Lecture 2

Design Solutions Description

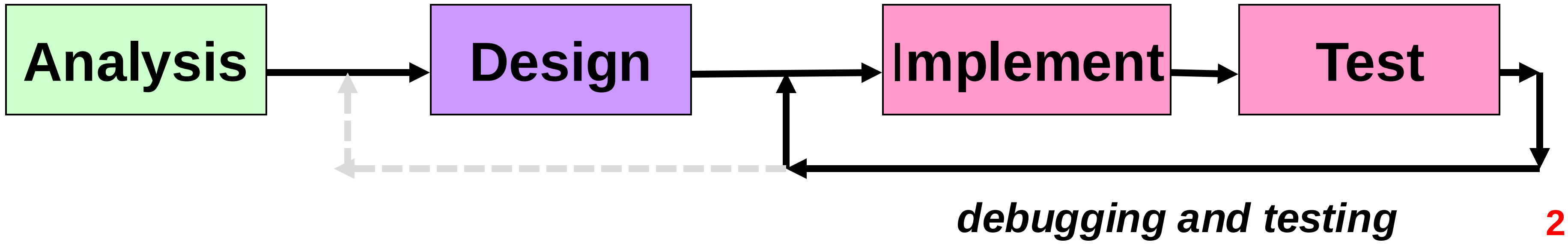
Objective: This lecture explains the use of pseudo-code and Flowchart in describing algorithms solution. It demonstrates how a good a design description leads to a successful and fast implementation

See the Notes pages (section) ⬇ of this presentation

I. Software Development Process

❖ With non-trivial problems, we can NOT just type in the code and always expect to solve the problem in hand. Instead, we need to:

1. Define clearly the problem
2. Analyse the problem and specify clearly its **requirements** along with any constraint
3. Design an **algorithm** solution: a sequence of steps that describe a solution to a problem
4. Code (implement) the algorithm
5. Test the code; Go to 4 or ….3 if needed

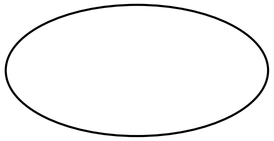
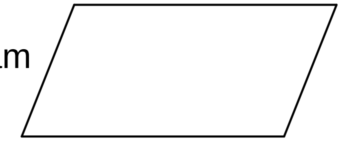
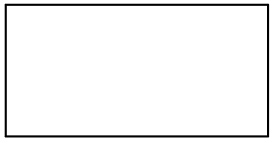
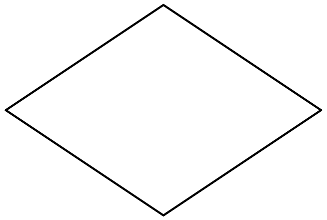
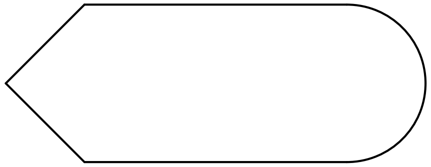
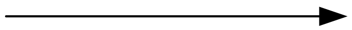


II. Design Documentation

- Design solutions (algorithms) can be described using either **Pseudo-code or Flowchart**

- **Pseudo-code:** is very similar to everyday English. It is an artificial and informal language that helps programmers develop and describe algorithms
- **Flowchart:** is a diagram, a graphical representation of the sequence of operations in an algorithm
  - Each operation is represented by a symbol
  - Symbols are connected with arrows to specify the sequence of operations

III. Flowchart Main Symbols

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes an operation to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the algorithm

