Selection Control Structures

Lecture 05 – ICT1132

Piyumi Wijerathna
Department of ICT
Faculty of Technology

Overview

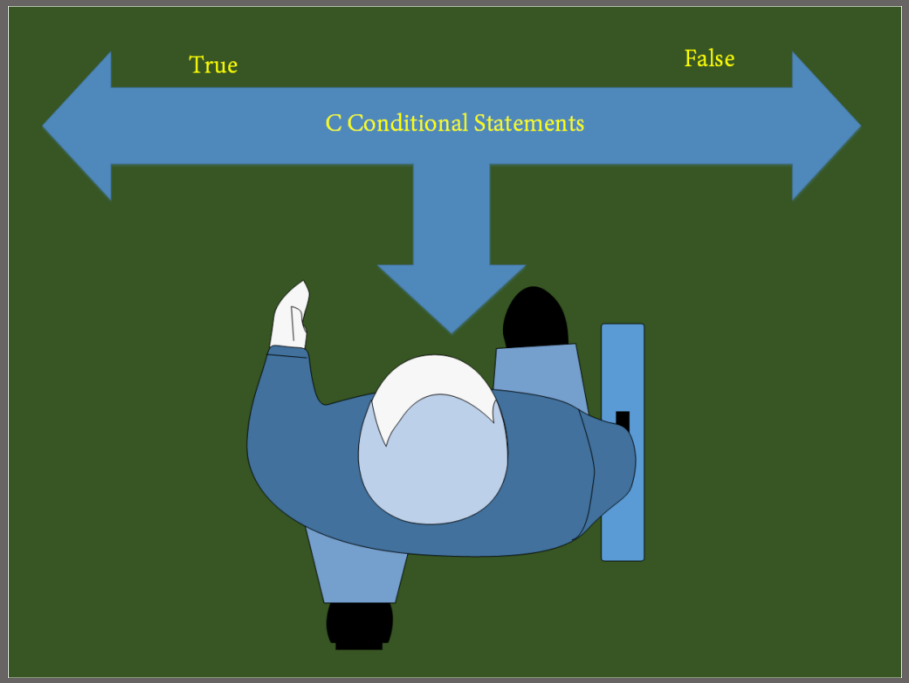
Selection Control Structures

- ✓ if statements
- ✓ if-else statements
- ✓ Nested if statements
- ✓ switch statement

Selection Control Structure

- With the selection control structure, the computer decides which statement to execute next depending on the value of a logical expression/condition.
- Hence can use to control the flow/order of execution of statements.
- Selection control structures in C are:
  - if statements
  - switch statements

if - Statements



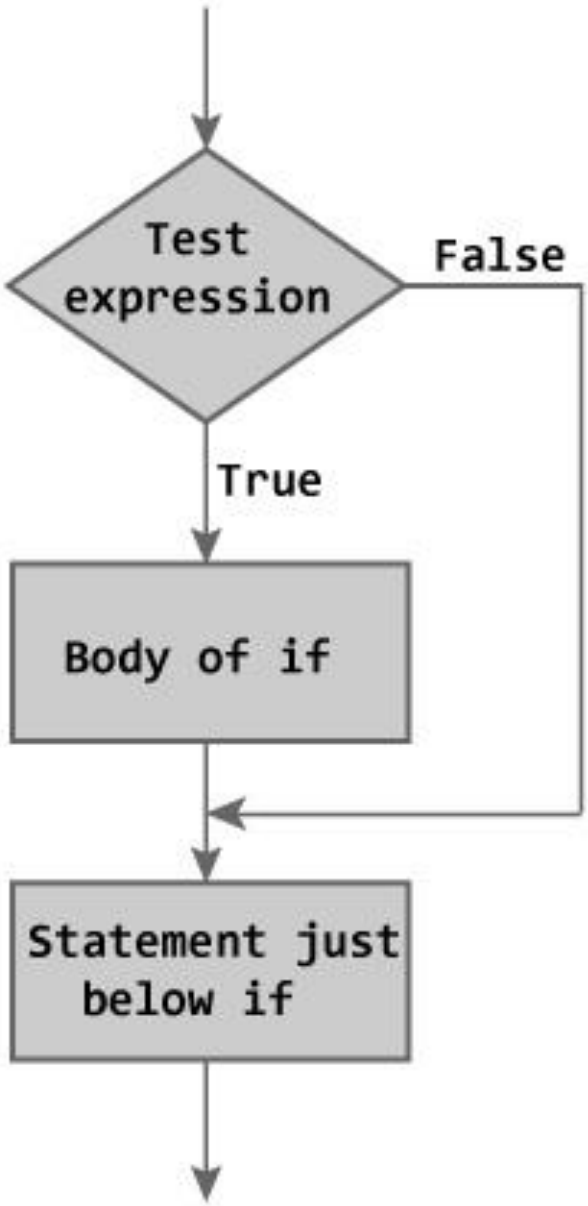
- Syntax of ***if*** statement is:

```
if (logical expression/condition){
        statements }
```

