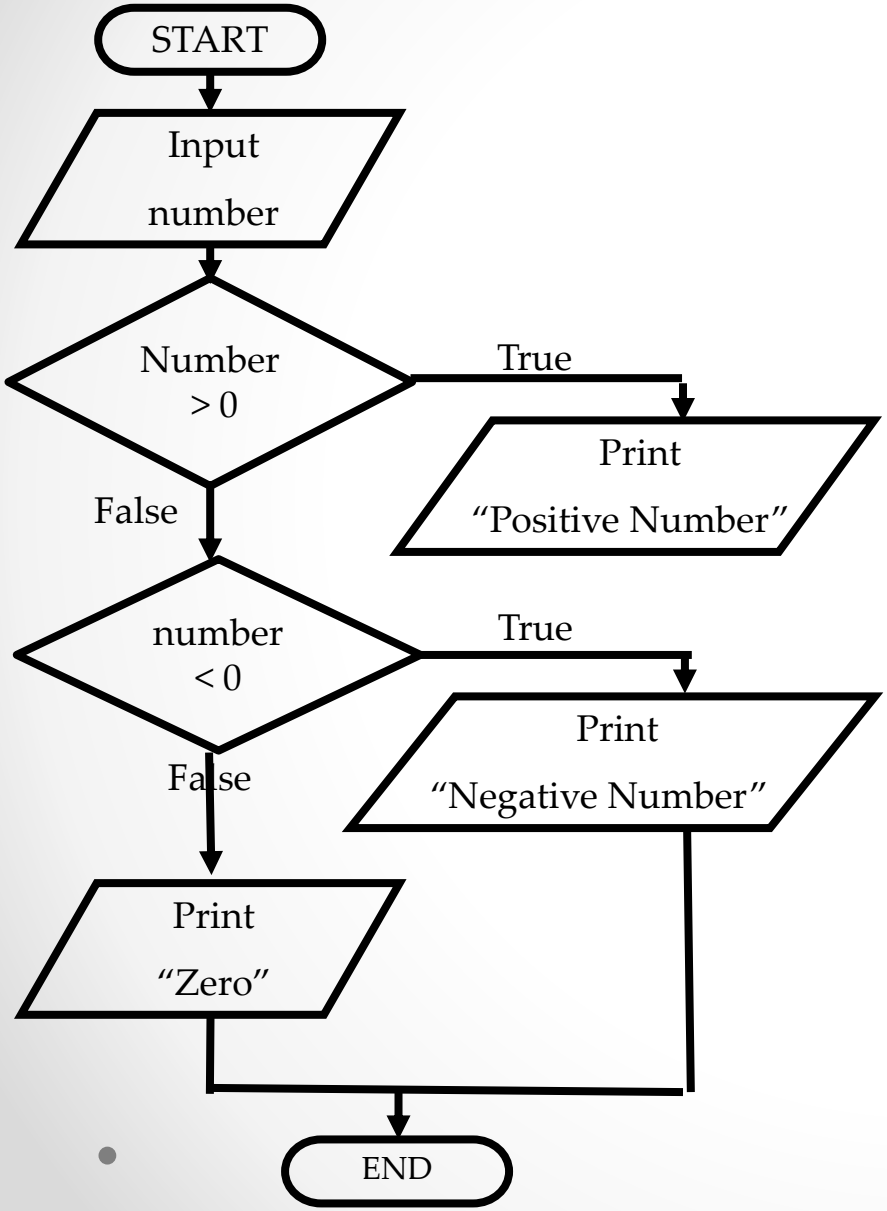


Program calculates and displays the sum of integers as long as it is less than 10.

Exercise 1

- Algorithm for program that reads a number and determines whether it is positive, negative or zero.

