

CS 199 Computer Programming



Spring 2018

Lecture 5

Control Statements

Control Structures

□ 3 control structures

▶ **Sequence structure**

- ▶ Programs executed sequentially by default

▶ **Branch structure**

- ▶ Unconditional branch

- ▶ goto

- ▶ Control proceeds dependent on conditions

- ▶ if, if/else, switch

▶ **Repetition structures (loop instructions)**

- ▶ Control repeated until condition met

- ▶ while, do/while, for

Control Structures (cont.)

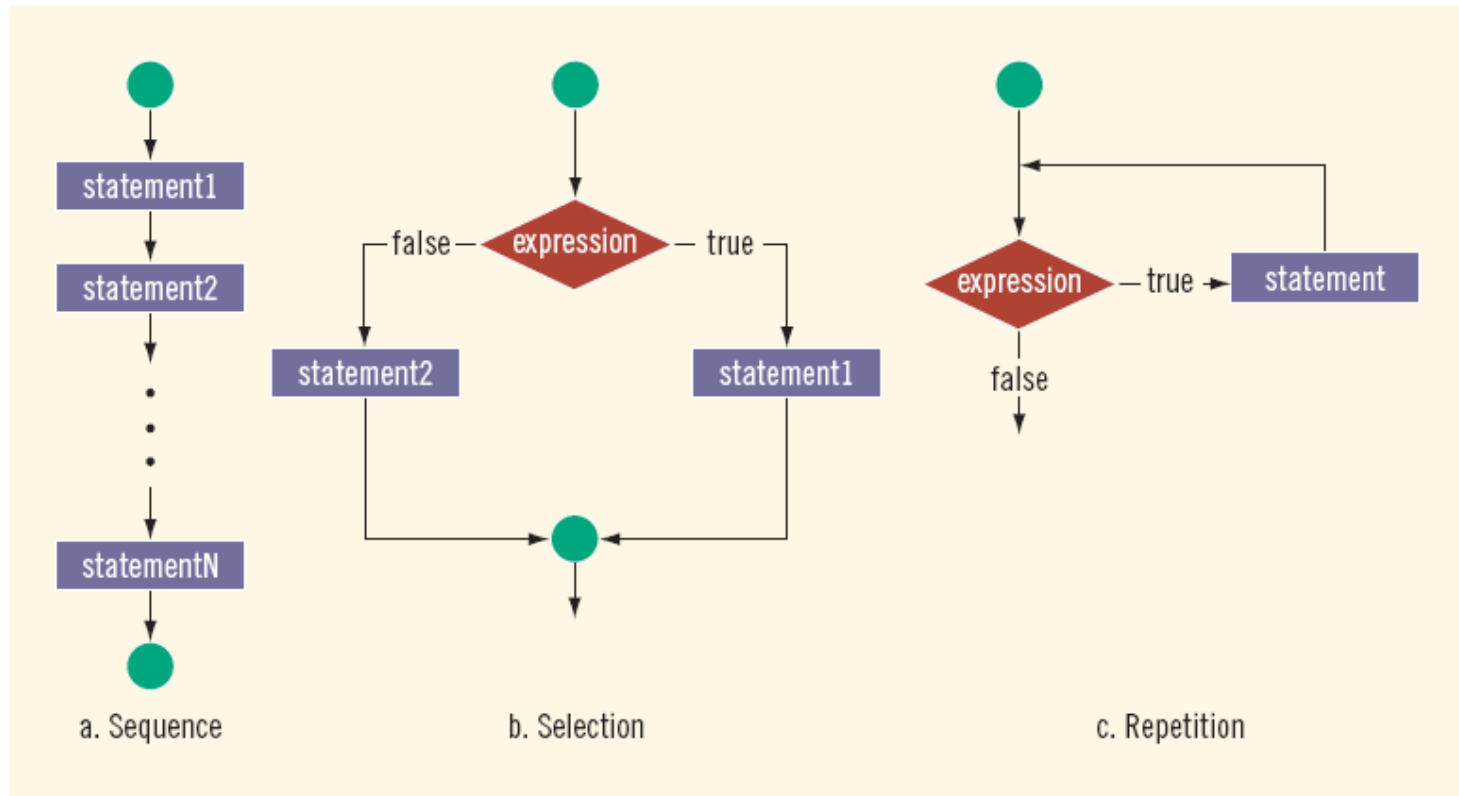


FIGURE 4-1 Flow of execution

Conditional Statements

- A conditional statement allows the program to make a decision or comparison and then select one of two paths, depending on the result of the comparison.