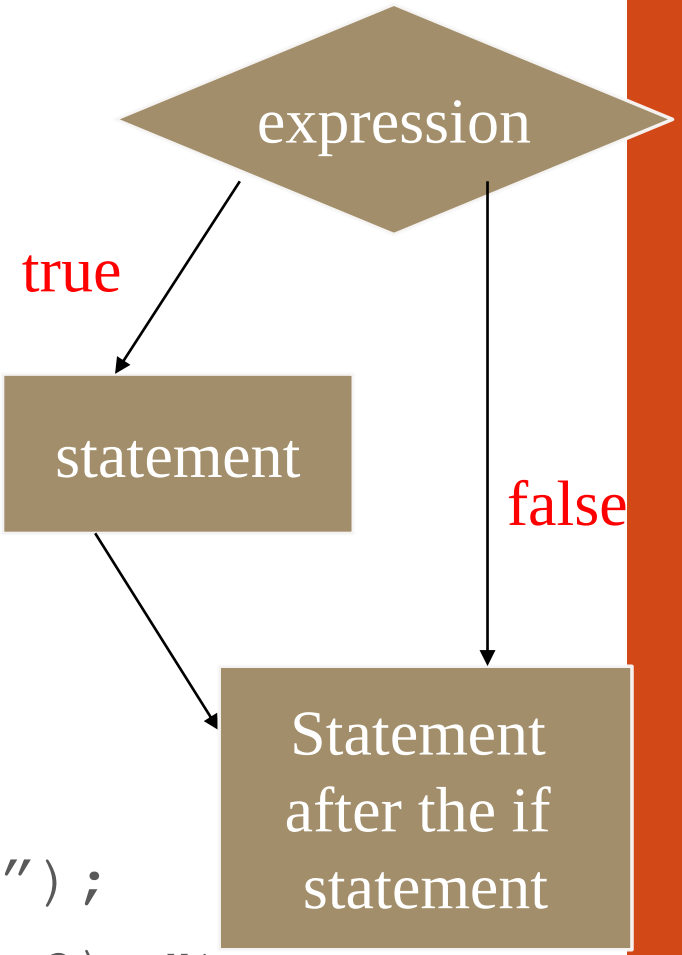
Returns true or false

- Each statement may be a single statement or a compound statement enclosed in curly braces (a block).
- The statement/block is
  - executed if the logical expression is ***true***
  - not executed if the logical expression is ***false***

Flowchart of if statement



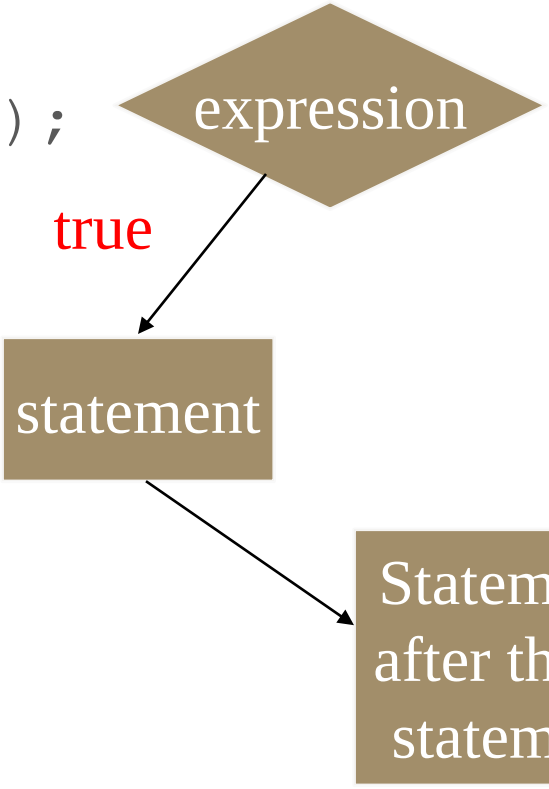
Example of “if”




```
if (3 < x)
{
  printf(“Inside IF block\n”);
  printf(“X is greater than 3\n”);
}
printf(“Outside IF block\n”);
```

when x is 5

```
if (3 < x)
{
  printf("Inside IF block\n");
  printf("X is greater than 3\n");
}
printf("Outside IF block\n");
```

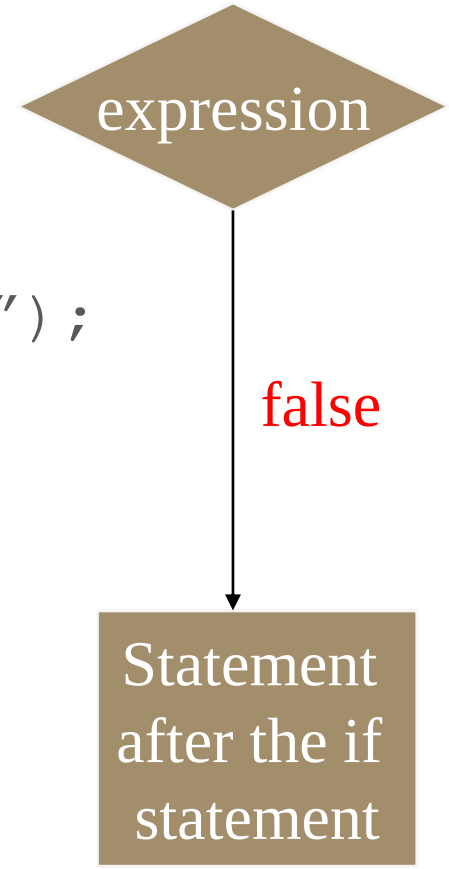


Output:

```
Inside IF block
X is greater than 3
Outside IF block
```

when x is 2

```
if (3 < x)
{
  printf("Inside IF block\n");
  printf("X is greater than 3\n");
}
printf("Outside IF block\n");
```



Output:

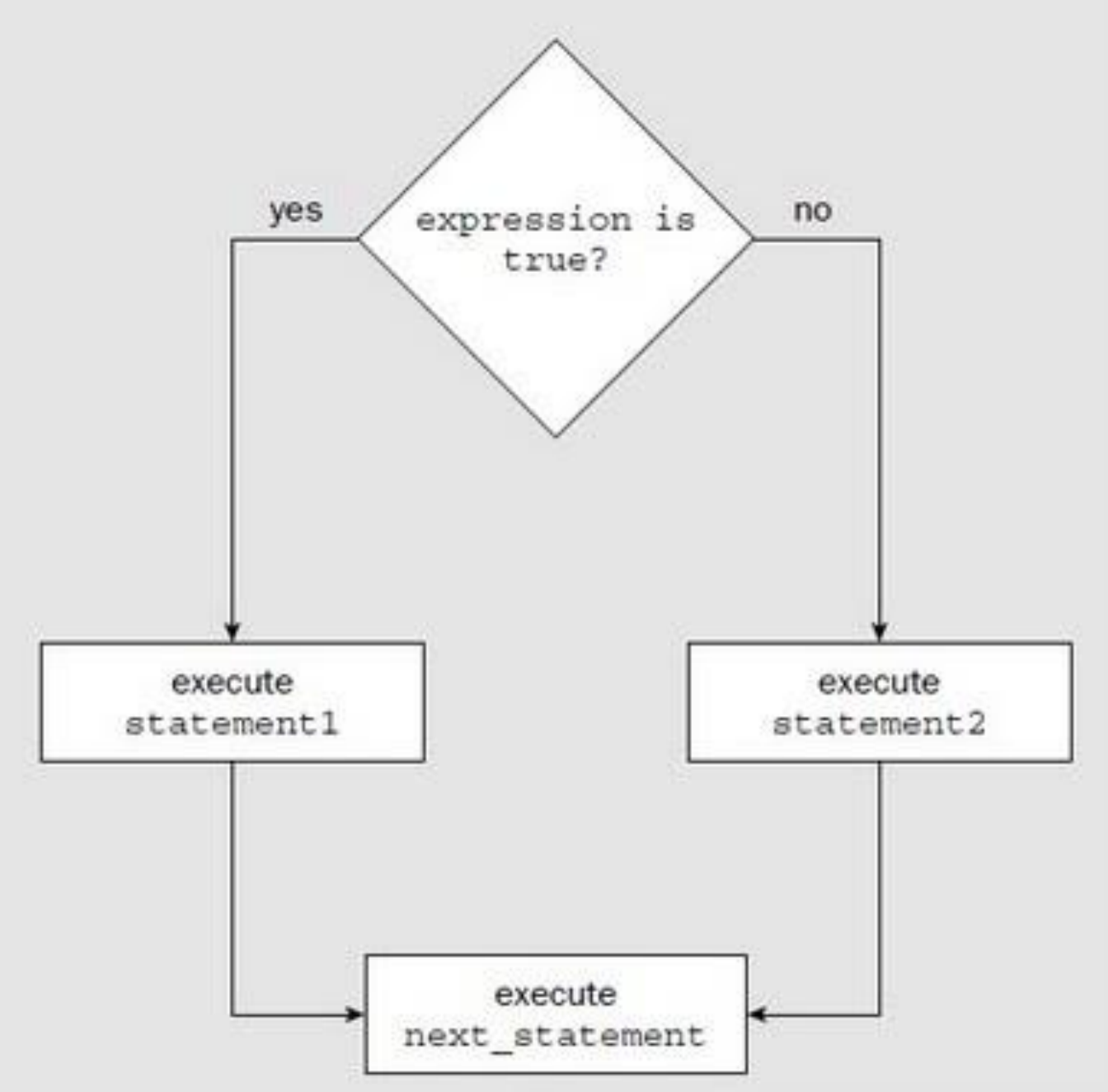
```
Outside IF block
```

if-else Statement

- Allows the program to choose between two alternatives based on a condition.
- Only one of the alternatives will be executed.
- The format of the if-else statement is:

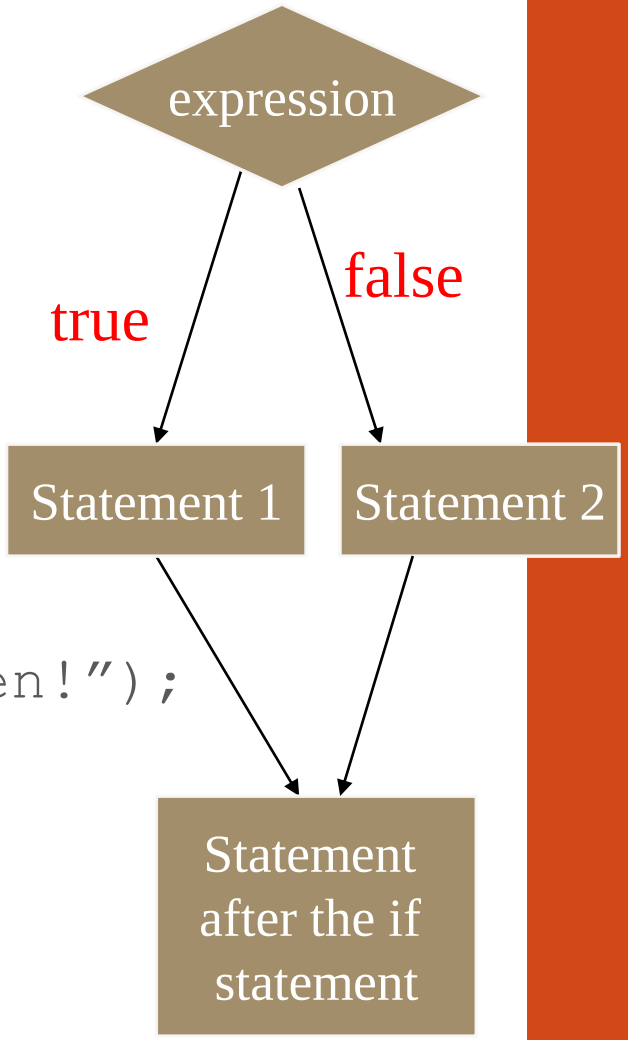
```
if (logical expression/condition){
    statement 1 }
else {
    statement 2 }
```

- Statement 1 is executed if the expression is ***true***
- Statement 2 is executed if the expression is ***false***


yes
expression is true?
no
execute statement1
execute statement2
execute next_statement

