Chapter 7 Control Statements

Control statements are used for changing the normal flow of program execution. Two types of control statements are

1. Decision making / selection statements.
2. Iteration statements / looping statements

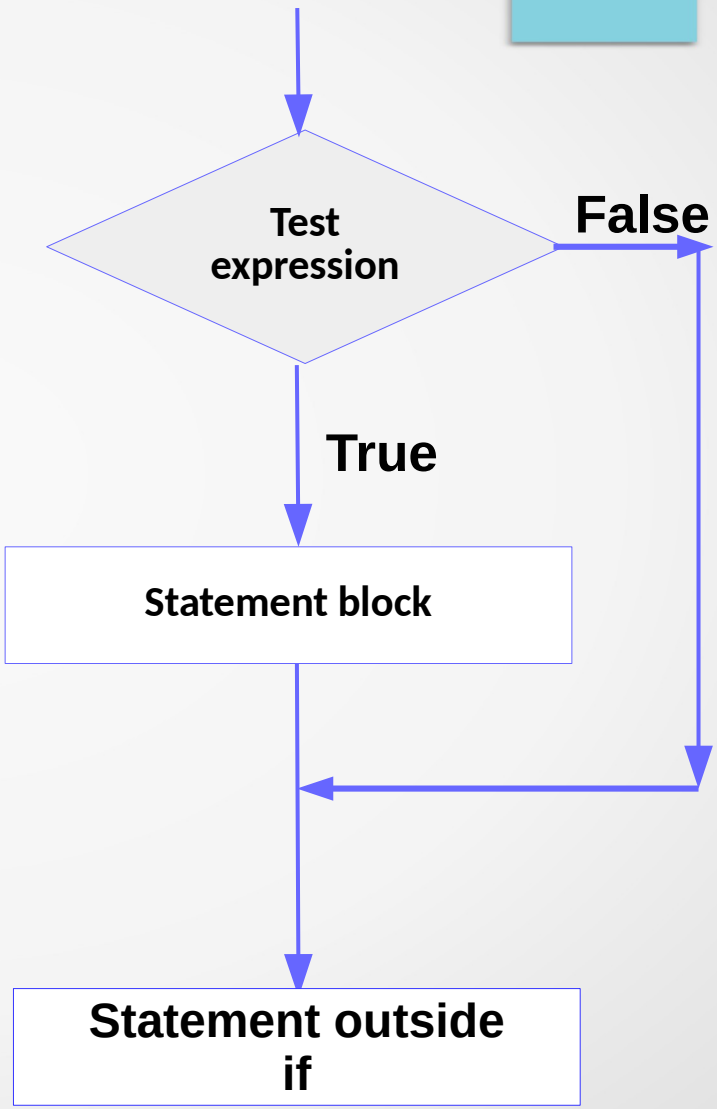
Selection/Decision statements

- If
- If else
- Nested if
- Else if ladder
- Switch
- Conditional operator(? :)