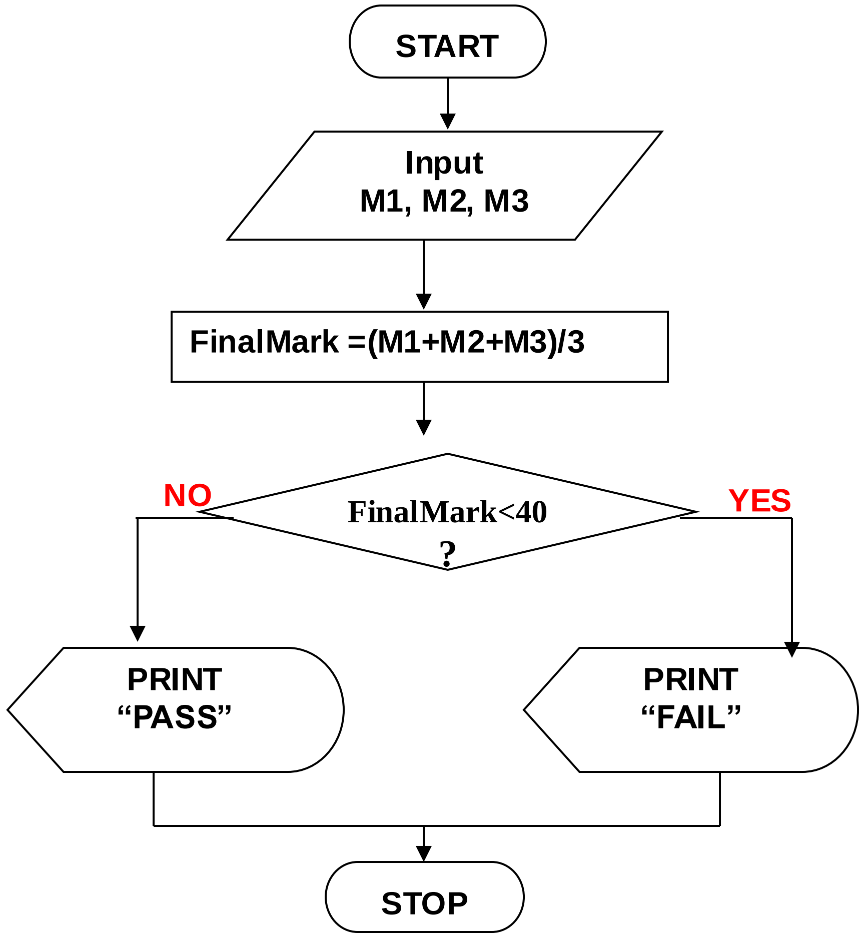
Example

- **Example 1**: write an algorithm to determine a student's final mark and indicate whether it is passing or failing. The final mark is calculated as the average of three marks
- **Solution:**
  - **Data Model**
    - **M1, M2, M3**: the three marks of the student
    - **FinalMark**: student's final mark
  - **Detailed Algorithm in pseudo-code:**

```
1. Input M1, M2, M3
2. FinalMark = (M1+M2+M3)/3
3. if (FinalMark  < 40) then
       Print "FAIL"
   Else
       Print "PASS"
   endif
```