Exercise 1: Solution



Exercise 2

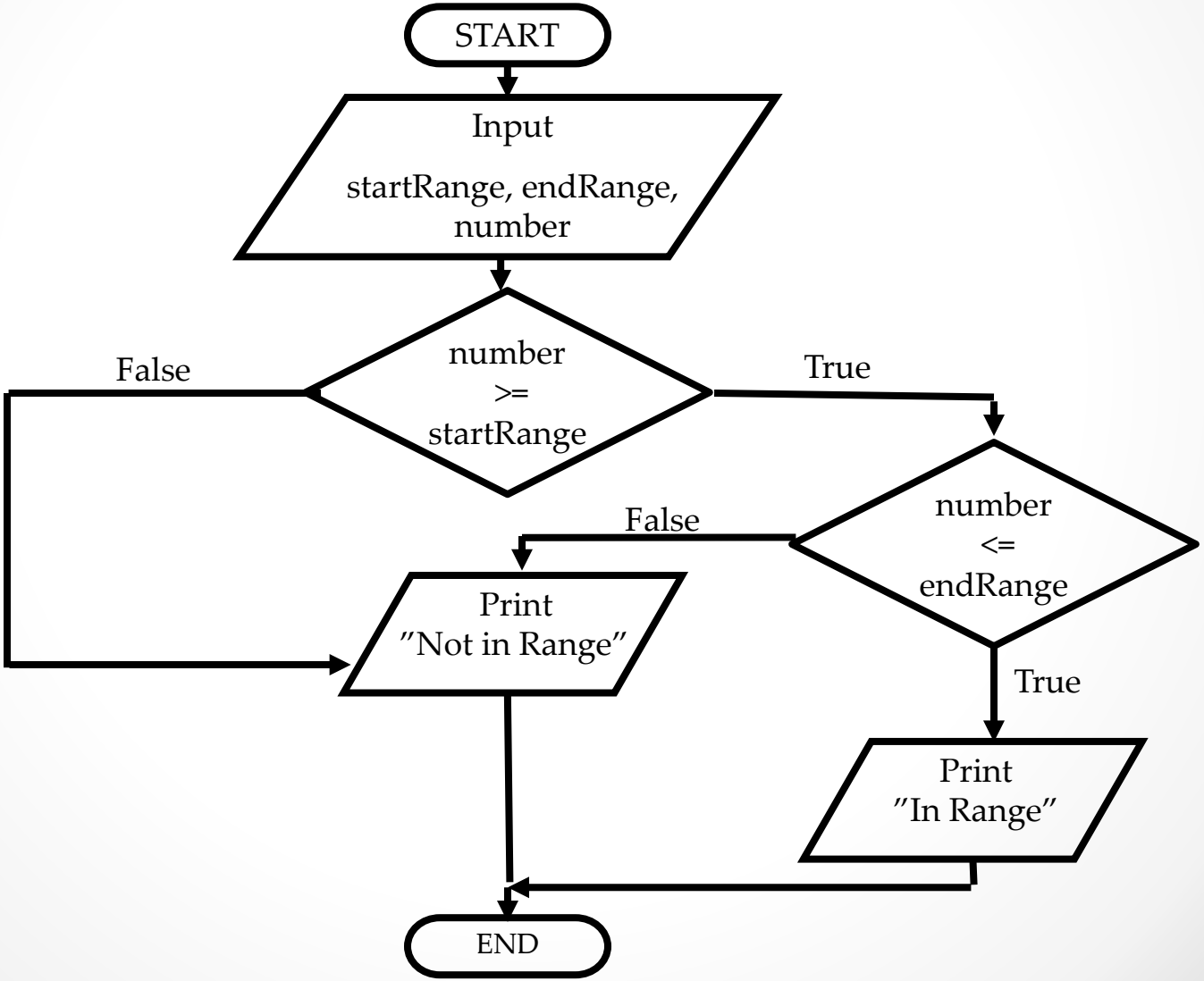
- Algorithm for program that checks whether an input number lies within a specified range.

  For example: If user enters range (20, 50) and queried number 34, the program should display “In range”.

  On the other hand, if he enters 76, the program should display “Not in Range”.