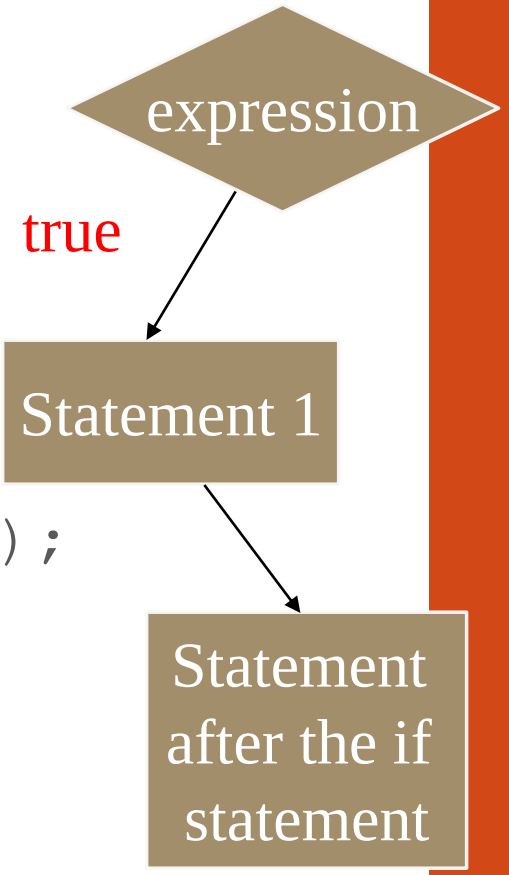
Example of if - else statement

```
if (Num == 7)
{
      printf("Congratulations,");
      printf(" you are lucky Seven!");
}
else
      printf("Sorry, try again");

printf("\nEnd of game");
```



When Num is 7

```
if (Num == 7)
{
  printf("Congratulations,");
  printf(" you are lucky Seven!");
}
else
  printf("Sorry, try again");

printf("\nEnd of game");
```

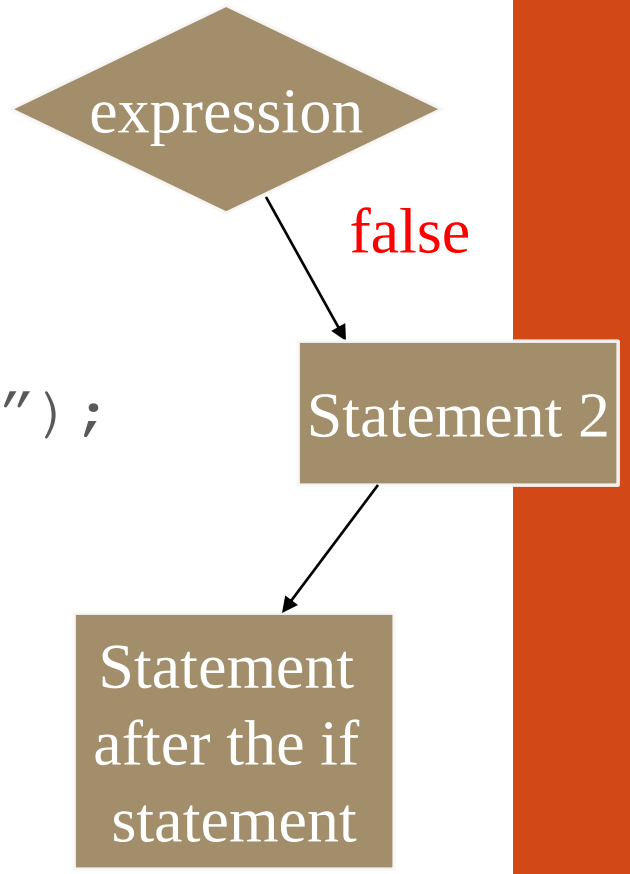


Output:

```
Congratulations, you are lucky Seven!
End of game
```

When Num is 5

```
if (Num == 7)
{
  printf("Congratulations");
  printf(" you are lucky Seven");
}
else
  printf("Sorry, try again");

printf("\nEnd of game");
```



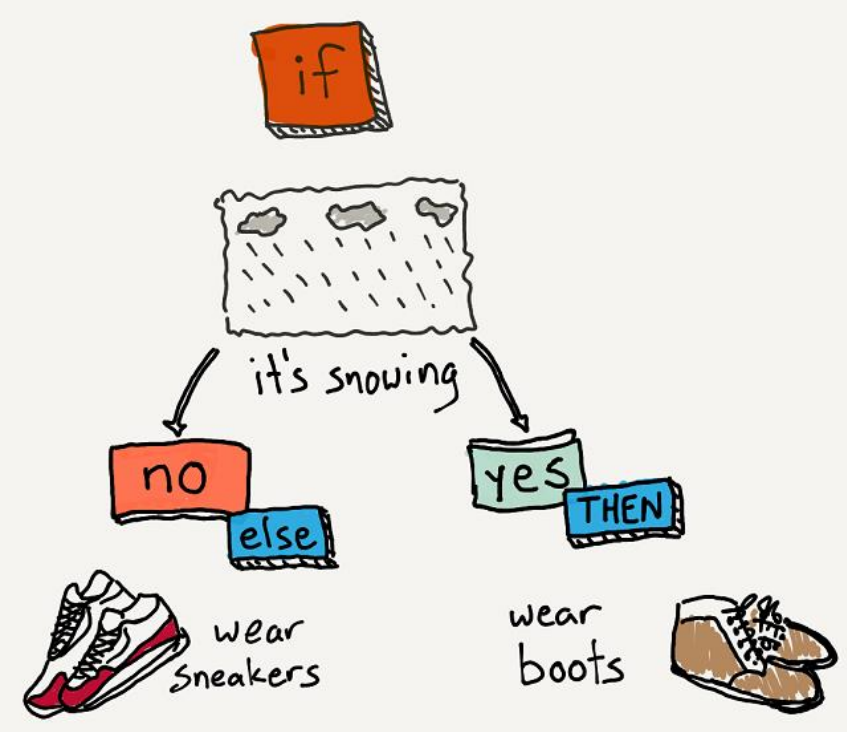
Output:

```
Sorry, try again
End of game
```

Warnings About Syntax

- The *else* must follow immediately after the *if* clause.
- If it is necessary to execute block of statements when the condition is true or false, need to create a block of the statements/compound statements enclosed in curly braces.