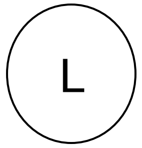
…Continued

Detailed Algorithm in Flowchart

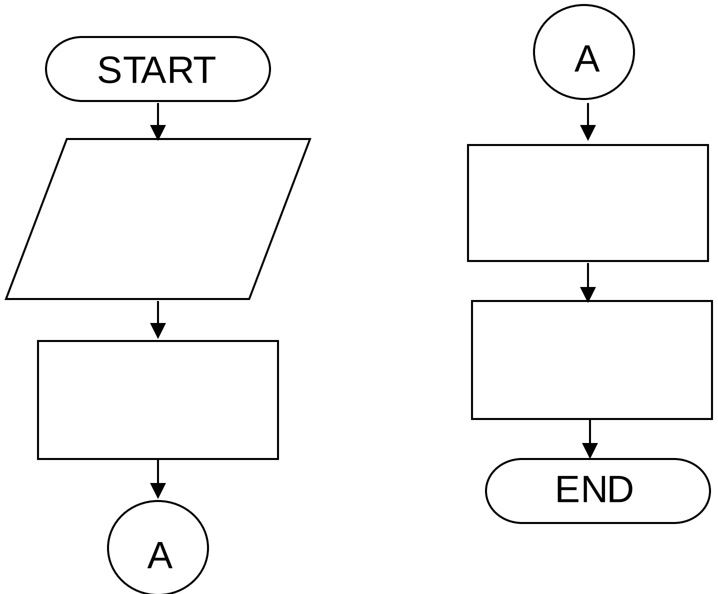


Flowchart Symbol: Connector

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

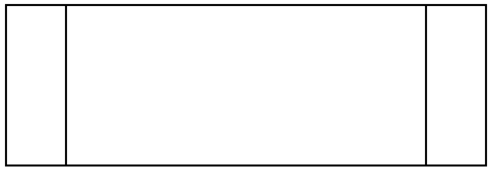


- The "A" connector in the diagrams below indicates that the second flowchart segment begins where the first segment ends