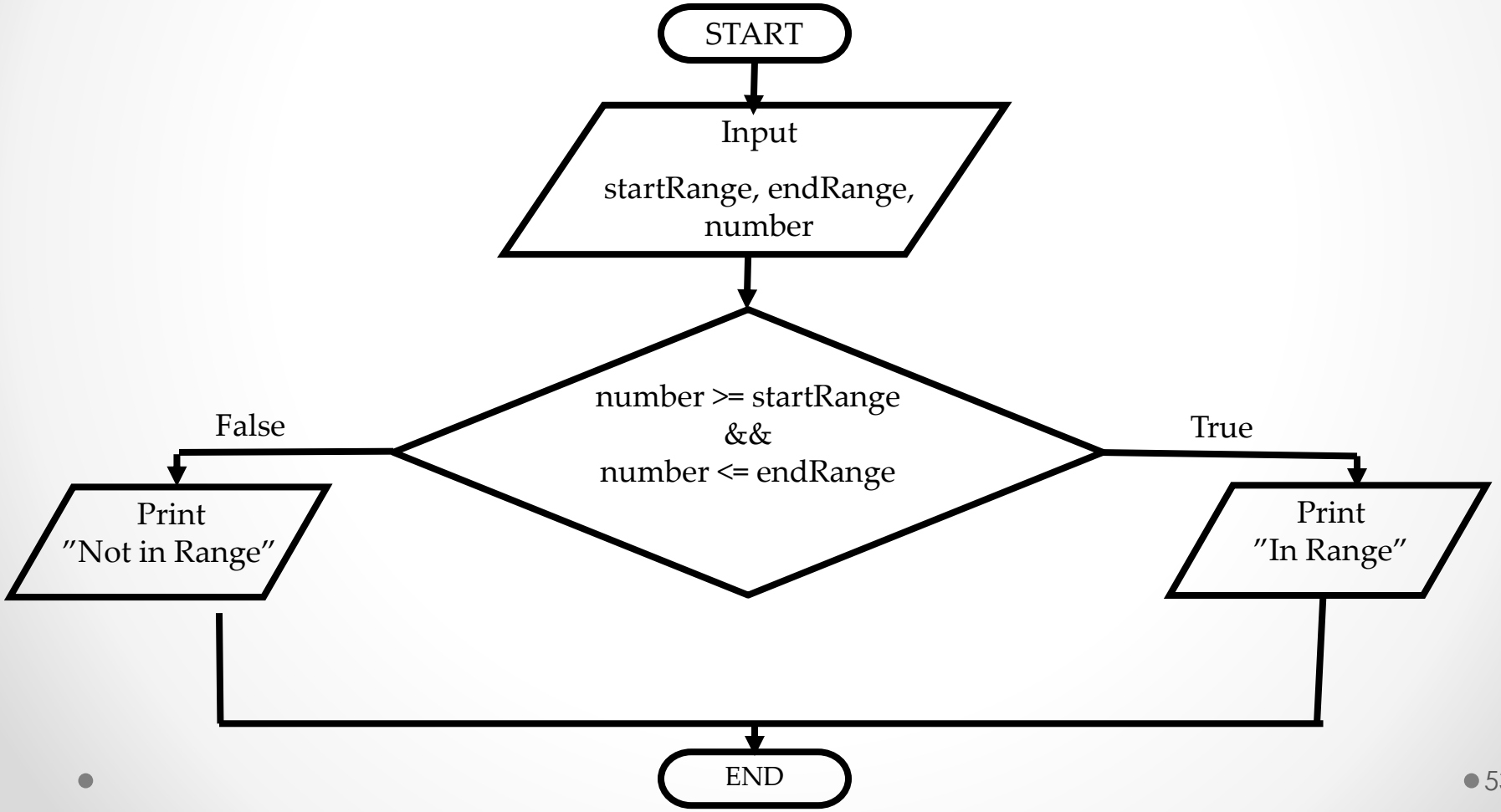
Exercise 2: Solution

- Using nested IF structure



Exercise 2: Another Solution

- Using logical operators in IF condition



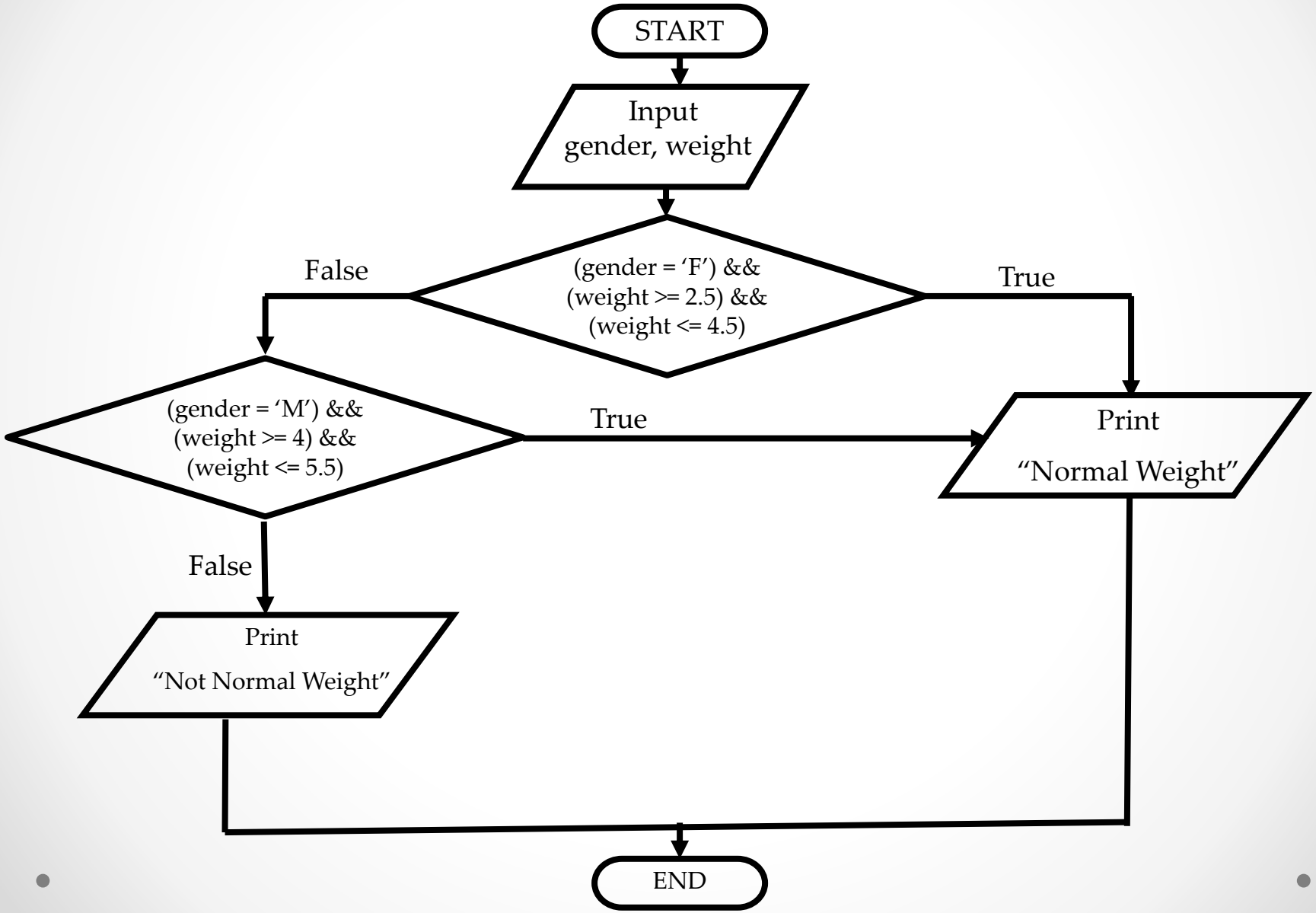