- Two constructs
 - if statement
 - if
 - if-else
 - if-else-if
 - Select case statement

Basic If-Statement

- Syntax

if (expression) then

body

End if

- Semantics: if the **expression** is true then execute **body**

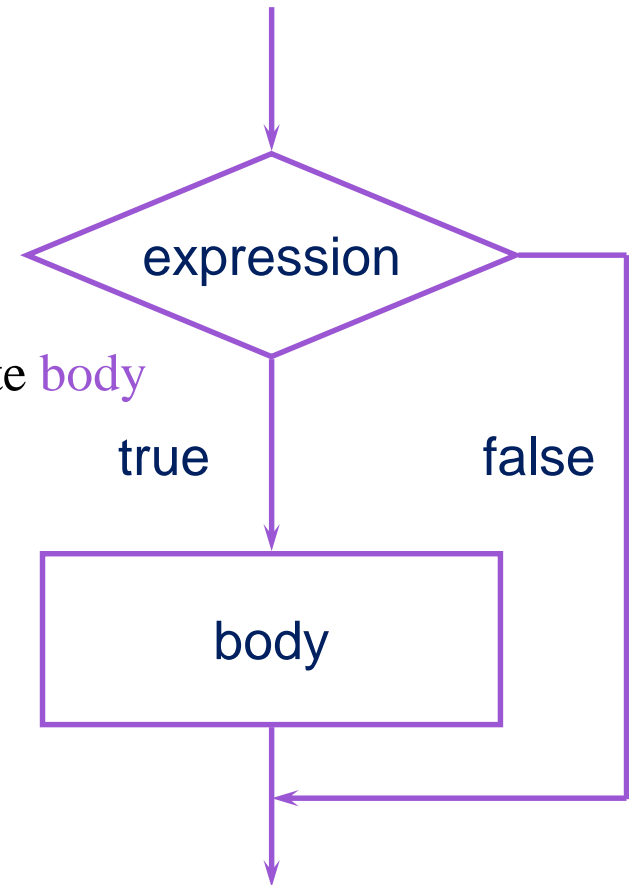
- **Body** is either a single statement or a group of statements.

- Example 1:

if ($v > 0$) then

$v = 0$

end if




Condition

- Condition” is a logical expression that evaluates to **true** or **false**. It could be a **relational** or **Boolean** expression.
- Simple conditions are built using
 - Relational Operators: $<$, $>$, $>=$, $<=$
 - Equality Operators: $==$, $/=$