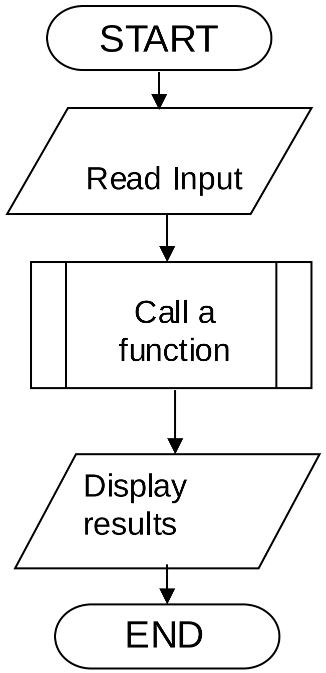


Flowchart Symbol for Modules / functions

- 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

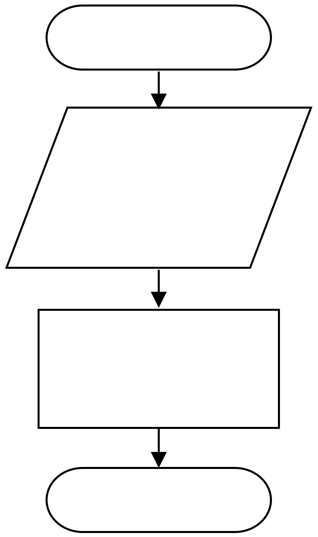


IV. Four Flowchart Structures

1. Sequence
2. Decision
3. Repetition
4. Case

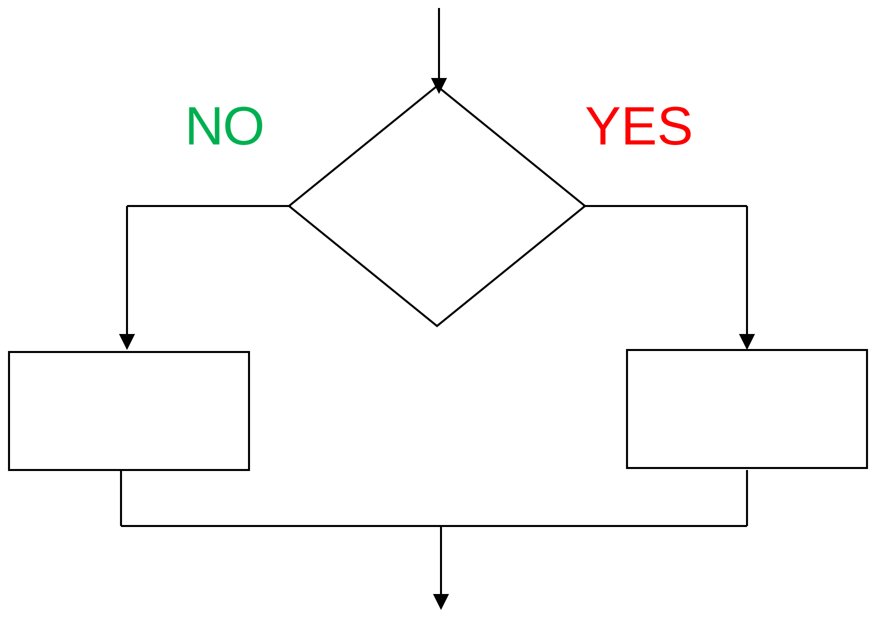
1. Sequence Structure

❖ a series of actions are performed in sequence



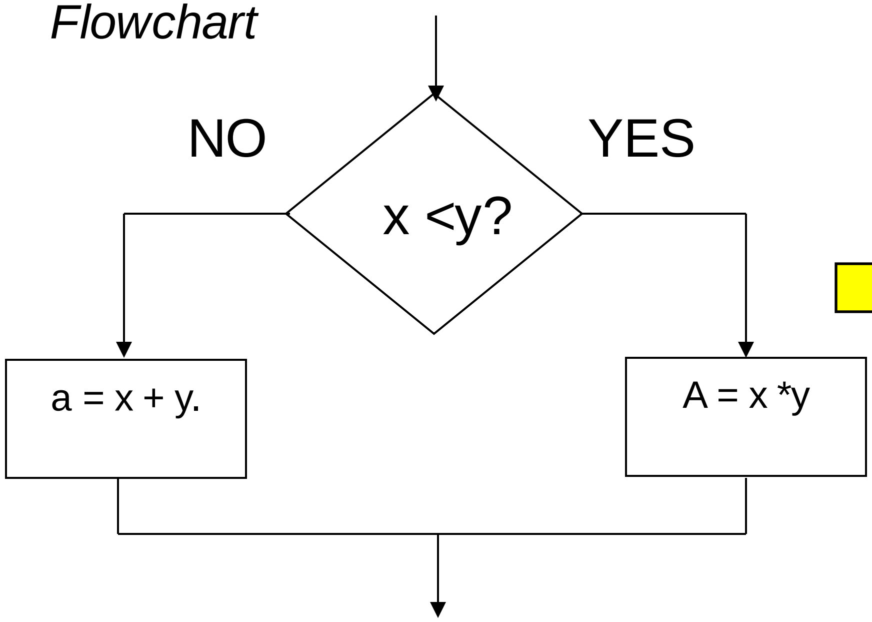
2. Decision Structure

- The **diamond** indicates a yes/no question. If the answer to the question is yes, the flow follows the YES path. If the answer is no, the flow follows the NO path