Exercise 3

- Algorithm for program that determines whether a baby's weight is normal or not.

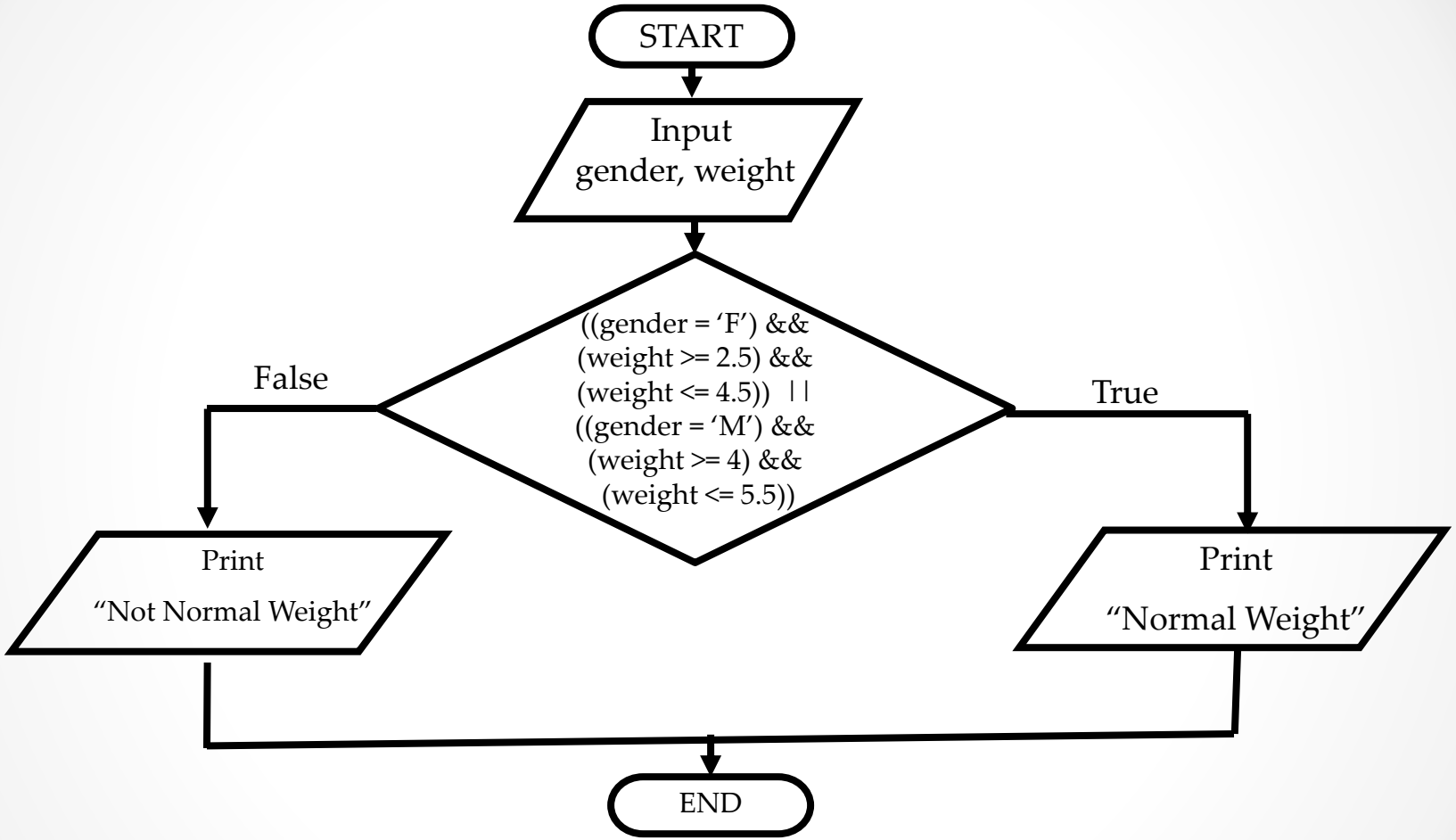
  For girls, normal babies weight are 2.5 to 4.5 KG.