Nested if Statements



Nesting of if statements

- A nested if is if-else statements within another if or else.
- The nested if statement is called a multiple selection statement because it selects among many different actions.
- In a nested if-else, the entire set of statements with its *if* and *else* clause is considered to be one statement.

Syntax of nested if statements

```
if (Exp1)
{
        // statements to be executed if Exp1 is true
        if(Exp2){
            // statements to be executed if Exp1 is true and Exp2 is true
        }
        else{
            // statements to be executed if Exp1 is true and Exp2 is false
        }
}
else {
        // statements to be executed if Exp1 is false
        if(Exp3){
            // statements to be executed if Exp1 is false and Exp3 is true
        }
        else{
            // statements to be executed if Exp1 is false and Exp3 is false
        }
}
```

if-else-if Ladder

- A common programming construct that is based upon a sequence of nested ifs.
- valuated from top to down.
- As soon as a condition from the ladder evaluates to true, the statements associated with that if are executed, and the remaining part of the ladder is bypassed.
- The last most else is executed only when no condition in the whole ladder returns true.

Syntax of if-else-if statements

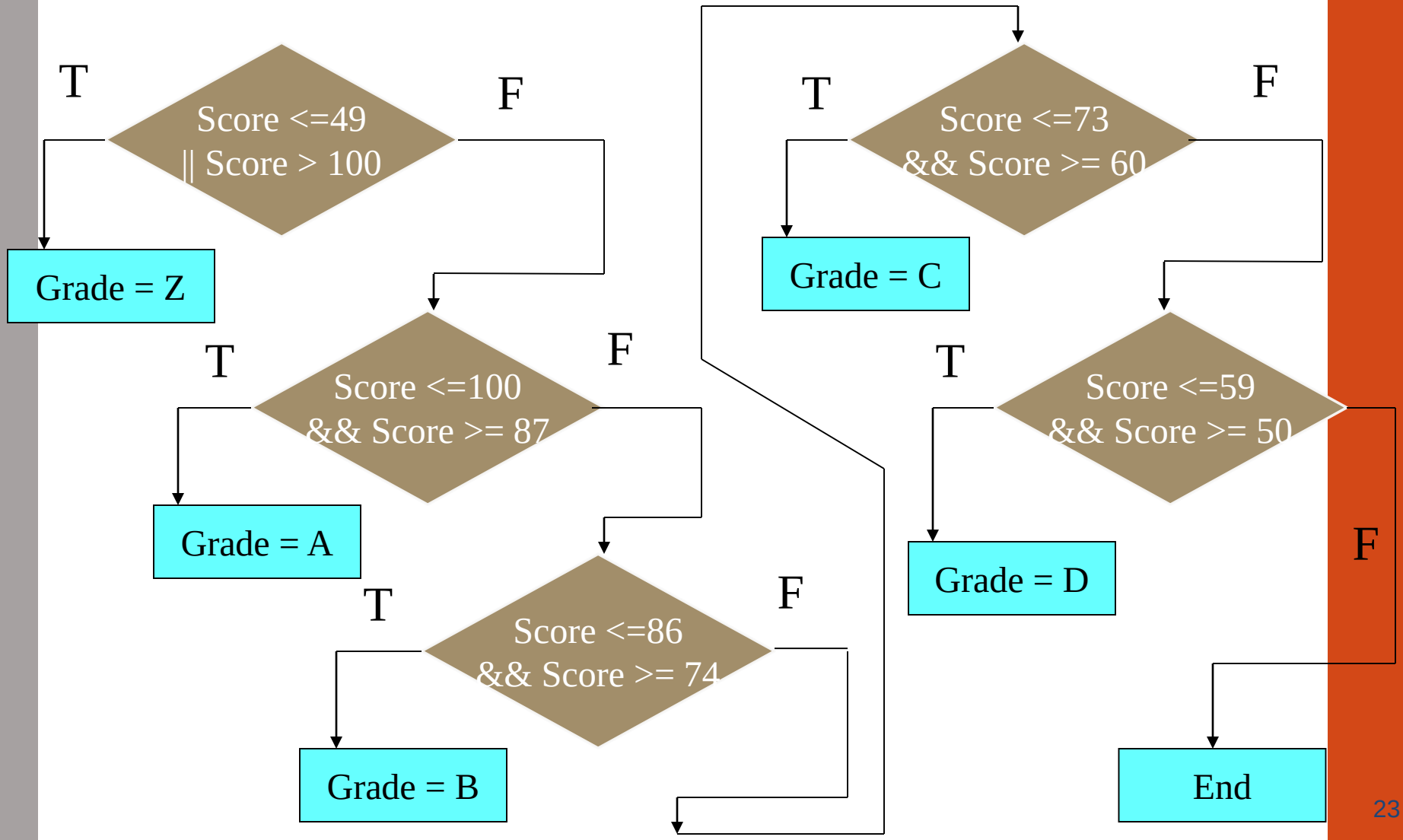
```
if (Exp1)
{
  // statements to be executed if Exp1 is true
}
else if(Exp2)
{
  // statements to be executed if Exp1 is false and Exp2 is
true
}
else if (Exp3)
{
  // statements to be executed if Exp1 and Exp2 is false and
Exp3 is true
}
…
else
{
  // statements to be executed if all expressions are false }
```

Multiple Decisions - Illustration

- Example: Determine the grade for a particular score.