Decision Structure to C code: examples

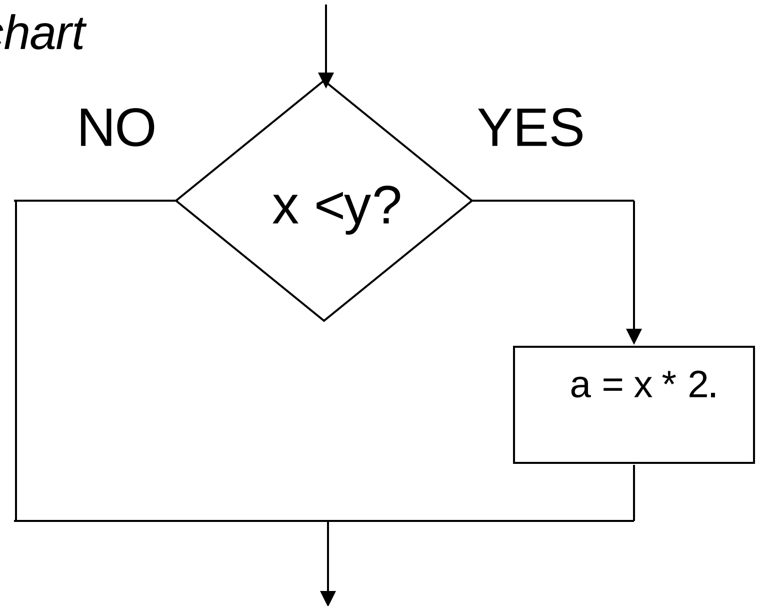
Flowchart



C Code

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

Flowchart

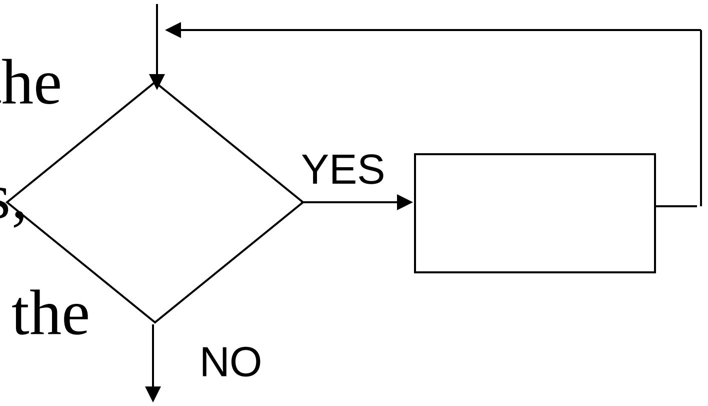


C Code

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

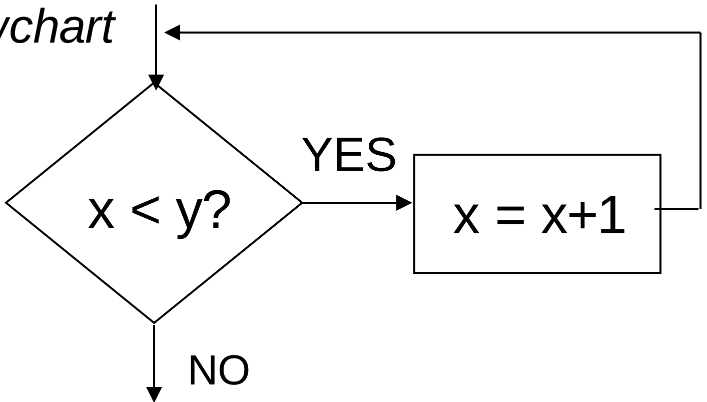
3. Repetition Structure

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



- **Example**

Flowchart

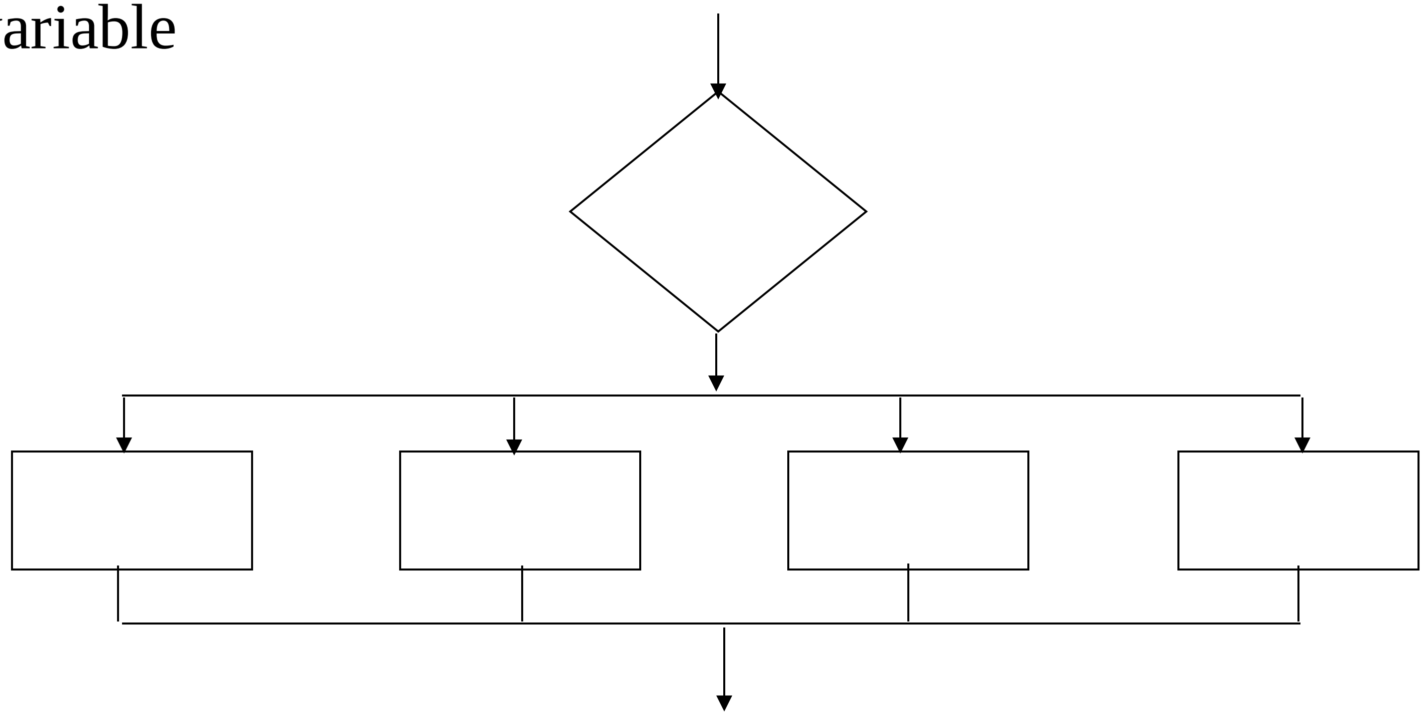


C Code

```
while (x < y)
        x = x+ 1;
```