  On the other hand, for boys the normal weights are 4 to 5.5 KG.

Exercise 3: Solution



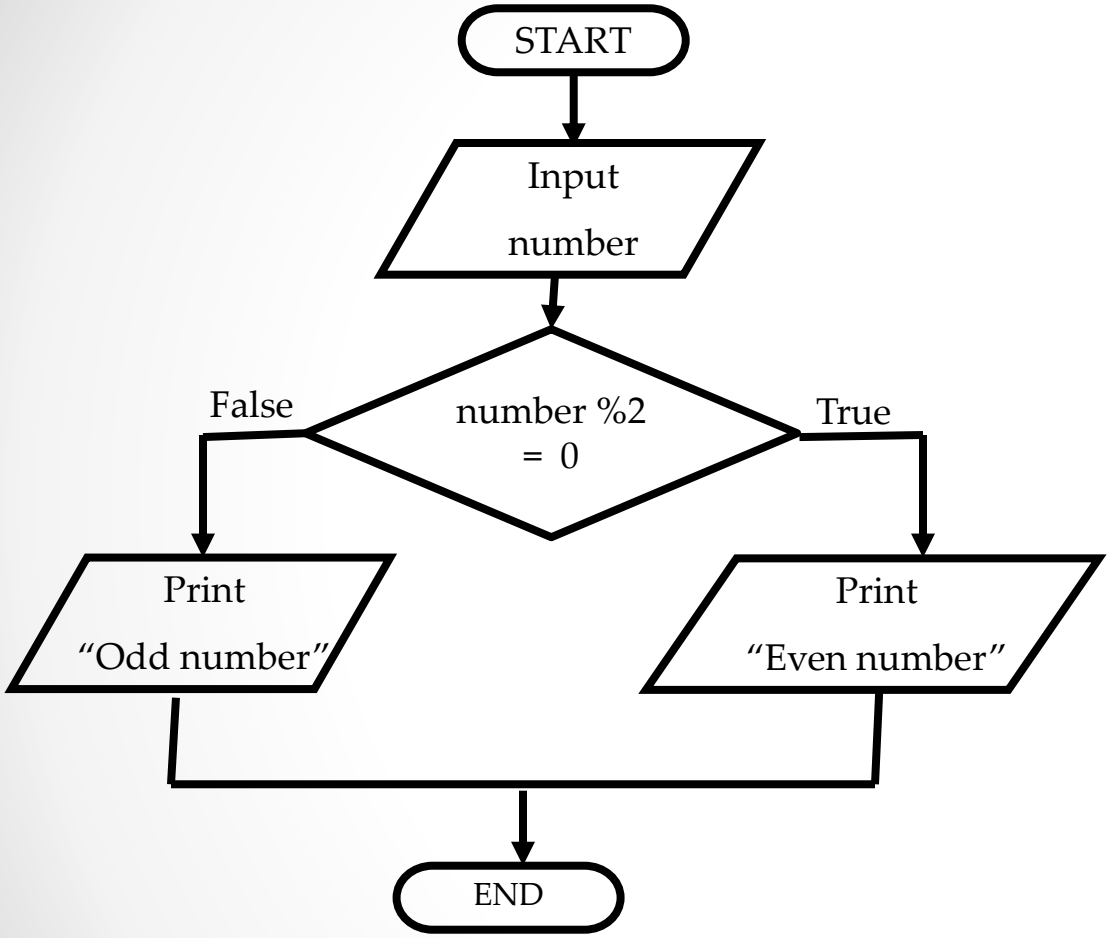
Exercise 3: Another Solution



Exercise 4

- Algorithm for program that reads a number and determines whether it is even or odd.

