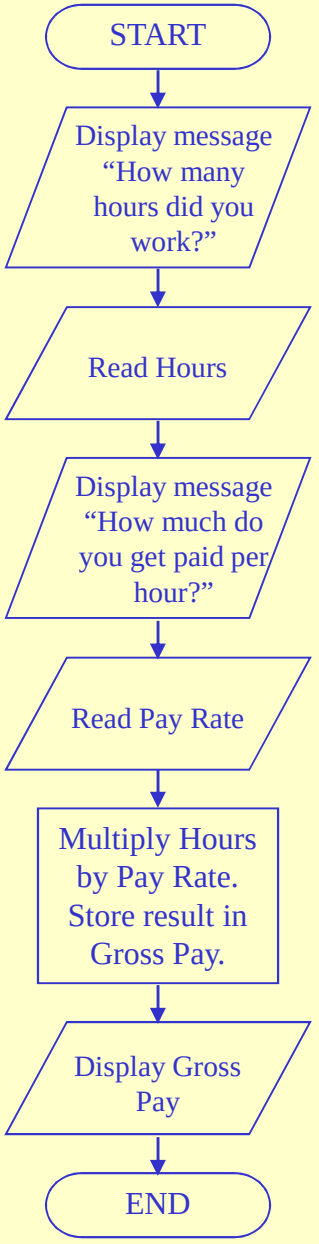
Introduction to Flowcharting

Computer Fundamentals
CSE1013

Md. Hasan Tareque [HT]
Em@il: hasantareque.cse.seu@gmail.com
Lecturer, Department of CSE
Southeast University

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 program in Chapter 1.

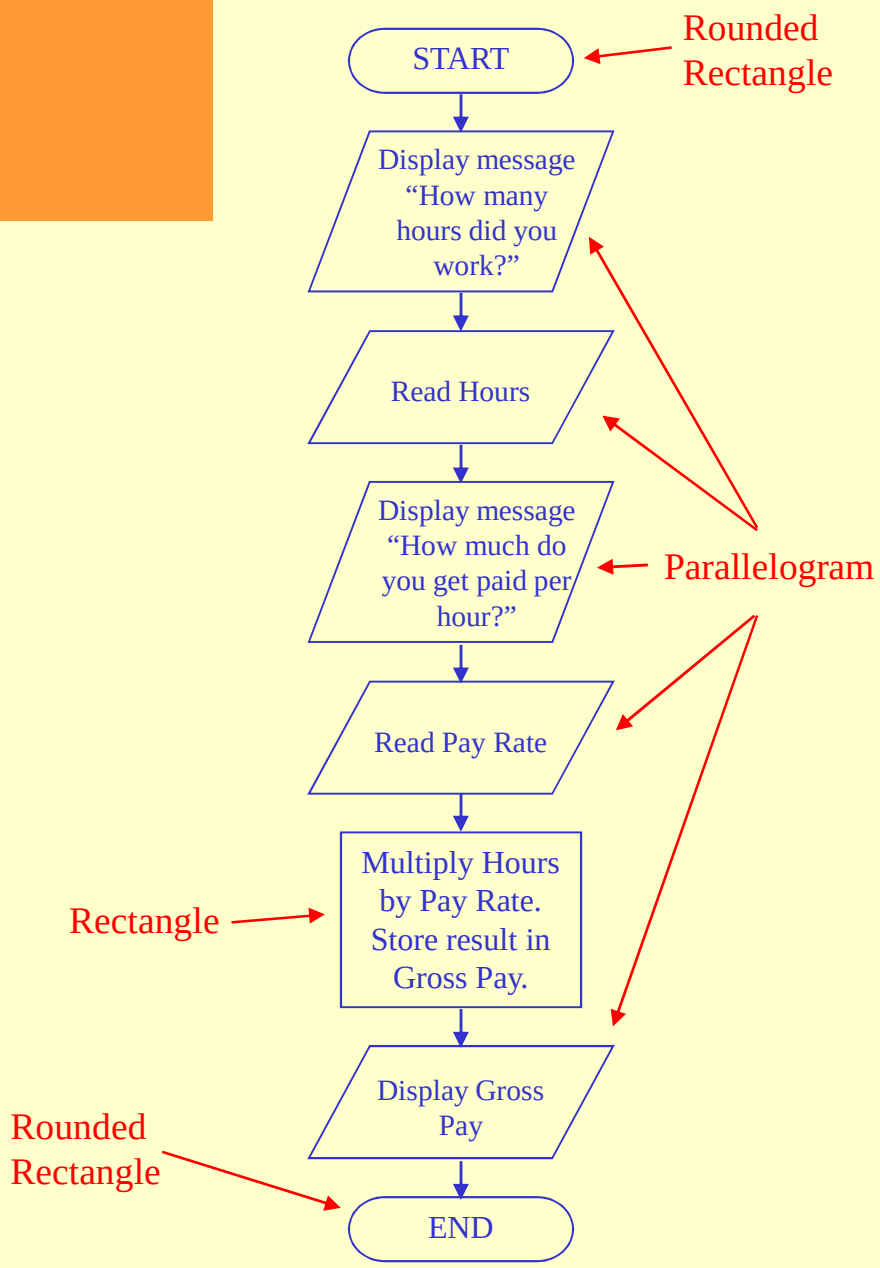


MEANING OF A FLOWCHART

A flowchart is a diagrammatic representation that illustrates the sequence of operations to be performed to get the solution of a problem. Flowcharts are generally drawn in the early stages of formulating computer solutions. Flowcharts facilitate communication between programmers and business people. These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems. Once the flowchart is drawn, it becomes easy to write the program in any high level language. Often we see how flowcharts are helpful in explaining the program to others. Hence, it is correct to say that a flowchart is a must for the better documentation of a complex program.

Basic Flowchart Symbols

- Notice there are three types of symbols in this flowchart:
  - rounded rectangles
  - parallelograms
  - a rectangle
- Each symbol represents a different type of operation.