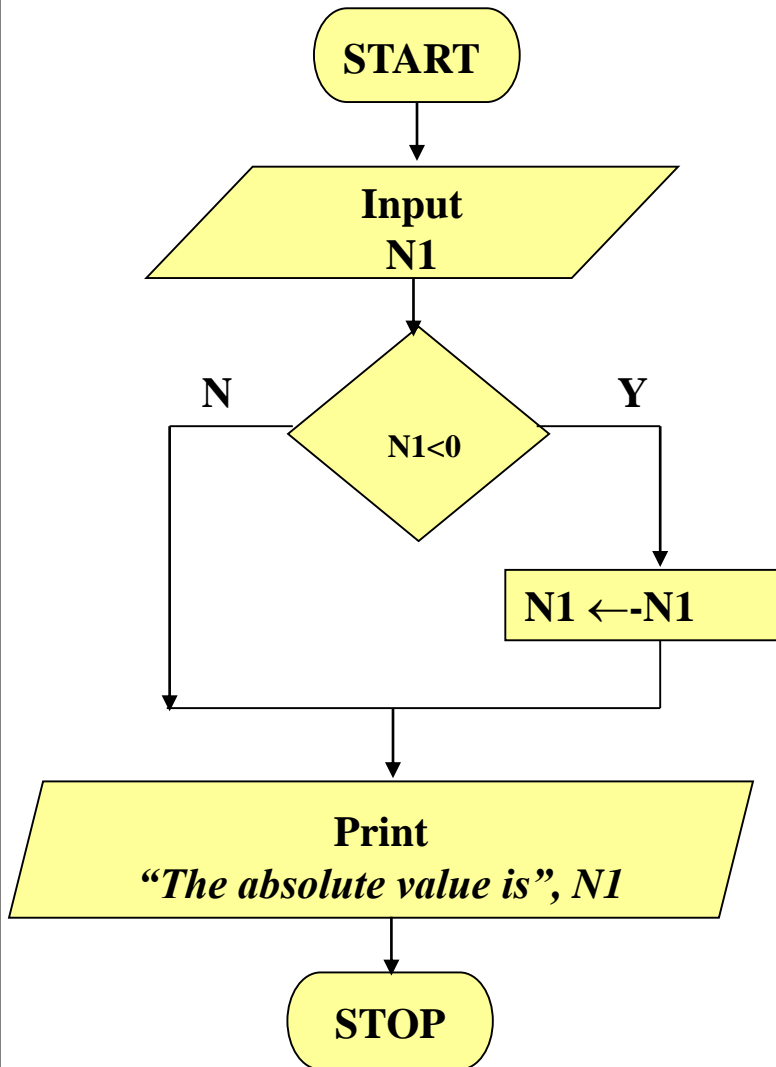
Standard algebraic equality operator or relational operator	FORTRAN equality or relational operator	Example of FORTRAN condition	Meaning of FORTRAN condition
<i>Relational operators</i>			
$>$	$>$	$x > y$	x is greater than y
$<$	$<$	$x < y$	x is less than y
\geq	$>=$	$x >= y$	x is greater than or equal to y
\leq	$<=$	$x <= y$	x is less than or equal to y
<i>Equality operators</i>			
$=$	$==$	$x == y$	x is equal to y
\neq	$/=$	$x /= y$	x is not equal to y

Beware of mistaking the assignment = for the equality ==



Example 1

The following FORTRAN program finds the absolute value of an integer.



```
Integer :: N1
```

```
Write (*,*) "Enter an integer: "
```

```
Read (*,*) N1
```

```
if (N1 < 0) then
```

```
    N1 = -N1
```

```
End if
```

```
Write (*,*) "The absolute value is ", N1
```

```
end
```

Logical Operators

- Logical operators are used to combine more than one condition forming a complex condition. FORTRAN logical operators are
 - **.OR. (Logical OR)**
 - only one of the conditions must be true for the compound condition to be true
 - **Syntax** (Condition_1 **.or.** Condition_2)
 - Example **if (x == 1 .or. x == y)**
 - **.and. (Logical AND)**
 - All of the conditions must be true for the compound condition to be true
 - **Syntax** (Condition_1 **.and.** Condition_2)
 - **Example** **if (2 < x .and. x < 7)**

If-else Statement

- Syntax

```
if (expression) then
```

```
    body1
```

```
else
```

```
    body2
```

```
end if
```