Exercise 4: Solution



Exercise 5

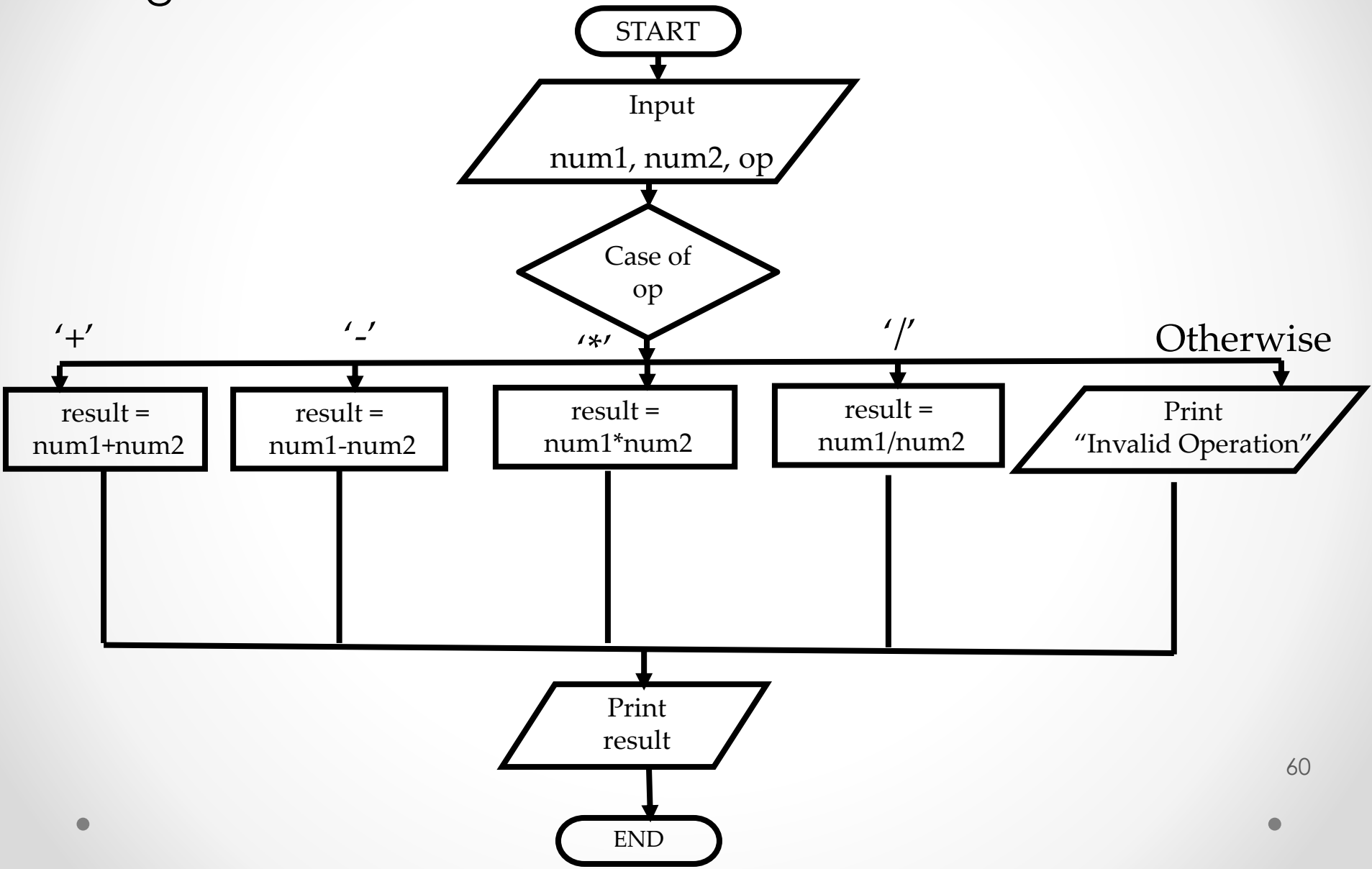
- Algorithm for a calculator that works on integer numbers. The user enters two numbers to perform only one of four basic arithmetic operations (+, -, * and /).
- The interaction with the user might look like this: Enter your expression:

  1 2 +

  The result is: 3
- Hint: The user is allowed to do only one operation at a time.