if statement

```
if (test expression)
    {
        statement block;
     }
```

If the test expression is true , the statement block is executed

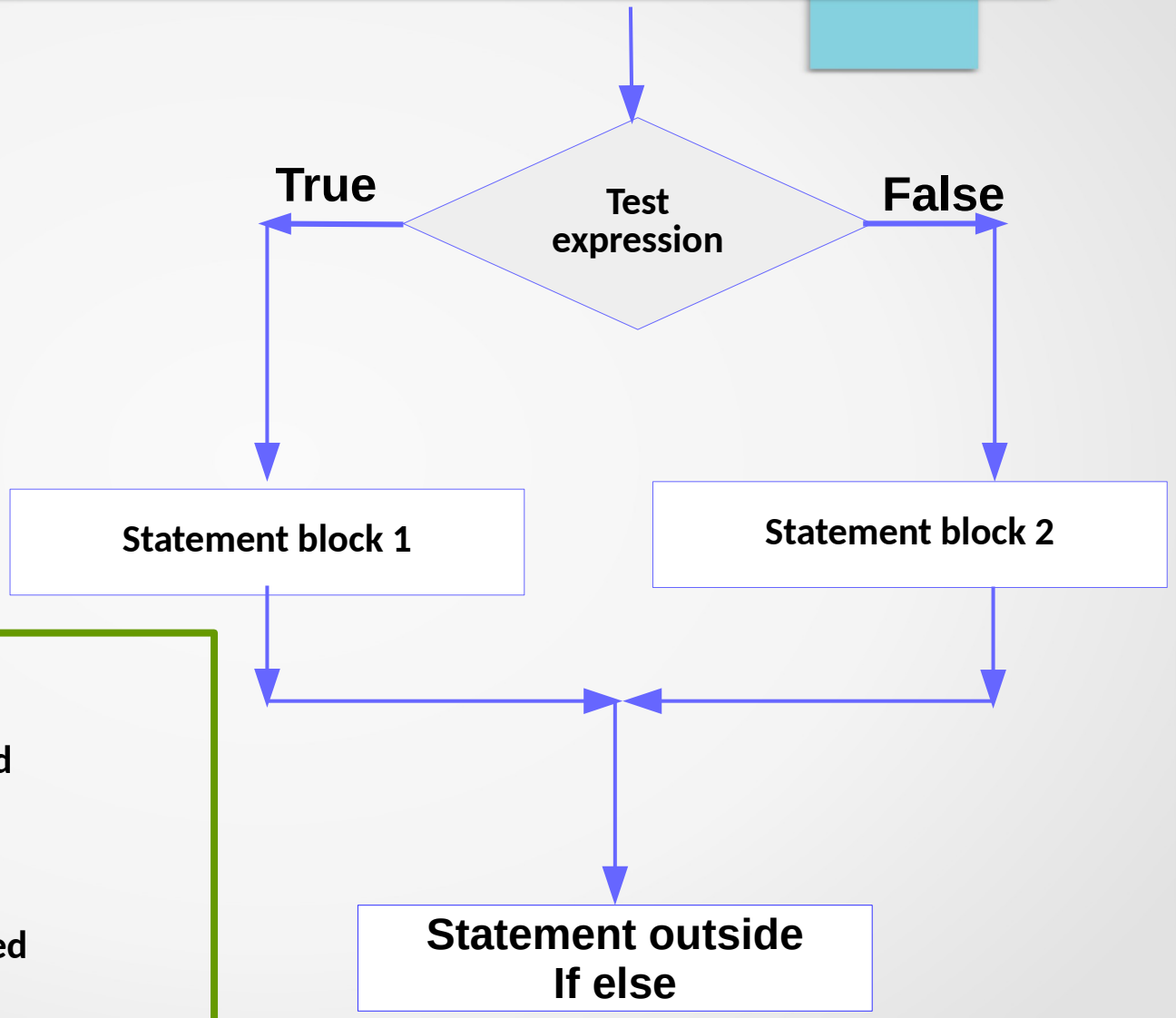


If-else statement

```
If (Test expression)
 {
        statements block 1;
 }

  Else

 {
        statements block 2;
 }
```



- If the Test expression is true,

  the statement block1 executed

- If the Test expression is false,

  the statements block 2 executed

Nested if

```
if(test expression 1)
  {
    if(test expression2)
       {
          Statement 1;
       else
            Statement 2;
         }
  else
     {
       Body of else;
      }
```

The else if ladder

```
if(test expression1)
    Statement block 1;
        else if (test expression 2)
            statement block 2;
                else if (test expression 3)
                    statement block 3;
                        .....................
                            else
                                statement block n;
```

Switch Statement

```
switch(expression)
{
 case constant_1 : statement block1;
                    break;
case constant_2 : statement block2;
                   break;
                  :
case constant_n-1 : statement block n-1;
                     break;
default                    : statement block n;
}
```

Comparison between switch and else if

switch statement	else if ladder
• Permits multiple branching.	• Permits multiple branching.
• Evaluates conditions with equality operator only.	• Evaluate any relational or logical expression.
• Case constant must be an integer or a character type value.	• Condition may include range of values and floating point constants.
• When no match is found, `default` statement is executed.	• When no expression evaluates to True, `else` block is executed.
• `break` statement is required for exit from the `switch` statement.	• Program control automatically goes out after the completion of a block.
• More efficient when the same variable or expression is compared against a set of values for equality.	• More flexible and versatile compared to `switch`.