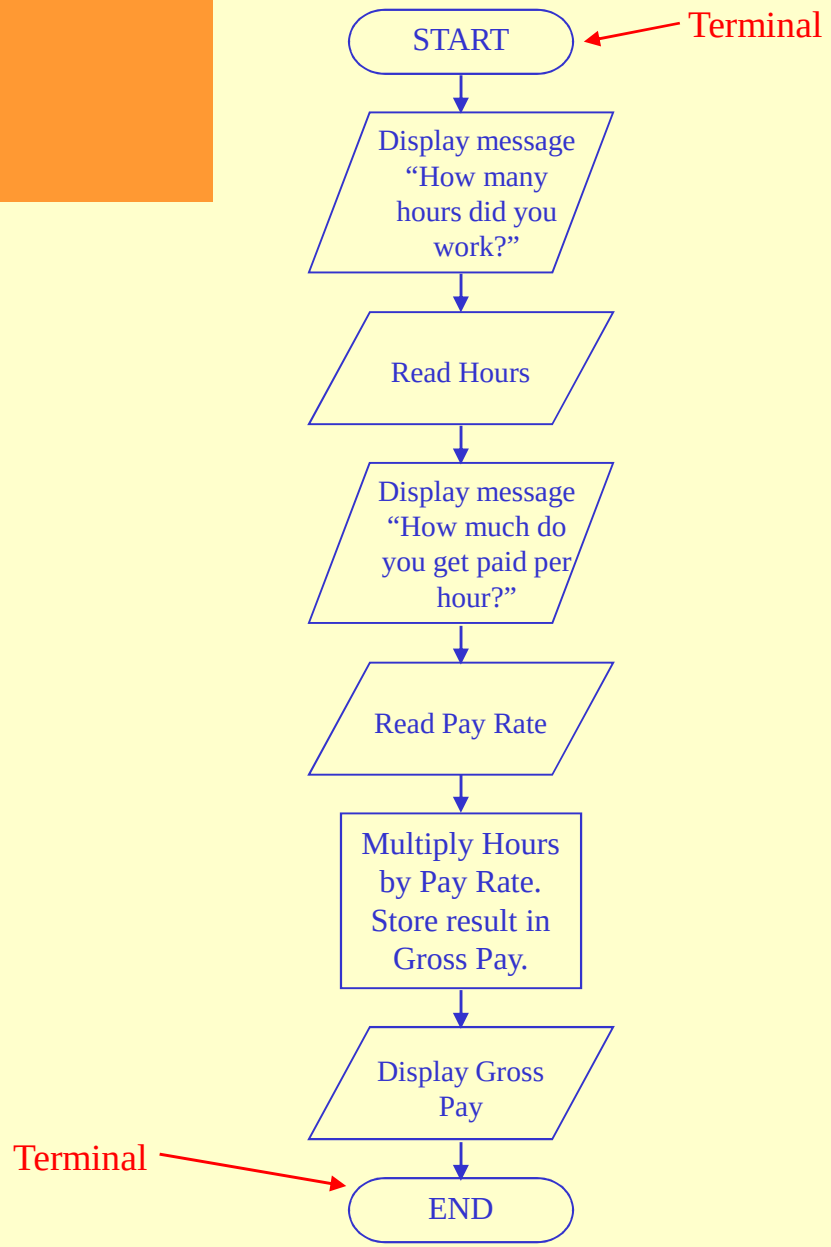


Basic Flowchart Symbols

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

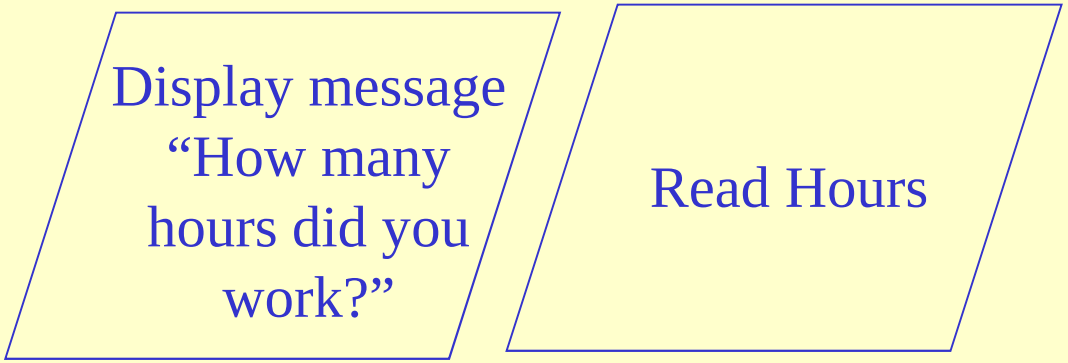
START

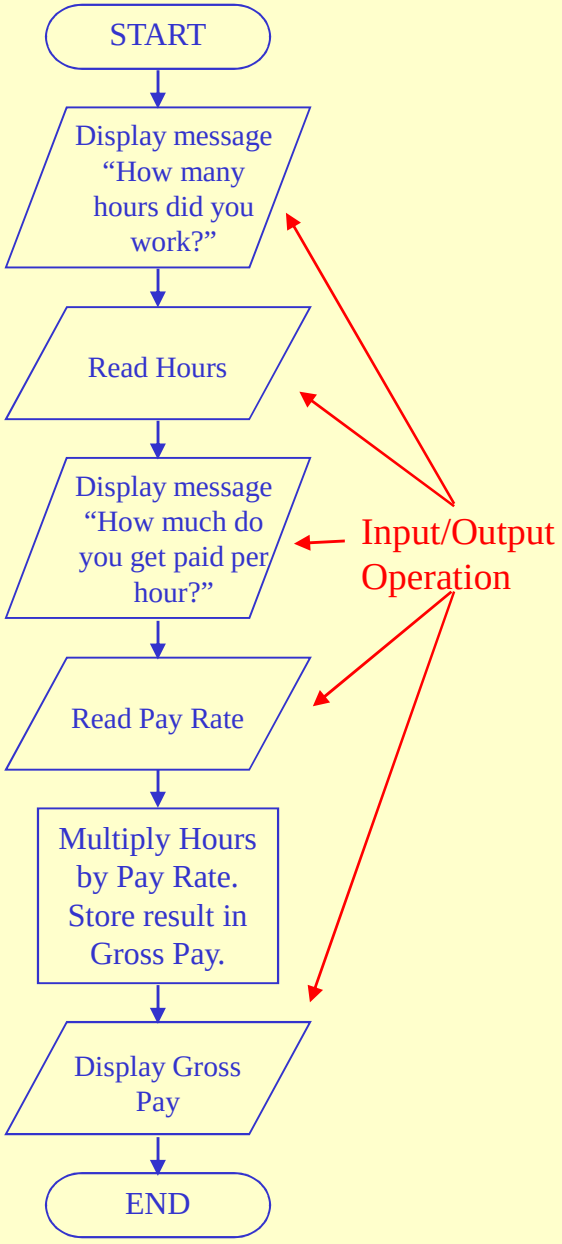
END



Basic Flowchart Symbols

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

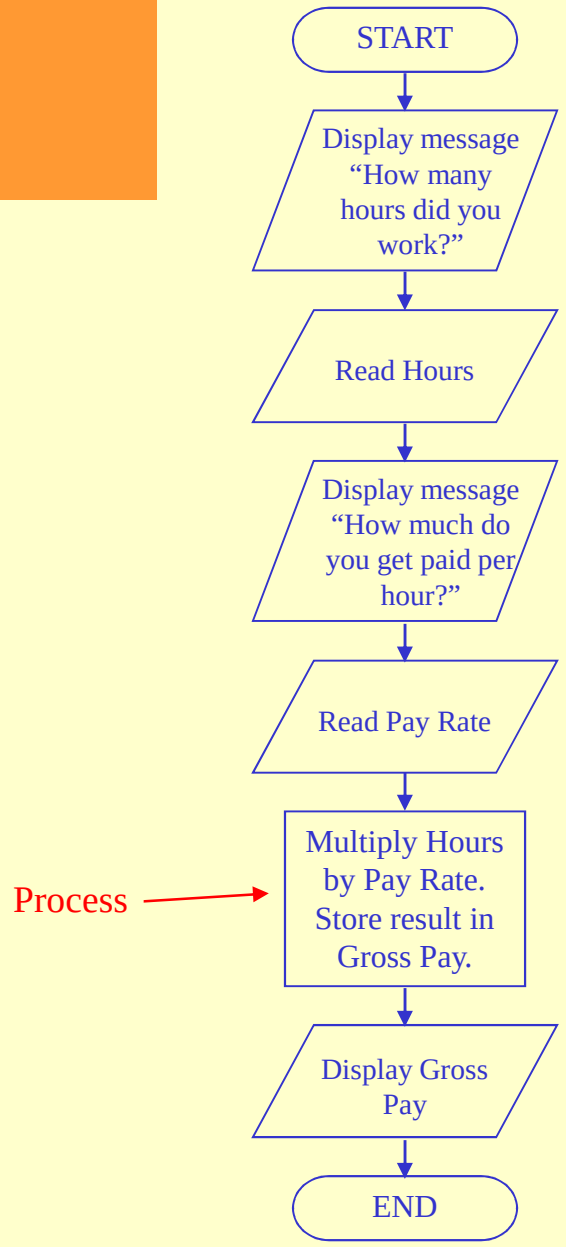




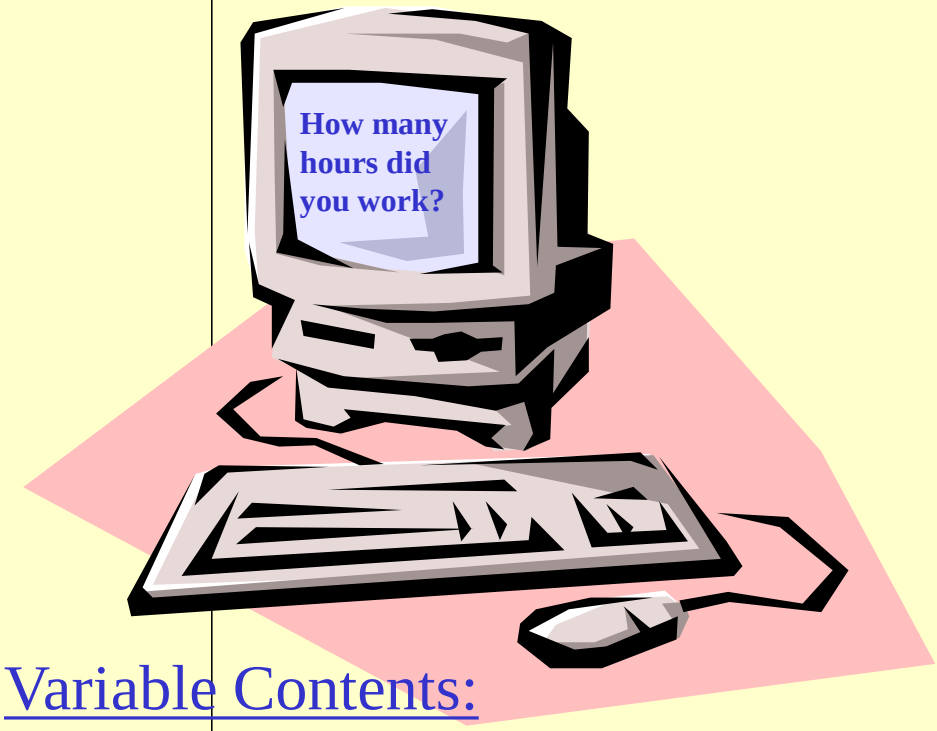
Basic Flowchart Symbols