- Semantics

if **expression** is true then
execute **body1** otherwise execute **body2**

- Example

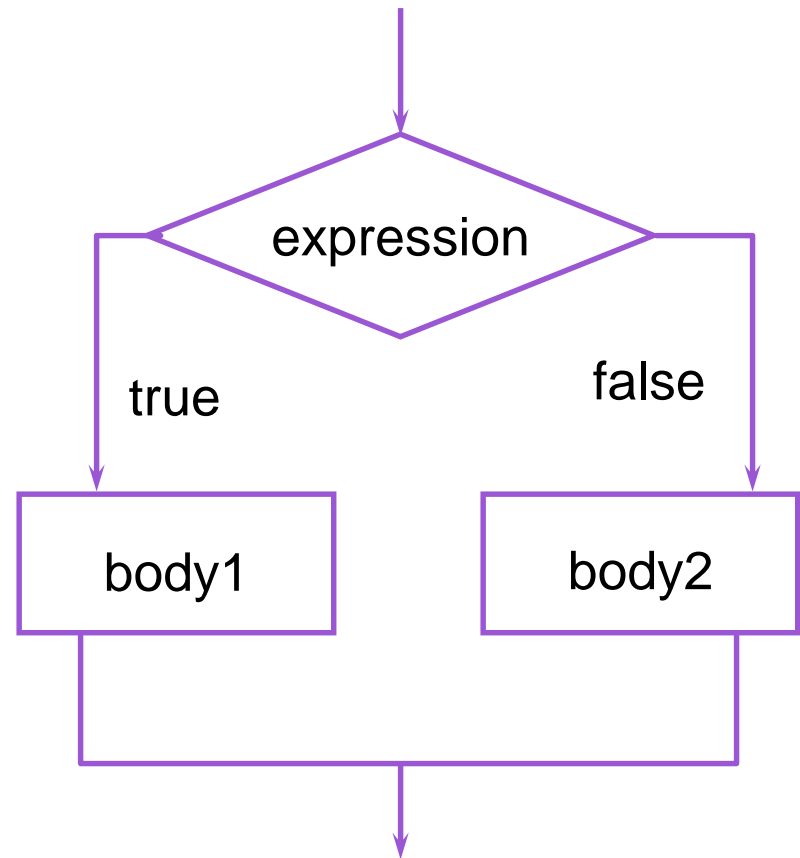
```
if (v == 0) then
```

```
    write (*,*) "v is 0"
```

```
else
```

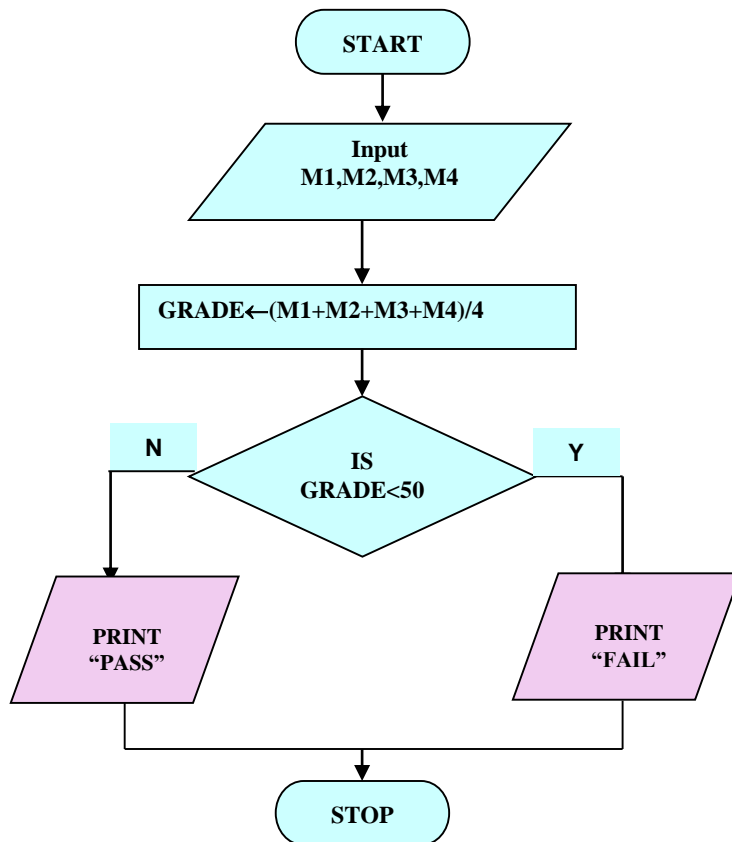
```
    write (*,*) "v is not 0"
```

```
End if
```