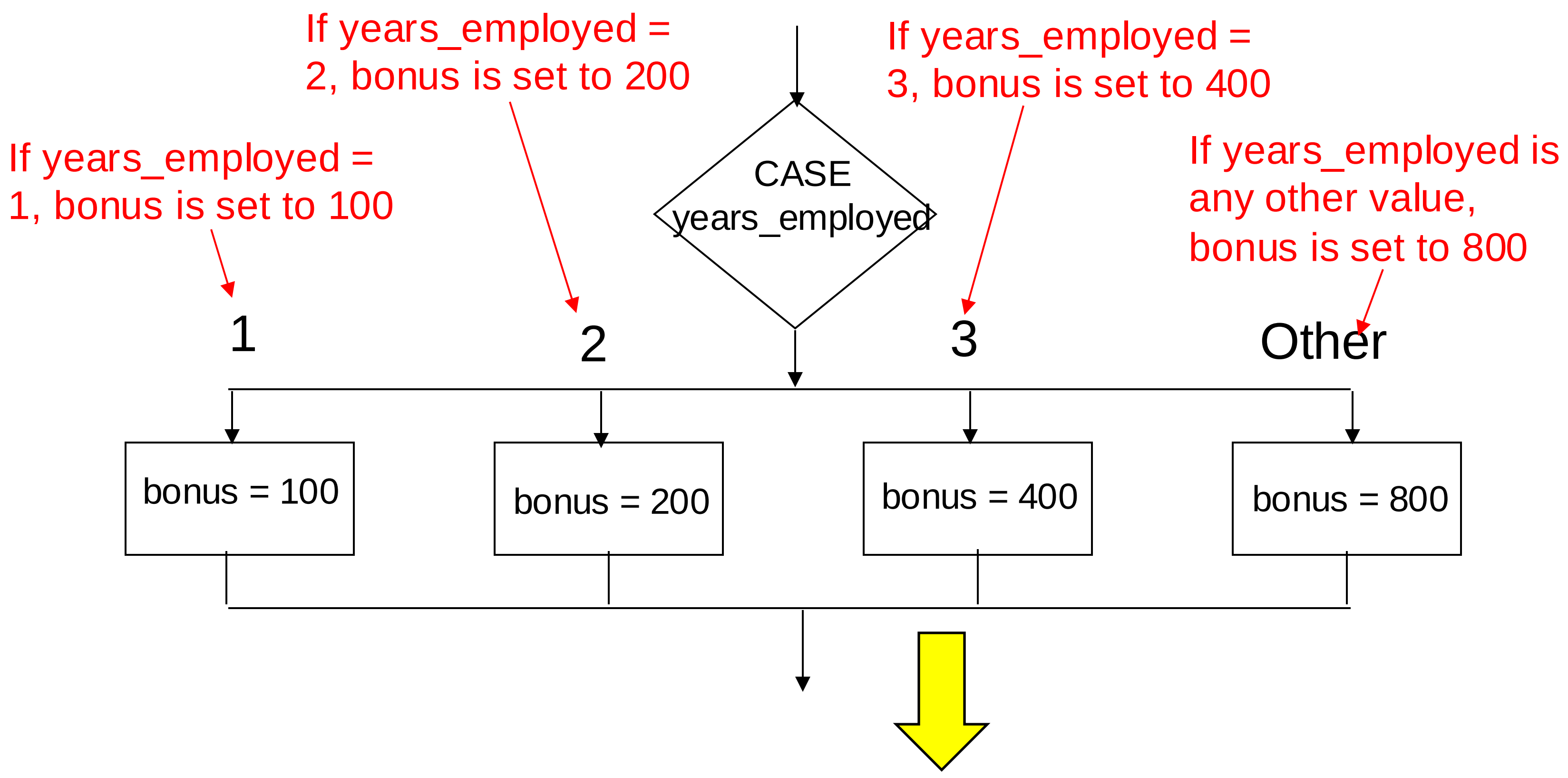
4. Case Structure

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



- The example case structure in the next slide indicates actions to perform depending on the value in *years_employed*

Case structure: example

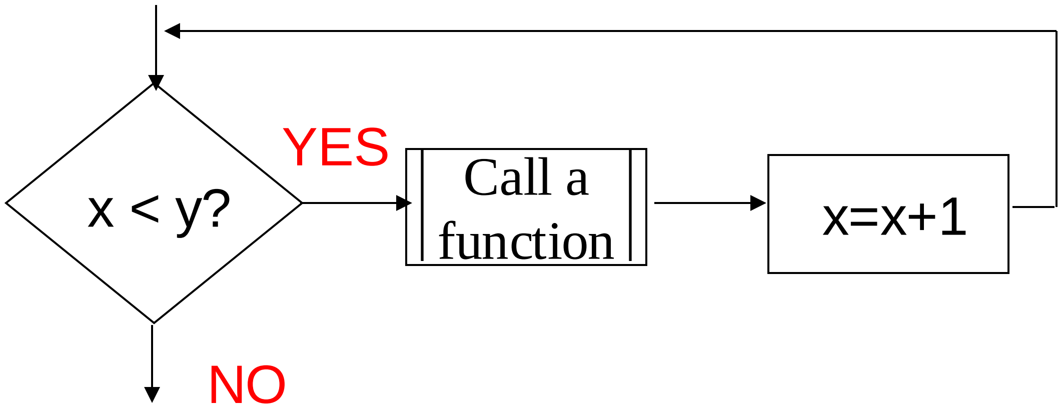


If years_employed = 2, bonus is set to 200
If years_employed = 3, bonus is set to 400
If years_employed = 1, bonus is set to 100
CASE years_employed
If years_employed is any other value, bonus is set to 800
1
2
3
Other
bonus = 100
bonus = 200
bonus = 400
bonus = 800

```
switch(years_employed )
{
case 1: bonus=100; break;
case 2 : bonus=200; break;
case 3 : bonus=400; break;
default: 800};
```

Combining structures

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



Conclusion

- A very good design flowchart can be translated on the fly into a programming language. This can be done by identifying the type of structure in the flowchart and associate to it the corresponding code (e.g. C) as shown in the previous slides