- 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

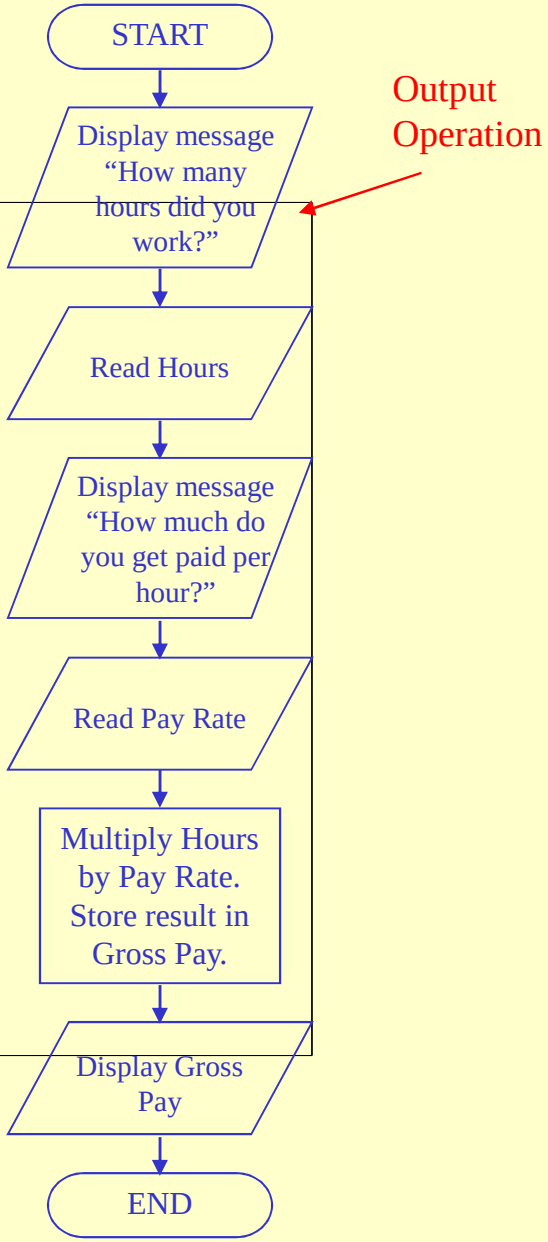


Variable Contents:

Hours: ?

Pay Rate: ?

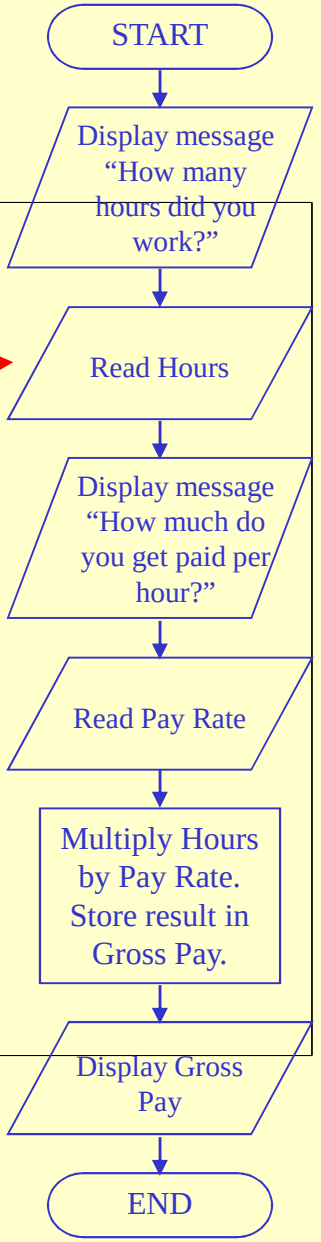
Gross Pay: ?



Stepping Through the Flowchart

Input Operation

(User types 40)



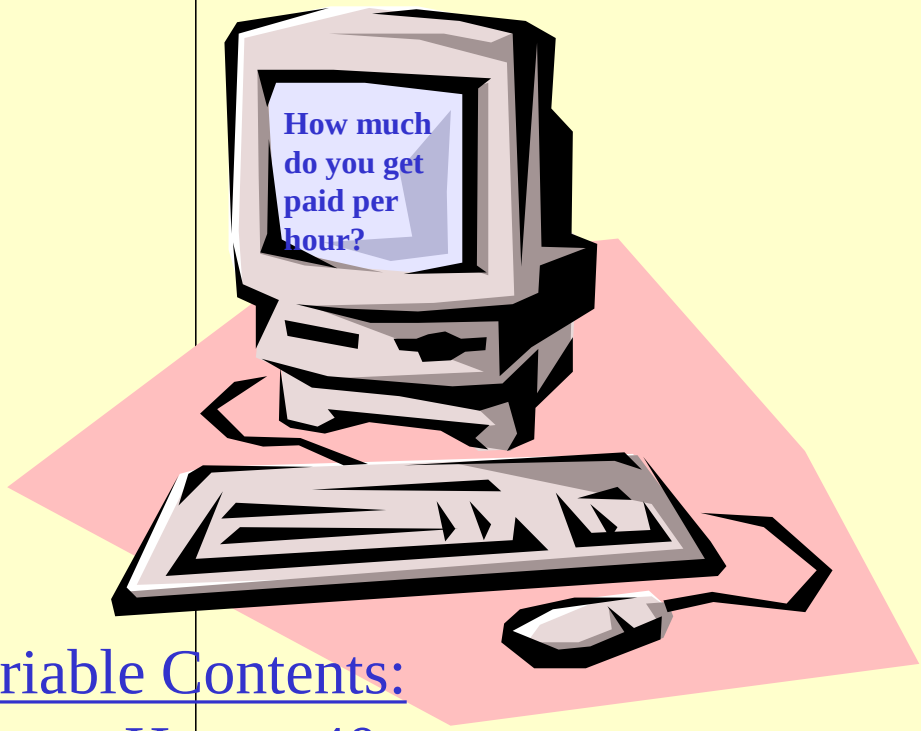
Variable Contents:

Hours: 40

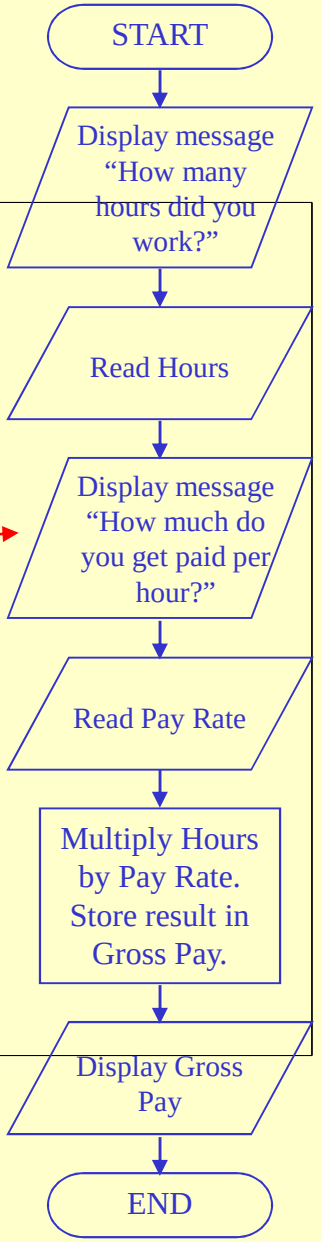
Pay Rate: ?

Gross Pay: ?