100-87	A
86-74	B
73-60	C
59-50	D
Scores bellow 50 or above 100	Z

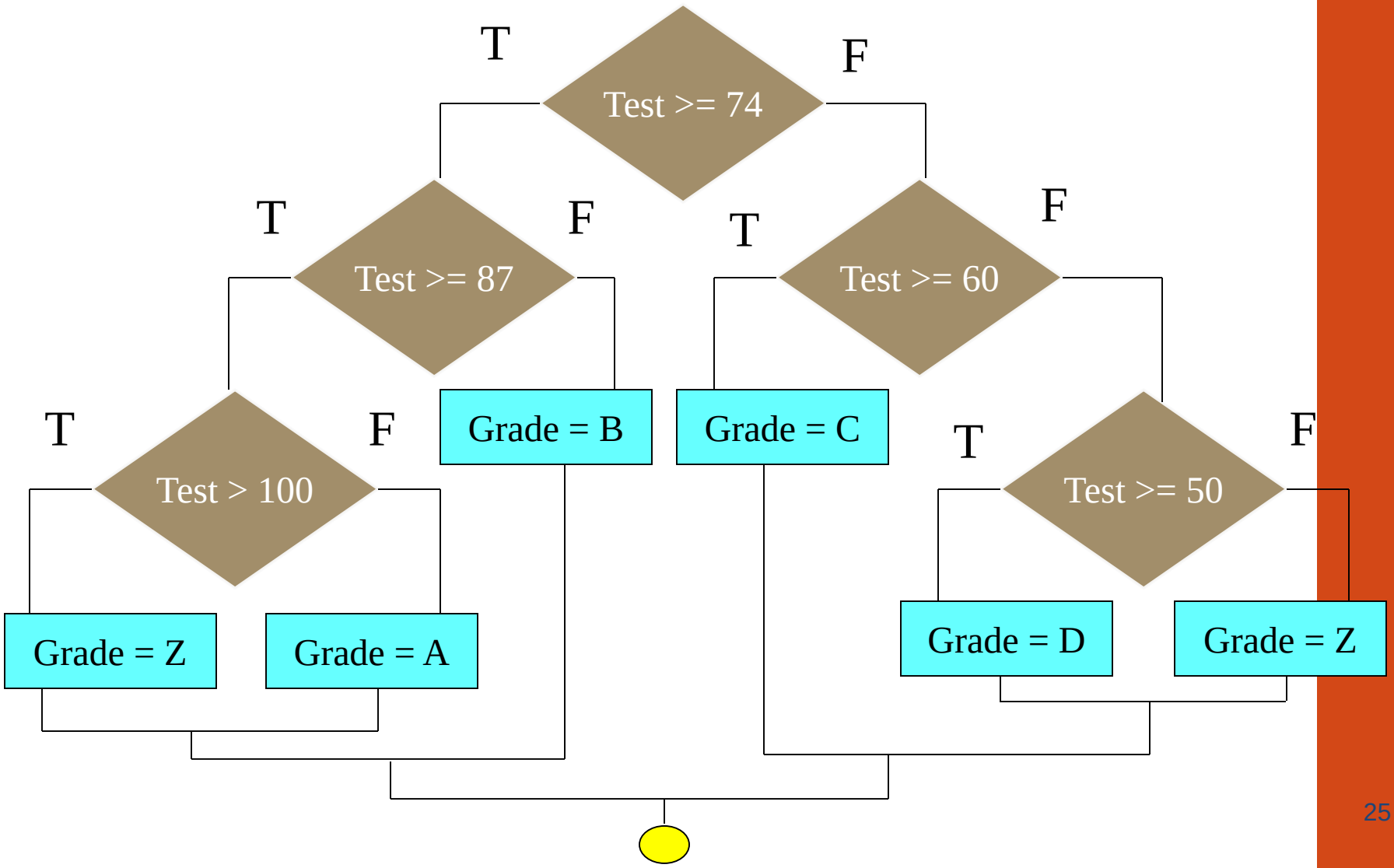
1. Using if statements – no nesting



Multiple Selection Without Nesting

```
if ((Test > 100) || (Test <= 49))
            Grade = Z;
if ((Test <= 100) && (Test >= 87))
            Grade = A;
if ((Test <= 86) && (Test >= 74))
            Grade = B;
if ((Test <= 73) && (Test >= 60))
            Grade = C;
if ((Test <= 59) && (Test >= 50))
            Grade = D;
```

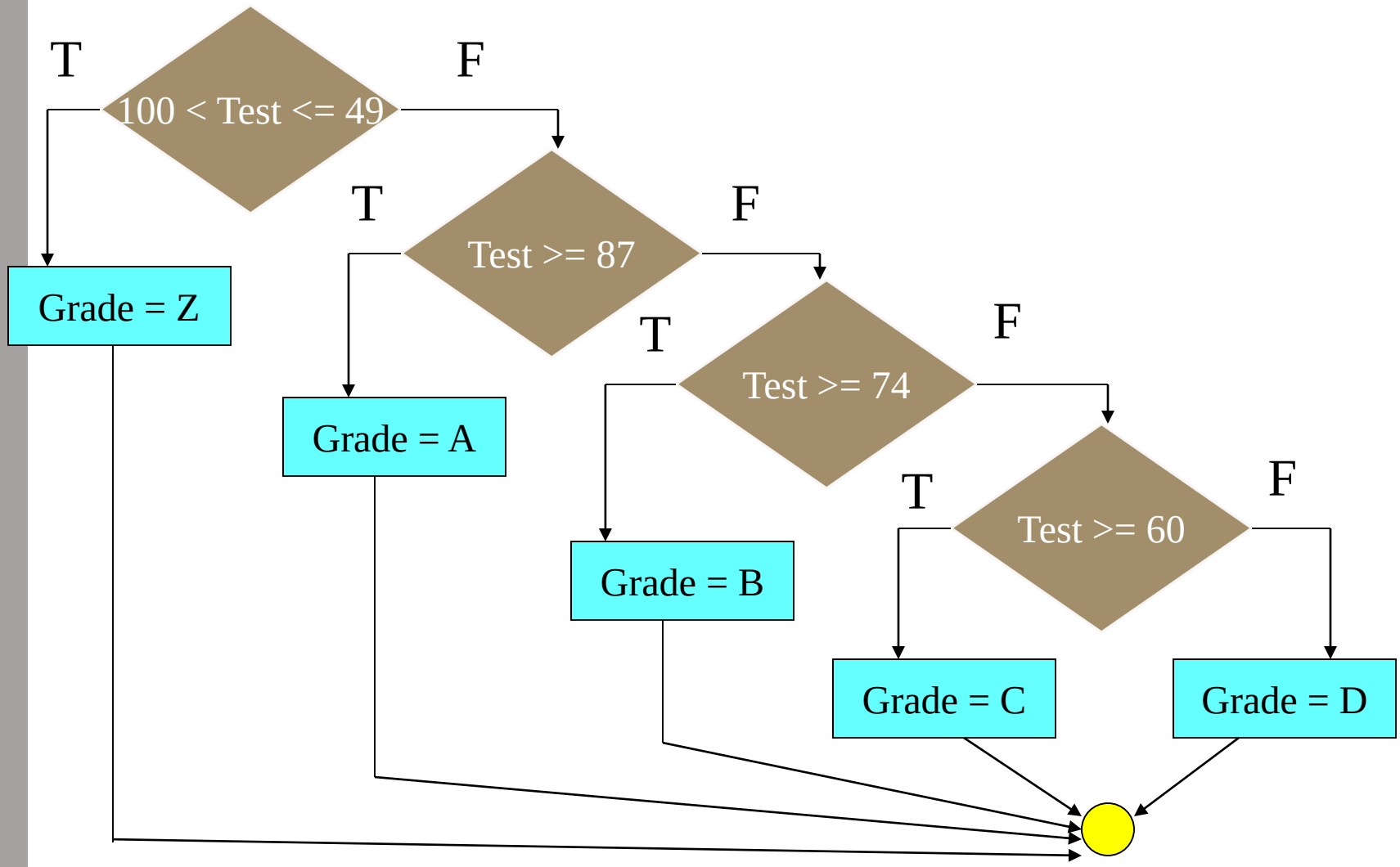
2. Using nested if - else construct for Binary Decisions