Example 2

Write a program to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks



Integer :: M1, M2, M3, M4, Grade

Read (*,*) M1, M2, M3, M4

Grade = (M1+M2+M3+M4)/4

If (Grade < 50) **then**

write (*,*) "fail"

else

write (*,*) "pass"

End if

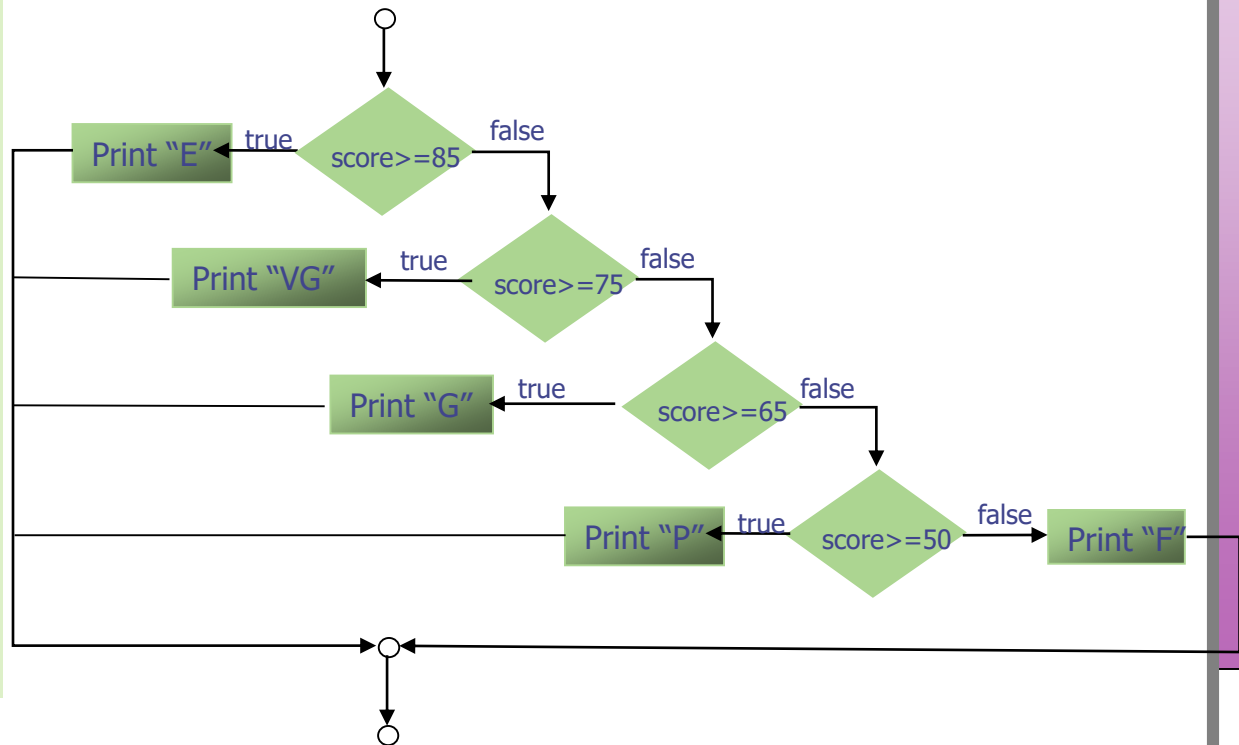
end

Multiple Selections: Nested if

Nesting: one control statement in another

```
Integer :: g1
Write (*,*) "your garde"
Read (*,*) g1
if (g1 >=85) then
  write (*,*) "excellent"
else if (g1 >=75) then
  write (*,*) "very good"
else if (g1 >=65) then
  write (*,*) "good"
else if (g1 >=50) then
  write (*,*) "pass"
else
  write (*,*) "fail"
End if
end
```