Stepping Through the Flowchart



Output Operation



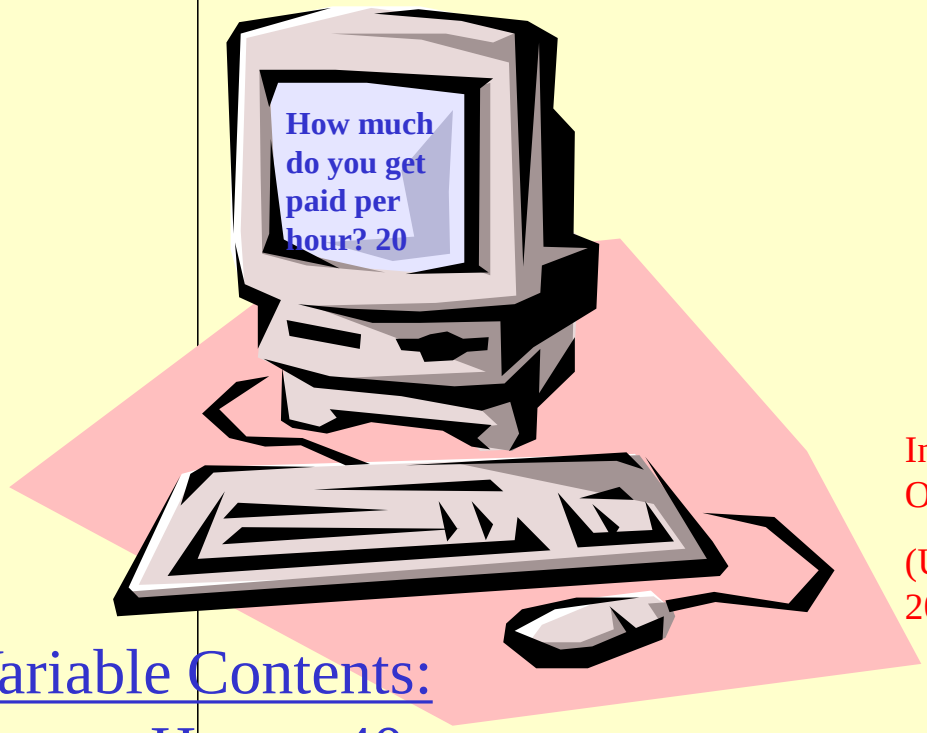
Variable Contents:

Hours: 40

Pay Rate: ?

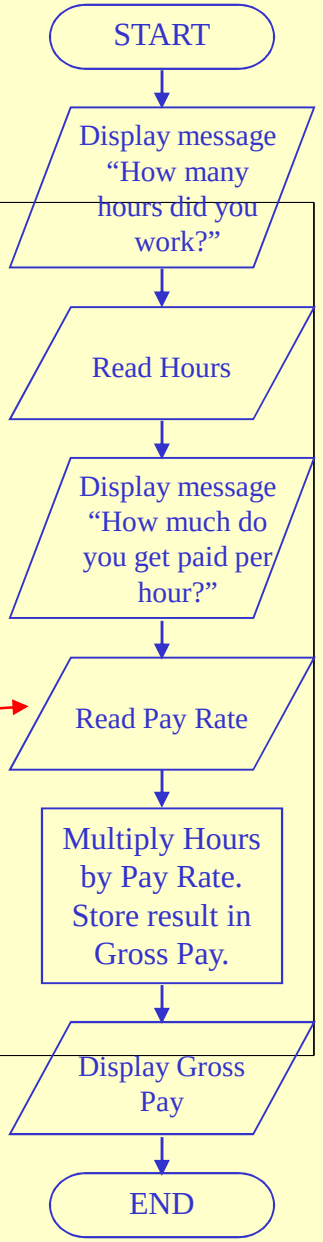
Gross Pay: ?

Stepping Through the Flowchart



Input Operation

(User types 20)



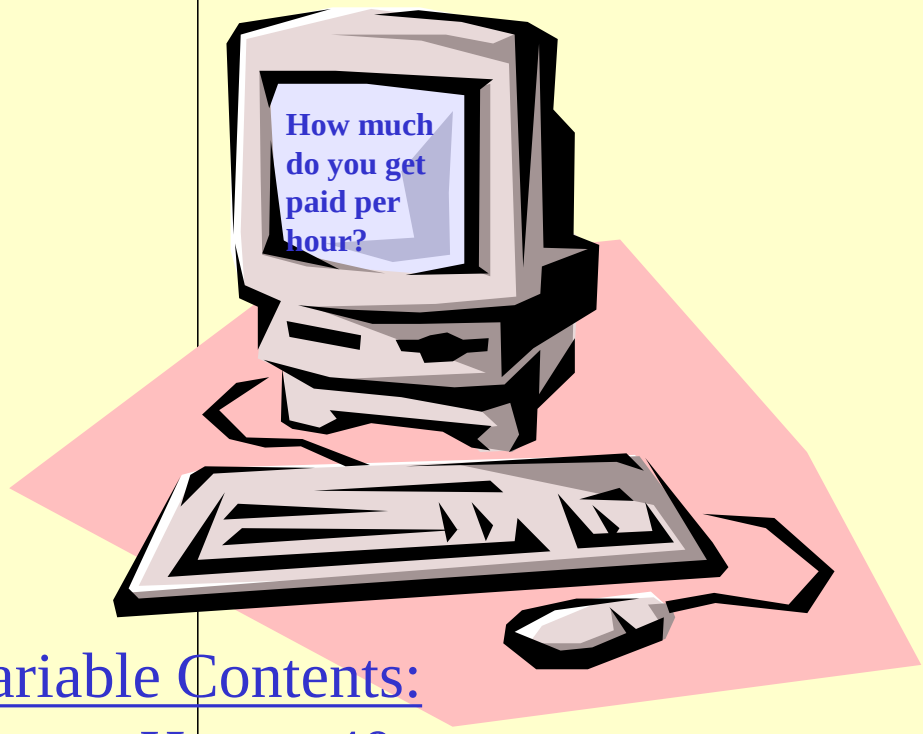
Variable Contents:

Hours: 40

Pay Rate: 20

Gross Pay: ?

Stepping Through the Flowchart



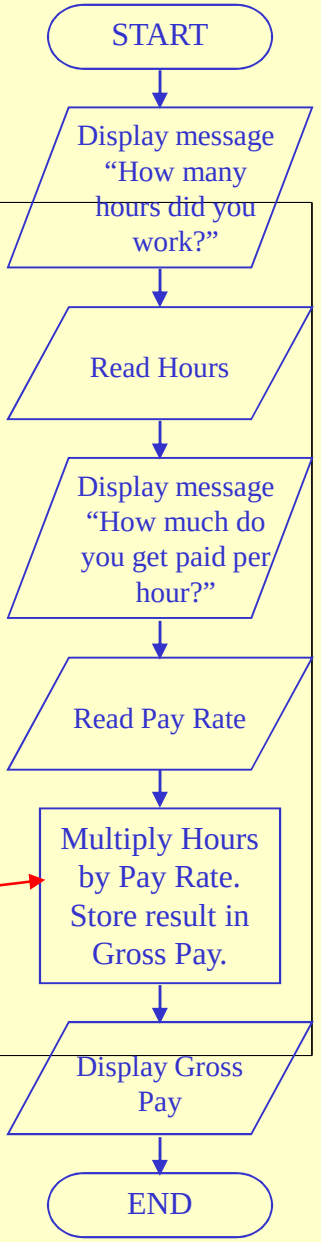
Variable Contents:

Hours: 40

Pay Rate: 20

Gross Pay: 800

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



Stepping Through the Flowchart

Variable Contents:

Hours: 40

Pay Rate: 20

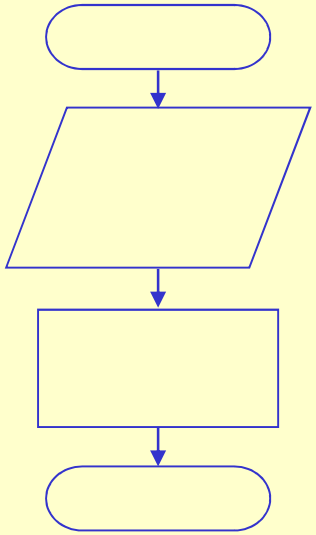
Gross Pay: 800

Four Flowchart Structures

- Sequence
- Decision
- Repetition
- Case

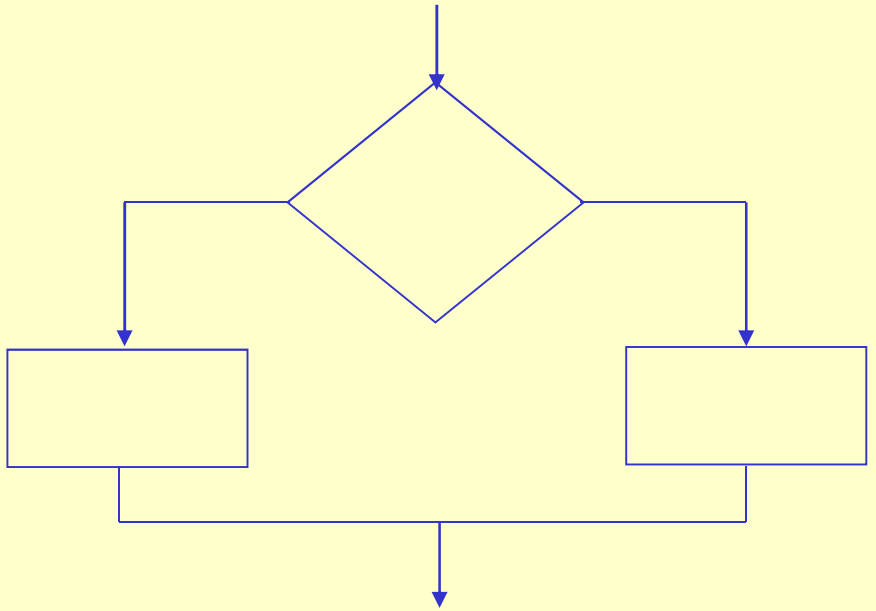
Sequence Structure

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



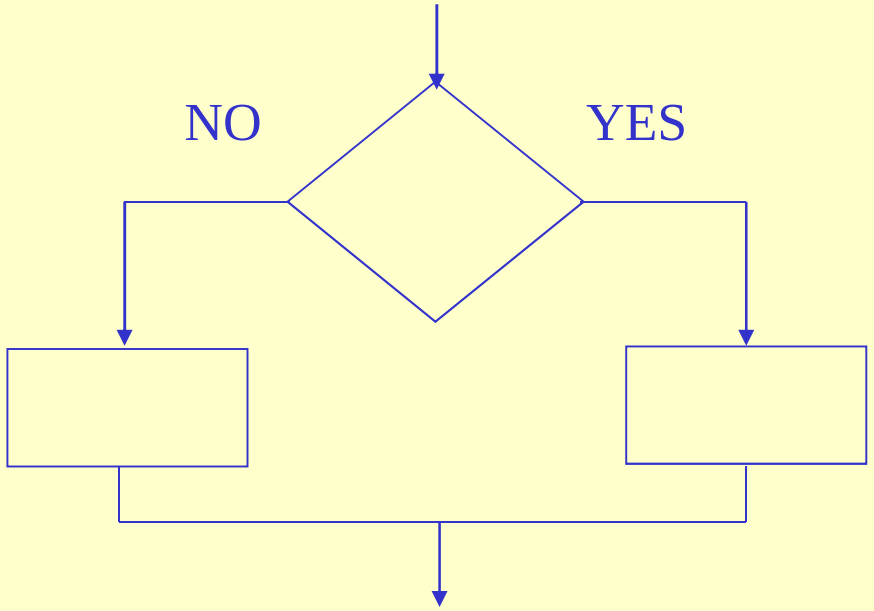
Decision Structure

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



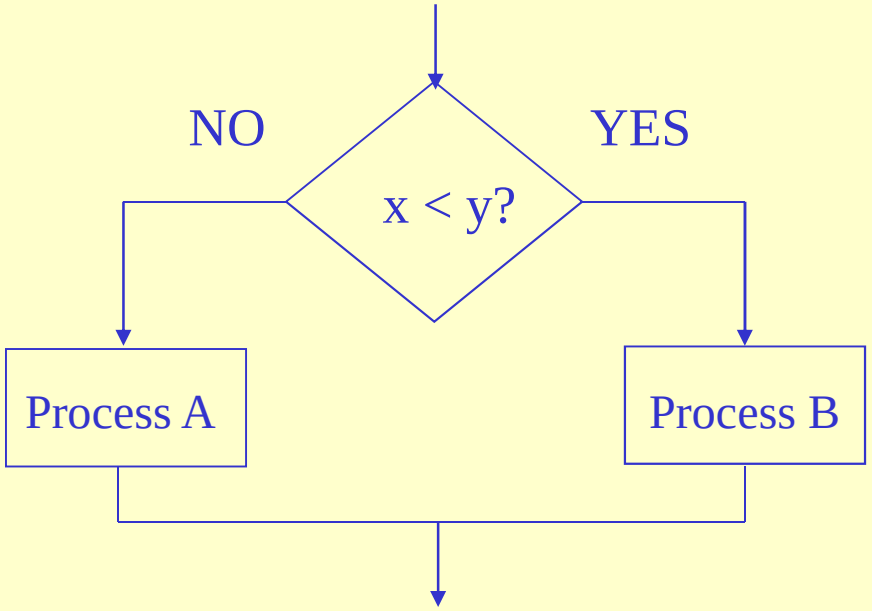
Decision Structure

- 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

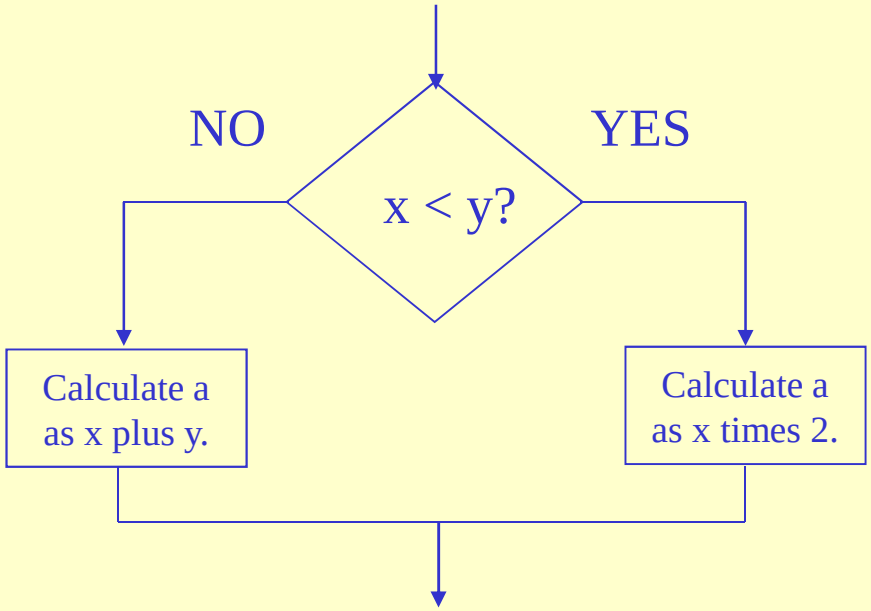
- In the flowchart segment below, the question “is x < y?” is asked. If the answer is no, then process A is performed. If the answer is yes, then process B is performed.



Decision Structure

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

Flowchart



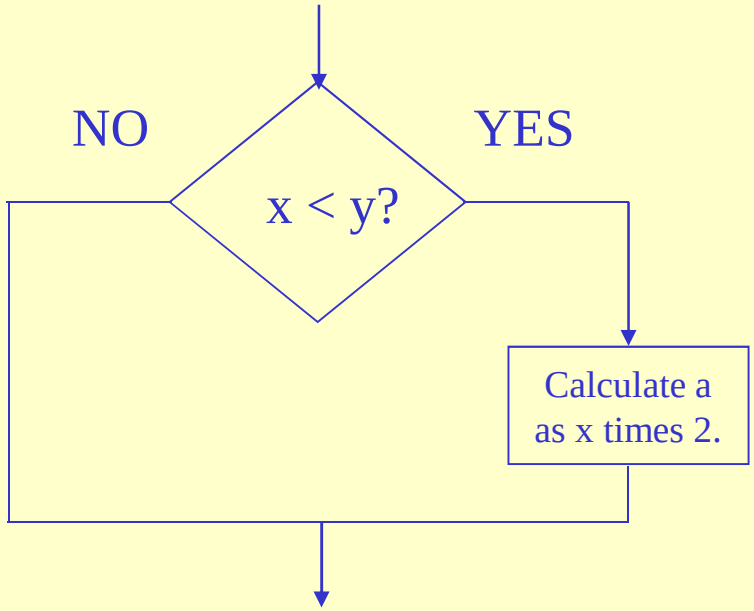
C++ Code

```
if (x < y)
        a = x * 2;
else
        a = x + y;
```

Decision Structure

- The flowchart segment below shows a decision structure with only one action to perform. It is expressed as an if statement in C++ code.