Exercise 5: Solution

- Using case structure



Exercise 6

- Algorithm for program that converts seconds to equivalent hours, minutes and seconds.

Exercise 6: How to Solve?

totalSeconds = 4000

How to convert to hours, minutes and seconds?

- First, to calculate the number of **seconds**, get the remainder of dividing *totalSeconds* by 60 and print it.

  seconds = 4000 % 60 = 40 seconds.

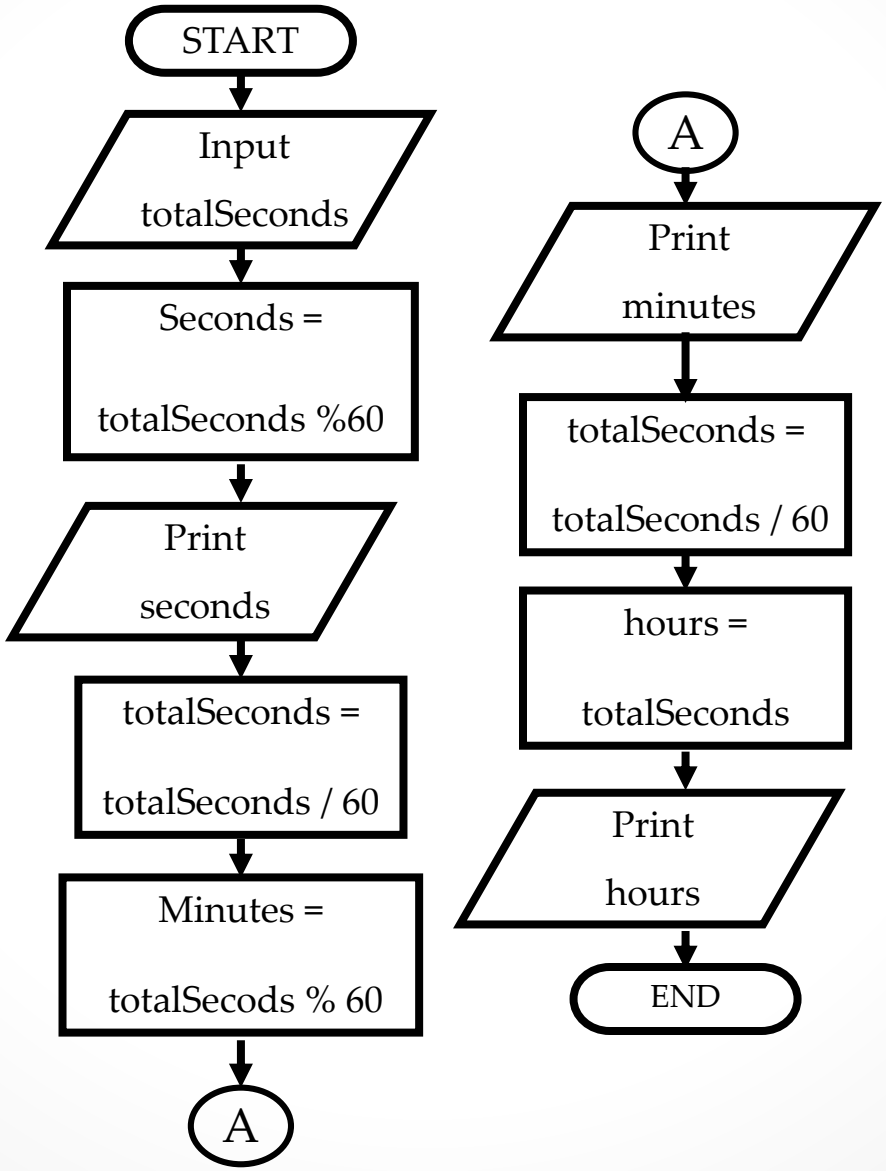
- Second, to calculate the number of **minutes**, divide *totalSeconds* by 60, then get the remainder of dividing it by 60 and print it.

  totalSeconds= 4000/60 = 66.666 then 66.666 % 60=6 minutes

- Third, to calculate the number of **hours**, divide *totalSeconds* by 60 and print it.