Write a program that calculate the student grade according to his score



Switch Statement

- **Syntax**

Select case (expression)

case (constant1)

statements

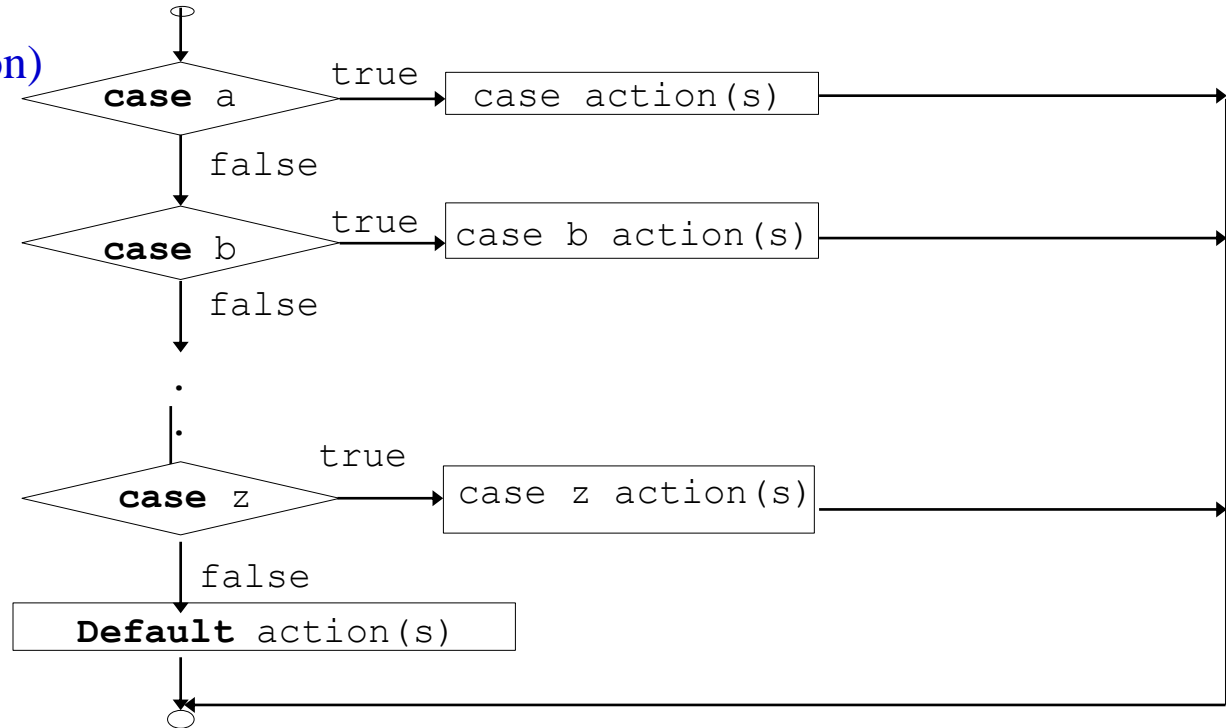
case (constant2)

statements

Case default

statements

End select



- **Semantics**

- The value of the **expression** is matched against a **case** value ,the statements execute
- If the value of the **expression** does not match any of the **case** values, the statements following the **default** label execute. If there is no **default** , the entire **switch** statement is skipped.