Flowchart

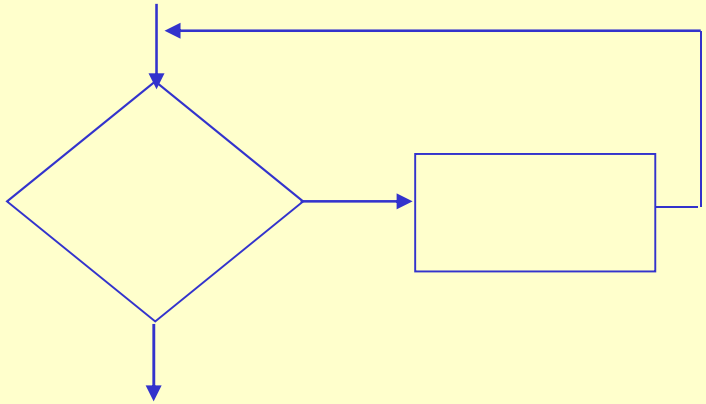


C++ Code

```
if (x < y)
        a = x * 2;
```

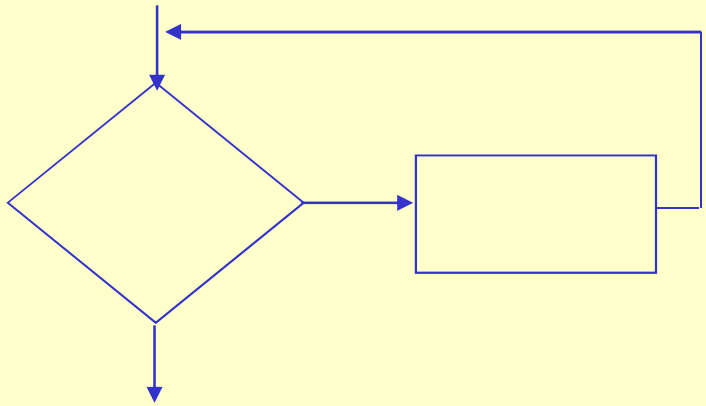
Repetition Structure

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



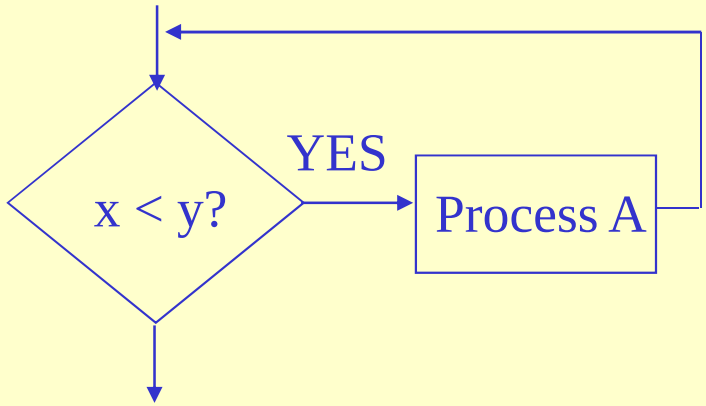
Repetition Structure

- 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

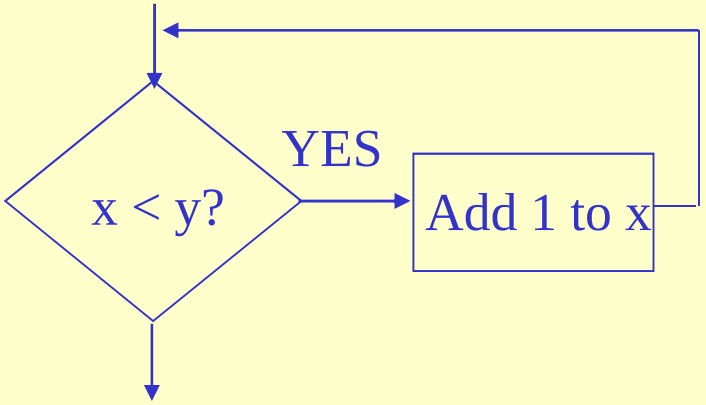
- In the flowchart segment, the question “is x < y?” is asked. If the answer is yes, then Process A is performed. The question “is x < y?” is asked again. Process A is repeated as long as x is less than y. When x is no longer less than y, the repetition stops and the structure is exited.



Repetition Structure

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

Flowchart

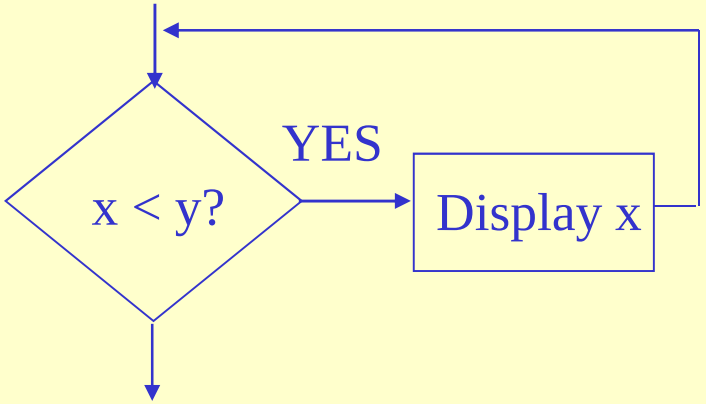


C++ Code

```
while (x < y)
       x++;
```

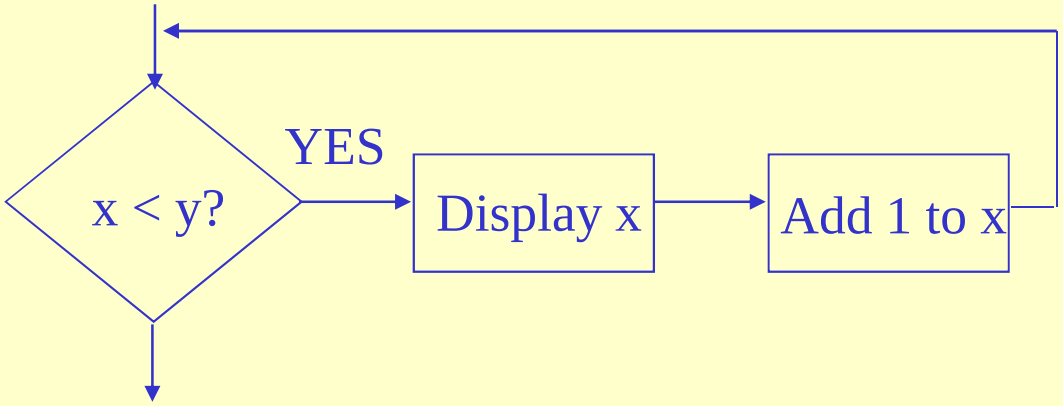
Controlling a Repetition Structure

- 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?