Using nested if - else construct for Binary Decisions

```
if (Test >= 74)
     if (Test >= 87)
          if (Test >= 100)
               Grade = Z;
          else
               Grade = A;
     else
          Grade = B;
else
     if(Test >= 60)
          Grade = C;
     else
          if(Test >= 50)
               Grade = D;
          else
               Grade = Z;
```

3. Using "if-else-if" Ladder construct



Using "if-else-if" Ladder construct

```
if ((Test > 100) || (Test <= 49))
    Grade = Z;
else if (Test >= 87)
    Grade = A;
else if (Test >= 74)
    Grade = B;
else if (Test >= 60)
    Grade = C;
else
    Grade = D;
```

Important Points:

- else and else if are optional statements, a program having only "if" statement would run fine.
- else and else if cannot be used without the "if".
- There can be any number of else if statement in a if- else if block.
- If none of the conditions are met then the statements in else block gets executed.
- Just like relational operators, we can also use logical operators such as AND (&&), OR(| |) and NOT(!).

Switch Statements

Multiple-Way Selection with switch Statement

- The switch statement is an alternative to the if-else-if statement provided the expressions can be written as:

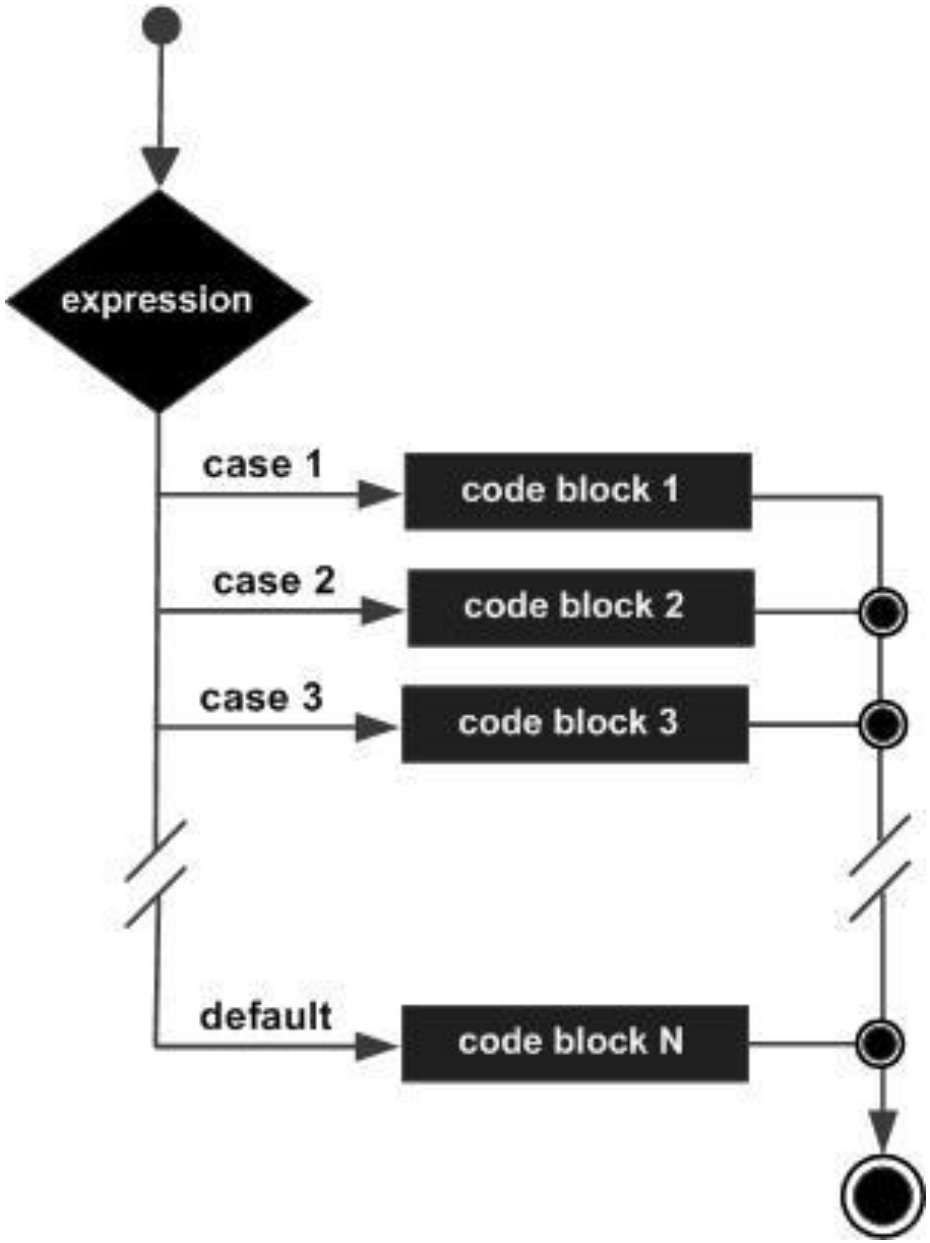
  (variable == value)