Select case Example 1

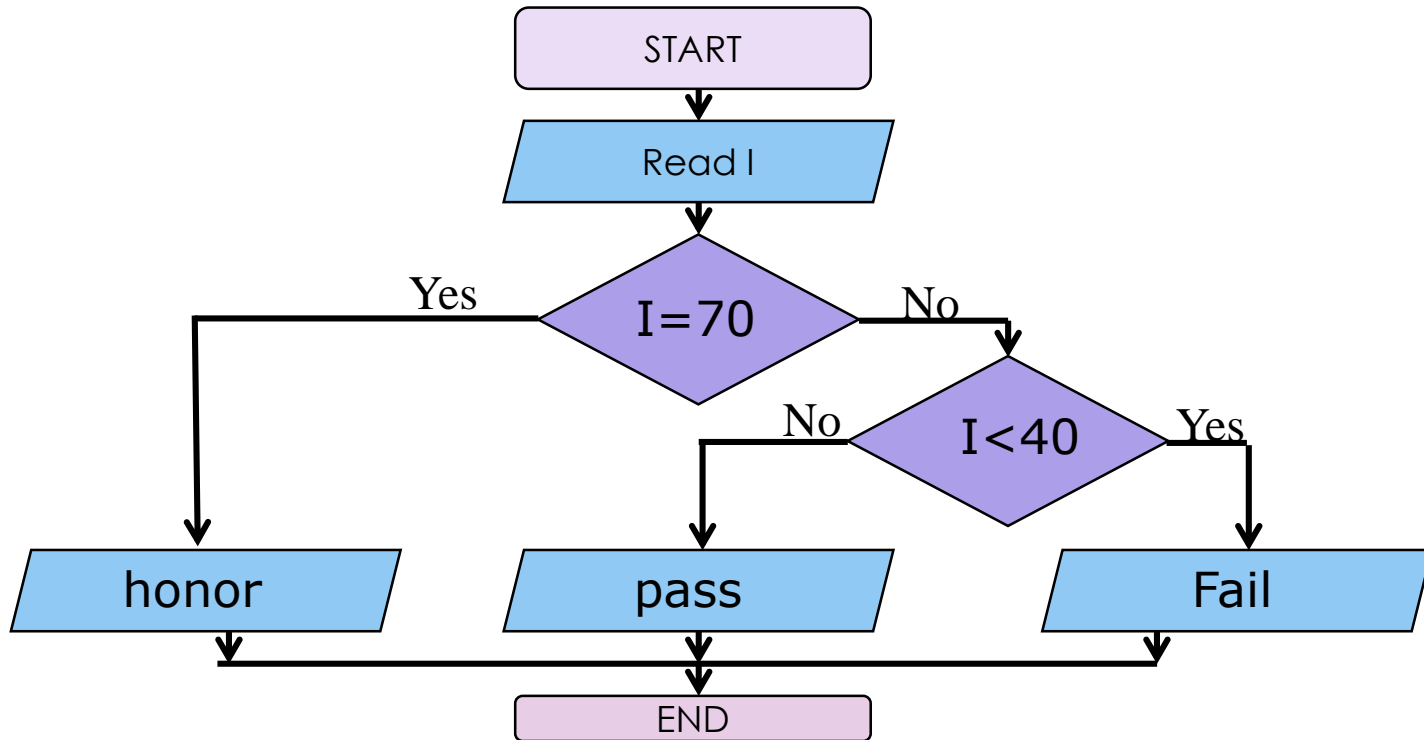
```
SELECT CASE (I)
CASE(1)
  Write(*,*) "I=1"
CASE(2:6,7,8,9)
  Write(*,*) "I >=2 and I<=9"
CASE(10:)
  Write(*,*) "I >=10"
CASE DEFAULT
  Write(*,*) "I<1 "
END SELECT
```

The result

- If I = 1, output is I=1
- If I = 2, output is I >=2 and I<=9
- If I = 5, output is I >=2 and I<=9
- If I = 10, output is I >=10
- If I = 15, output is I >=10
- If I = 0, output is I < 1

Example 2

- We want to create a flowchart that prints out the word “Honour” if the number input is 70, if the number is less than 40 print out the word “Fail”, otherwise print out the word “Pass”.



Select case Example 2

```
Integer :: A
Read (*,*) A
SELECT CASE (A)
CASE(70)
    Write(*,*) "Honor"
CASE(:39)
    Write(*,*) "Fail"
CASE DEFAULT
    Write(*,*) "pass "
END SELECT
```

The result

- If A = 70, output is Honor
- If A = 40, output is pass
- If A = 30, output is Fail
- If A = 50, output is pass
- If A = 75, output is pass

Unconditional GO TO

- This is the only GOTO in FORTRAN 77
 - Syntax: **GO TO label**
 - Unconditional transfer to labeled statement

```
10  -code-  
    GO TO 30  
    -code that is bypassed-  
30  -code that is target of GOTO-  
    -more code-  
    GO TO 10
```

- Flowchart:



- Problem: leads to confusing “spaghetti code”