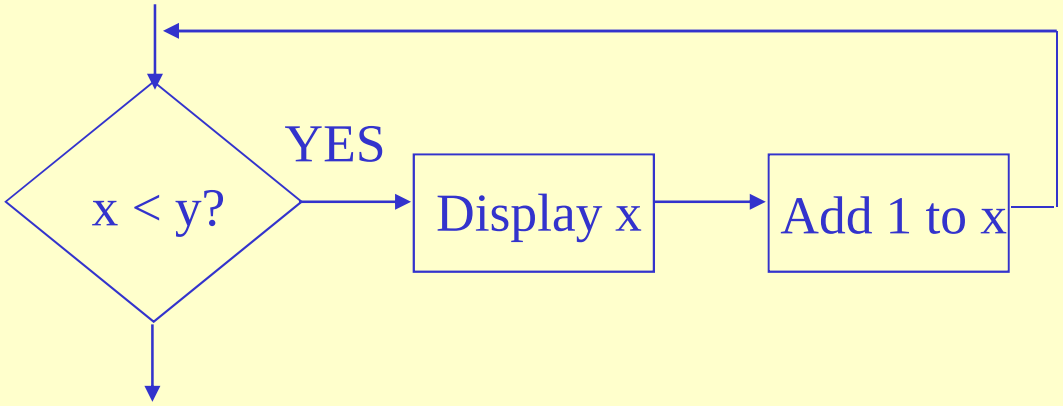
Controlling a Repetition Structure

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



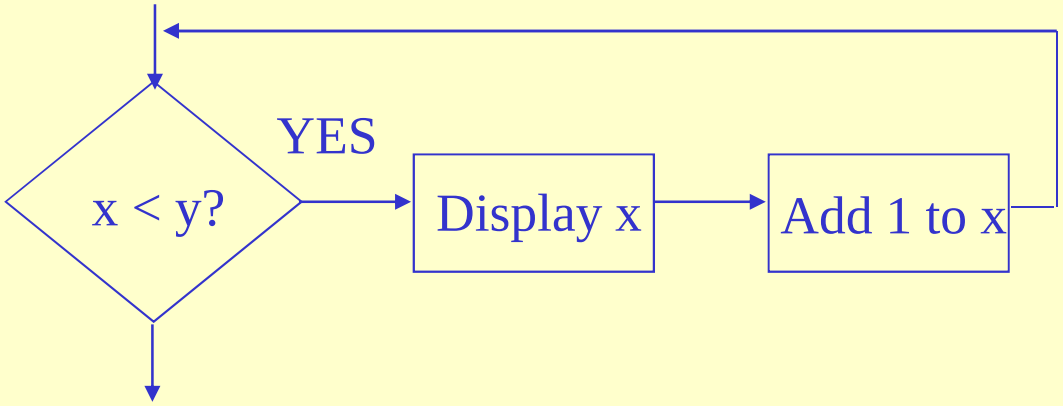
A Pre-Test Repetition Structure

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



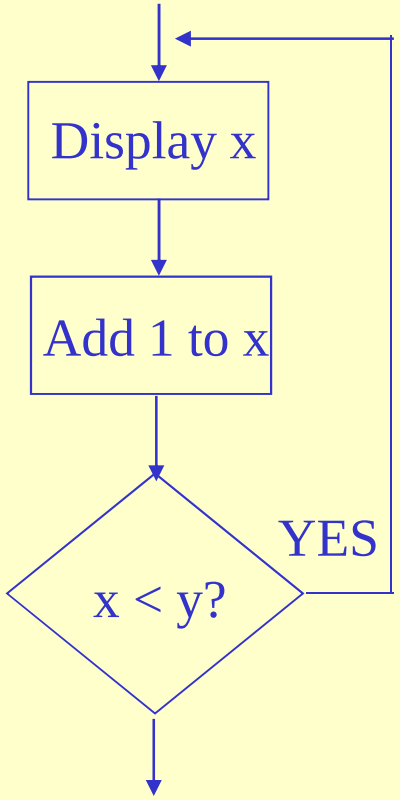
A Pre-Test Repetition Structure

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



A Post-Test Repetition Structure

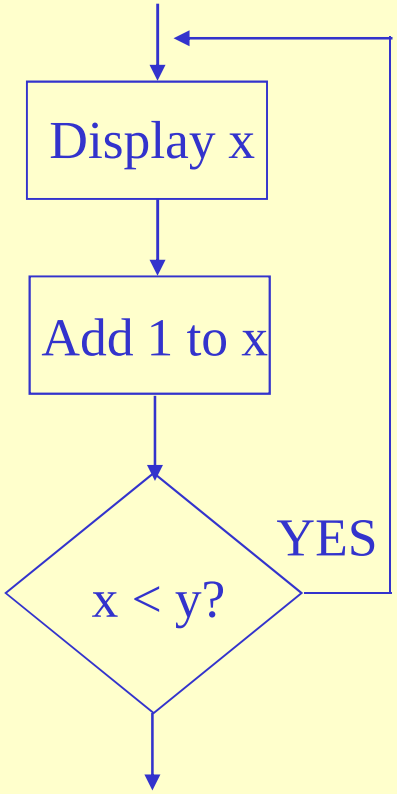
- This flowchart segment shows a post-test repetition structure.
- The condition is tested *AFTER* the actions are performed.
- A post-test repetition structure always performs its actions at least once.