  hours = 66.666 /60= 1hour

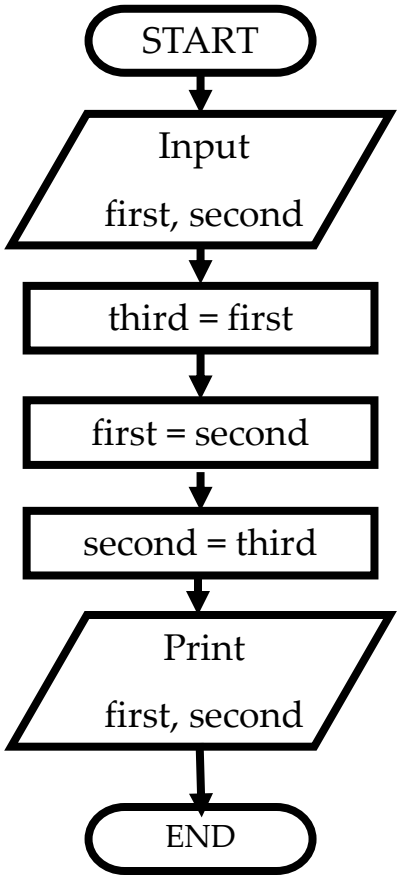
Exercise 6: Solution



Exercise 7

- Algorithm for a program to swap the values of two integers using third variable.

Exercise 7: Solution



Break 10 minutes

…

Programming Using C++

…

C++ Program

```
#include <iostream>
using namespace std;
int main()
{
      // This is my first C++ program
      cout << "Hello World" << endl;
      return 0;
}
```

What does the code mean?

N.B.: C++ is case sensitive.

- **//Hello World Program.**
  - Comments are parts of the source code disregarded by the compiler.
  - It can be a description for the program or a given snippet of code - internal documentation for the code.
  - For single line comment → //.....
  - For multiple line comments → /* .....
    ..... */
- **#include <iostream>**
  - Lines beginning with a hash sign (#) are called **pre-processor directives** .
  - They are not regular code lines with expressions but indications for the compiler's preprocessor.
  - The directive **#include <iostream>** tells the preprocessor to include the *iostream* standard file.
  - This specific file (iostream) includes the *declarations* of the basic standard input-output library in C++, and it is included because its functionality is going to be used later in the program.

What does the code mean?

- **using namespace std;**
  - All the elements of the standard C++ library are *declared* within what is called a namespace.
  - Namespaces allows us *to group a set of* global classes, objects and/or functions under a name.
  - Namespace ***std*** contains all the classes, objects and functions of the standard C++ library.

Without namespace

Code:

```
#include <iostream>

int main () {
  std::cout << "Hello world!\n";
  return 0;
}
```

If you specify **using namespace std** then you don't have to put **std::** throughout your code.

With namespace

Code:

```
#include <iostream>
using namespace std;

int main () {
  cout << "Hello world!\n";
  return 0;
}
```

What does the code mean?

- int **main()**
  - Remember, every C++ program is made of one or more **functions.**
    - A *function* is a piece of code that accomplishes one specific task.
  - Every executable C++ program has **one** function called main()
    - This is where execution (the actual running) of the program starts.
  - main( ) function is the heart of the program and *also called program building block*.
  - When you run the program you are essentially telling the computer to ***call*** this main() function
    - ***Calling*** a function means to execute the code in the function.

- int **main()**
  - The int is the return type of the function.
    - In other words when it finishes what kind of result does it give back. A whole number? A character?
  - int means a whole number.
  - We want to give this number back to tell us if the program was successfully completed
- int **main()**
  - The brackets is the **parameter list** for this function.
  - The **parameter list** indicates what this function needs to be given from the outset for it to work.
  - we give nothing to this function, we can also write:

    int main (void)

What does the code mean?

- **`cout << "Hello World" << endl;`**
  - This line tells the computer to print something to the screen.
  - If you want to *print* something *to the screen,* use the **cout** function.
  - The **endl** adds an "end line" and flushes the buffer.
  - **cout** is the name of the standard output stream in C++, and the meaning of the entire statement is to insert a sequence of characters (in this case the Hello World sequence of characters) into the standard output stream (***cout***, which usually corresponds to the screen).
  - ***cout*** is declared in the ***iostream*** standard file within the *std* namespace, so that's why we needed to include that specific file and to declare that we were going to use this specific namespace earlier in our code.
  - **cout** is used in conjunction with the ***insertion operator***, which is written as **<<**
  - The **<<** operator inserts the data that follows it into the stream.

What does the code mean?

- **return 0;**
  - This is the last line in our program.
  - It basically means that the program is now done.
  - return means to go back to where the program was called from.
    - 0 is used to indicate that it was successfully completed (a general convention)

Another method:

- write: void main( ) → in this case, the function will return void, means it returns nothing and so, we do not need **return** statement.

What does the code mean?

- Note also that the code of `main()` is contained within the `{ }`
- These `{}` group code together.
- In this case it tells us that all the code between them are part of the `main` function:

```
#include <iostream>
 using namespace std;
 int main(){
   cout << "Hello World" << endl;
   return 0;
 }
```

- Also note that every command is terminated by a semi colon ;

Thank You