- The switch statement allows for execution of multiple statements for a given condition, using the break statement to terminate execution for the condition.

Syntax for the switch Statement

```
switch (variable/expression)
{
    case c1: any_number_of_statements1;
             break;
    case c2: any_number_of_statements2;
             break;
    …
    default: any_number_of_statements;
}
```

Flow Diagram of Switch statement



Example

- Write a program to enter a letter and print the flavor of the ice cream.

Letter	Flavor
s	strawberry
v	vanilla
c	chocolate

```
char letter;
printf(“ Enter the first letter of the flavour : “);
scanf(“%c”,&letter);

switch (letter)
{
  case ‘s’ :
                printf(“Strawberry\n”);
                break;
  case ‘v’ :
                printf(“Vanilla\n ”);
                break;
  case ‘c’ :
                printf(“Chocolate \n”);
                break;
   default  :
                printf(“Invalid Letter ”);
}
```

The “break” statement

- Syntax : break;
- The break statement causes the switch statement to terminate and begin execution with the statements after the switch statement.
- If a break statement does not appear at the end of a group of statements for a *case*, processing continues sequentially even though the next statements may be specified for another *case*.

The “default” statement

- Syntax : default;
- The default case specifies the switch section to execute if the match expression does not match any other case label.
- If a default case is not present and the match expression does not match any other case label, program flow falls through the switch statement (no output).
- The default case can appear in any order in the switch statement.
- Regardless of its order in the source code, it is always evaluated last, after all case labels have been evaluated.

Rules of Using Switch Case

1. Case Label must be unique.
2. Case Labels must ends with Colon.
3. Case labels must have constants / constant expression.
4. Case label must be of integral type (Integer/Character).
5. Case label should not be 'floating point number'.
6. Switch case should have at most one default label.
7. Default label is Optional.
8. Default can be placed anywhere in the switch.
9. Break statement takes control out of the switch.
10. Two or more cases may share one break statement.
11. Nesting (switch within switch) is allowed.
12. Relational Operators are not allowed in Switch Statement.
13. const variable is allowed in switch Case Statement.

1. Case Labels must be unique.

```
int id = 3 ;
switch(id)
        {
        case 1:
                printf("C Programming Language");
                break;
        case 2:
                printf("C++ Programming Language");
                break;
        case 2:
                printf("Web Technology");
                break;
        default :
                printf("No student found");
                break;
         }
```

2. Case Labels must ends with a Colon.

```
case 1 :
        printf("C Programming Language");
        break;
```

3. Case labels must have constants / constant expression

```
case 1+1:
case 'A':
case 67:
```

```
case var :
case num1 :
case n1+n2 :
```

4. Case label must be of integral type (Integer/Character)

```
case 10:
case 20+20:
case 'A':
case 'a':
```

5. Case label should not be ‘floating point number’.

```
case 10.12:
case 7.5:
```

6. Switch case should have at most one default label.

```
switch(roll)
        {
        case 1:
                printf("C Programming Language");
                break;
        case 2:
                printf("C++ Programming Language");
                break;
        case 3:
                printf("Web Technology");
                break;
        default :
                printf("Default Version 1");
                break;
        default :
                printf("Default Version 2");
                break;
         }
```

7. Default label is Optional.

```
switch(roll)
        {
        case 1 :
                printf("C Programming Language");
                break;
        case 2 :
                printf("C++ Programming Language");
                break;
        case 3 :
                printf("Web Technology");
                break;
         }
```

8. Default can be placed anywhere in the switch.

```
switch(roll)
        {
        case 1 :
                printf("C Programming Language");
                break;
        default:
                printf("No Student Found");
                break;
        case 2 :
                printf("C++ Programming Language");
                break;
        case 3 :
                printf("Web Technology");
                break;
         }
```

10. Two or more cases may share one break statement.

```
switch(alpha)
        {
        case 'a':
        case 'A':
                printf("Alphabet A");
                break;
        case 'b':
        case 'B':
                printf("Alphabet B");
                break;
         }
```

11. Nesting (switch within switch) is allowed.

```
switch(alpha)
        {
        case 'a':
        case 'A':
                printf("Alphabet A");
                break;
        case 'b':
        case 'B':
                switch(alpha)
                {
                }
                break;
         }
```

12. Relational Operators are not allowed in Switch Statement.

```
switch(num)
        {
        case >15:
                printf("Number > 15");
                break;
        case =15:
                printf("Number = 15");
                break;
        case <15:
                printf("Number < 15");
                break;
         }
```

13. const variable is allowed in switch Case Statement.

```
int const var = 2;

switch(num)
        {
        case var:
                printf("Number = 2");
                break;
         }
```

Try This

```
switch (n)
{
   case 24:     printf("A");
   case 6:      printf("B");
                break;
   case 7:      printf("C");
   case 5:      printf("D");
   default:     printf("E");
   }
```

What is the output when n is 6?

What is the output when n is 24?

What is the output when n is 5?

What is the output when n is 7?

What is the output when n is 9?


Any
Question

THANK YOU… !