A Post-Test Repetition Structure

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

Flowchart

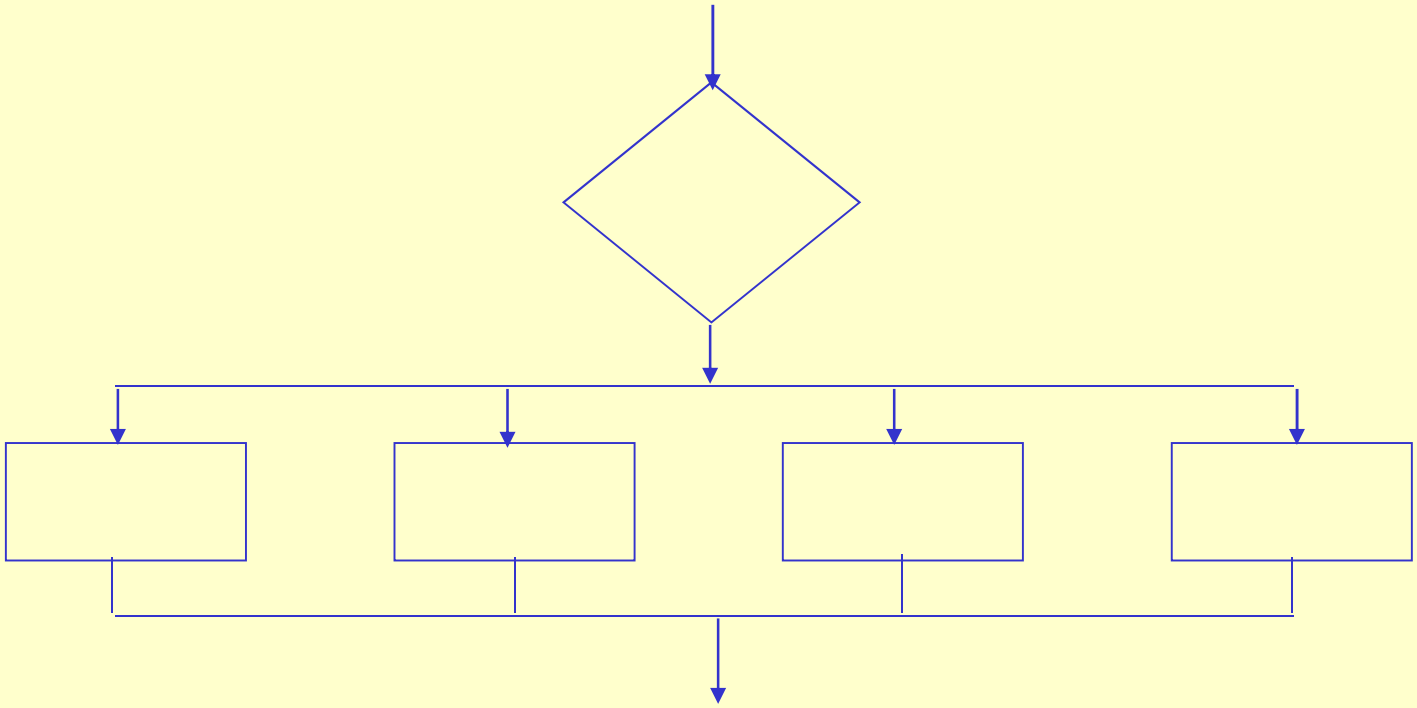


C++ Code

```
do
{
        cout << x << endl;
        x++;
} while (x < y);
```

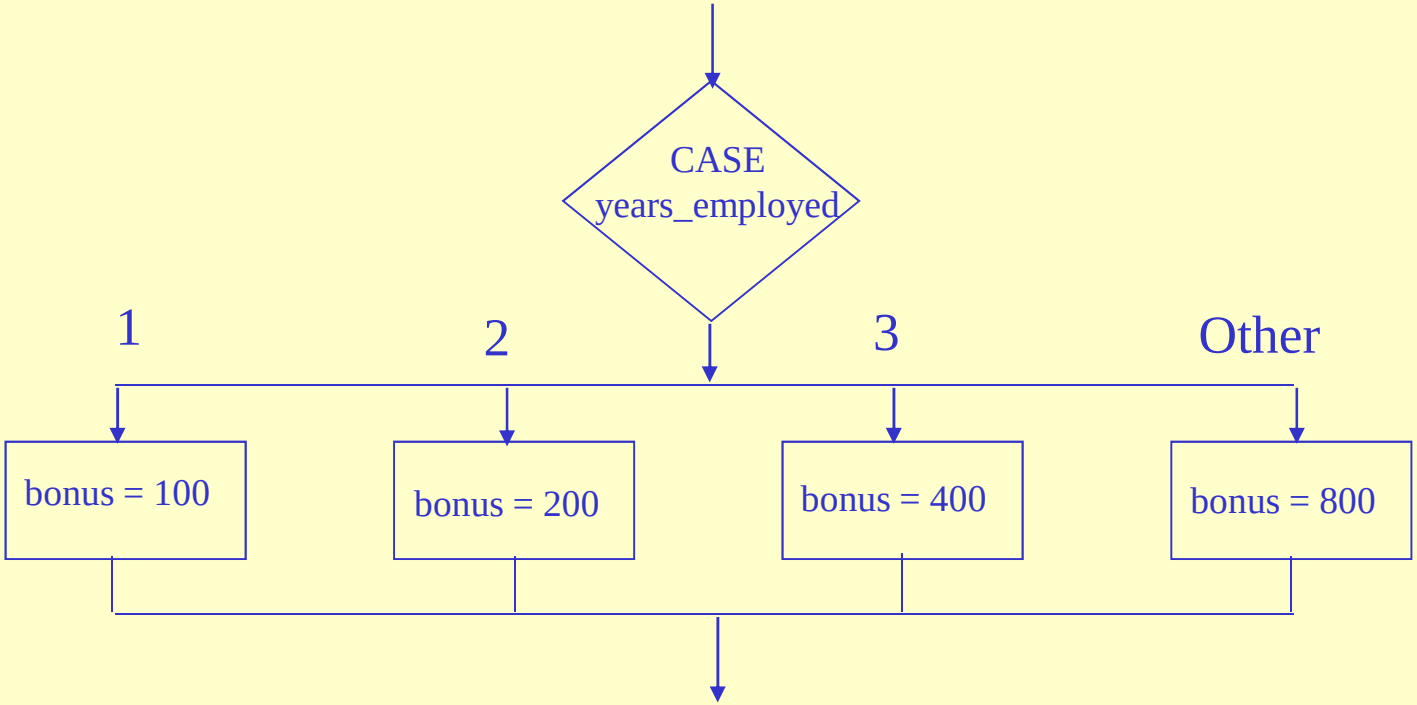
Case Structure

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

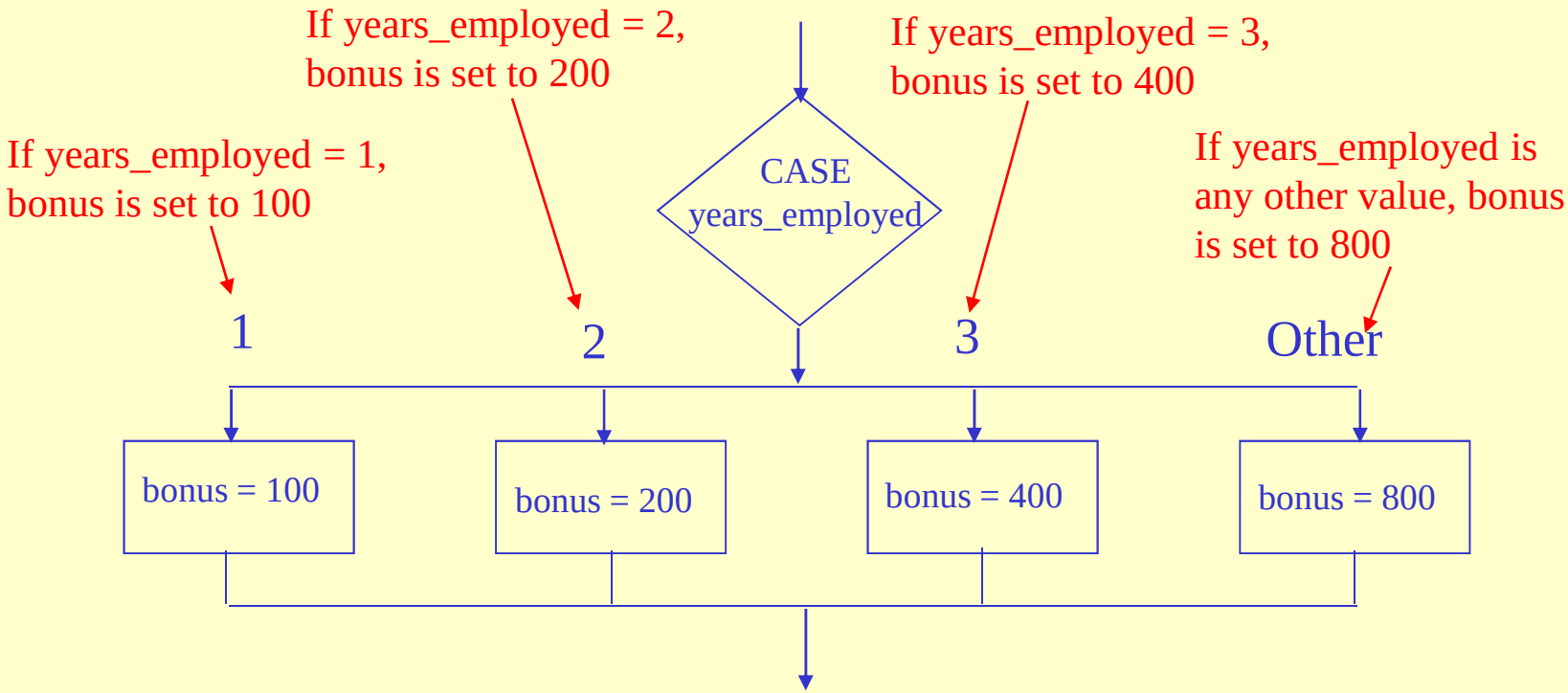


Case Structure

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

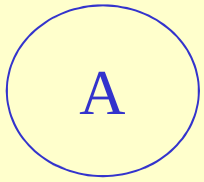


Case Structure



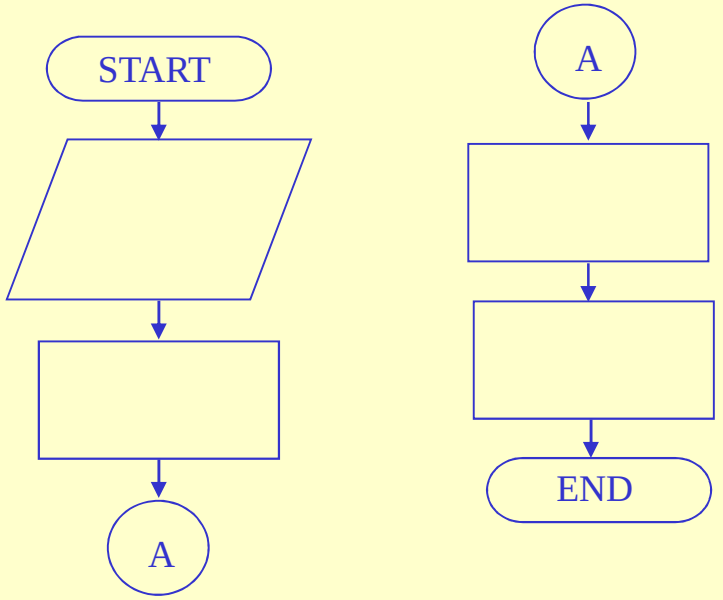
Connectors

- Sometimes a flowchart will not fit on one page.
- A connector (represented by a small circle) allows you to connect two flowchart segments.



Connectors

•The “A” connector indicates that the second flowchart segment begins where the first segment ends.



ADVANTAGES OF USING FLOWCHARTS

The benefits of flowcharts are as follows:

- Communication: Flowcharts are better way of communicating the logic of a system to all concerned.
- Effective analysis: With the help of flowchart, problem can be analysed in more effective way.
- Proper documentation: Program flowcharts serve as a good program documentation, which is needed for various purposes.
- Efficient Coding: The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- Proper Debugging: The flowchart helps in debugging process.
- Efficient Program Maintenance: The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

LIMITATIONS OF USING FLOWCHARTS

- Complex logic: Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- Alterations and Modifications: If alterations are required the flowchart may require re-drawing completely.
- Reproduction: As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.
- The essentials of what is done can easily be lost in the technical details of how it is done.