The conditional operator(? :)

Syntax

Test expression ? True case code: false case code;

Iteration Statement

Loops(Iteration Statement) repeat a statement a certain number of times until a condition satisfied

There are three type of loops

- **While**
- **For**
- **do while**

Parts of a loop

Initialisation

This executed first and only once

This initialise the first value to loop control variable

Test Expression

- **Condition evaluated, if it true loop body executed. If it false loop terminated**

Update statement

This statement allows to update loop control variable

Body of loop

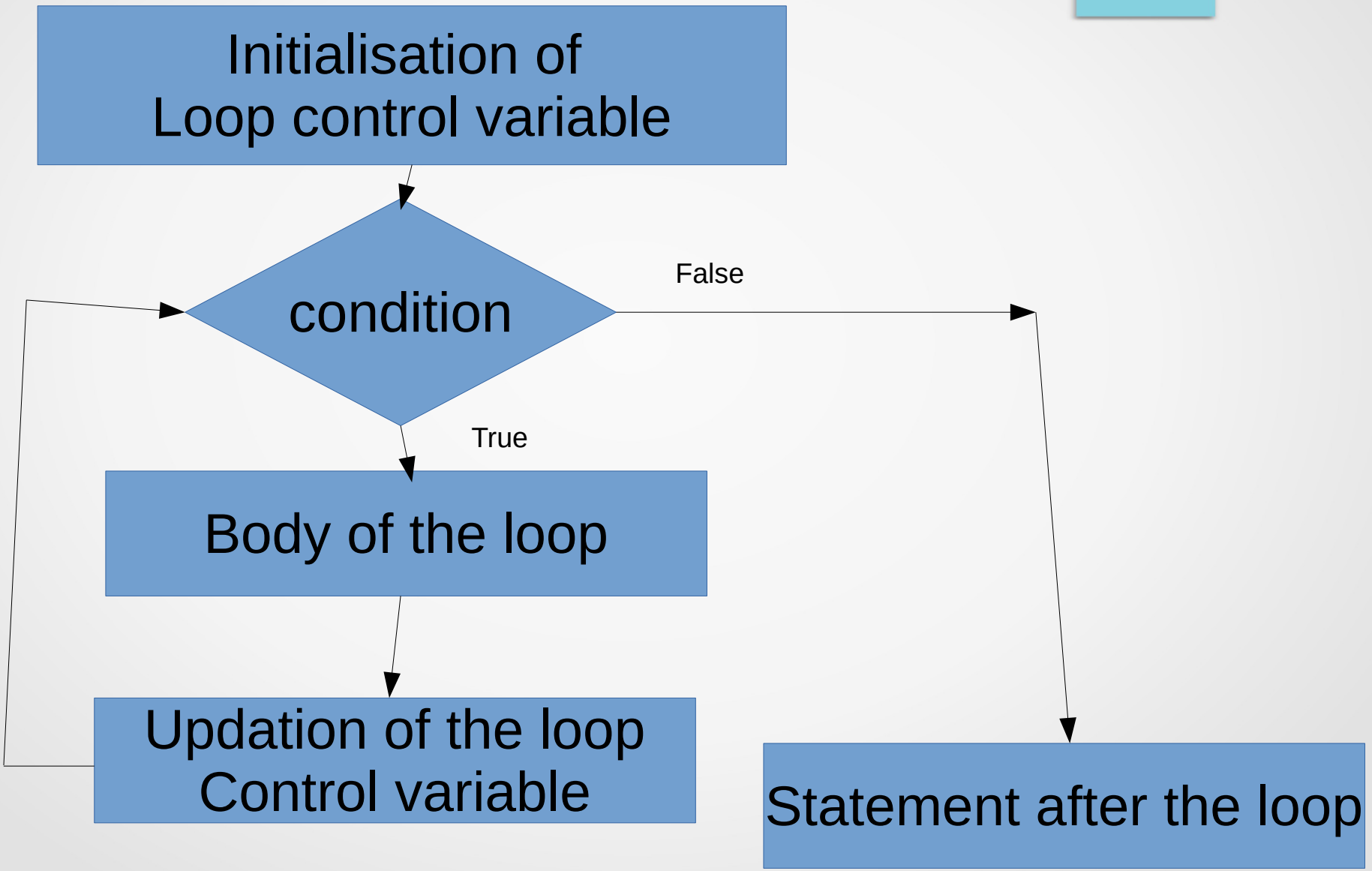
- **It include the statements to be executed repeatedly**

while statement

Syntax

```
Initialisation of loop control variable;
while(test expression)
   {
     body of the loop;
     updation of  loop control variable;
  }
```

While statement working



For statement

```
for(initialisation;test expression;update statement)
   {
      body of the loop;
   }
```

For statement working

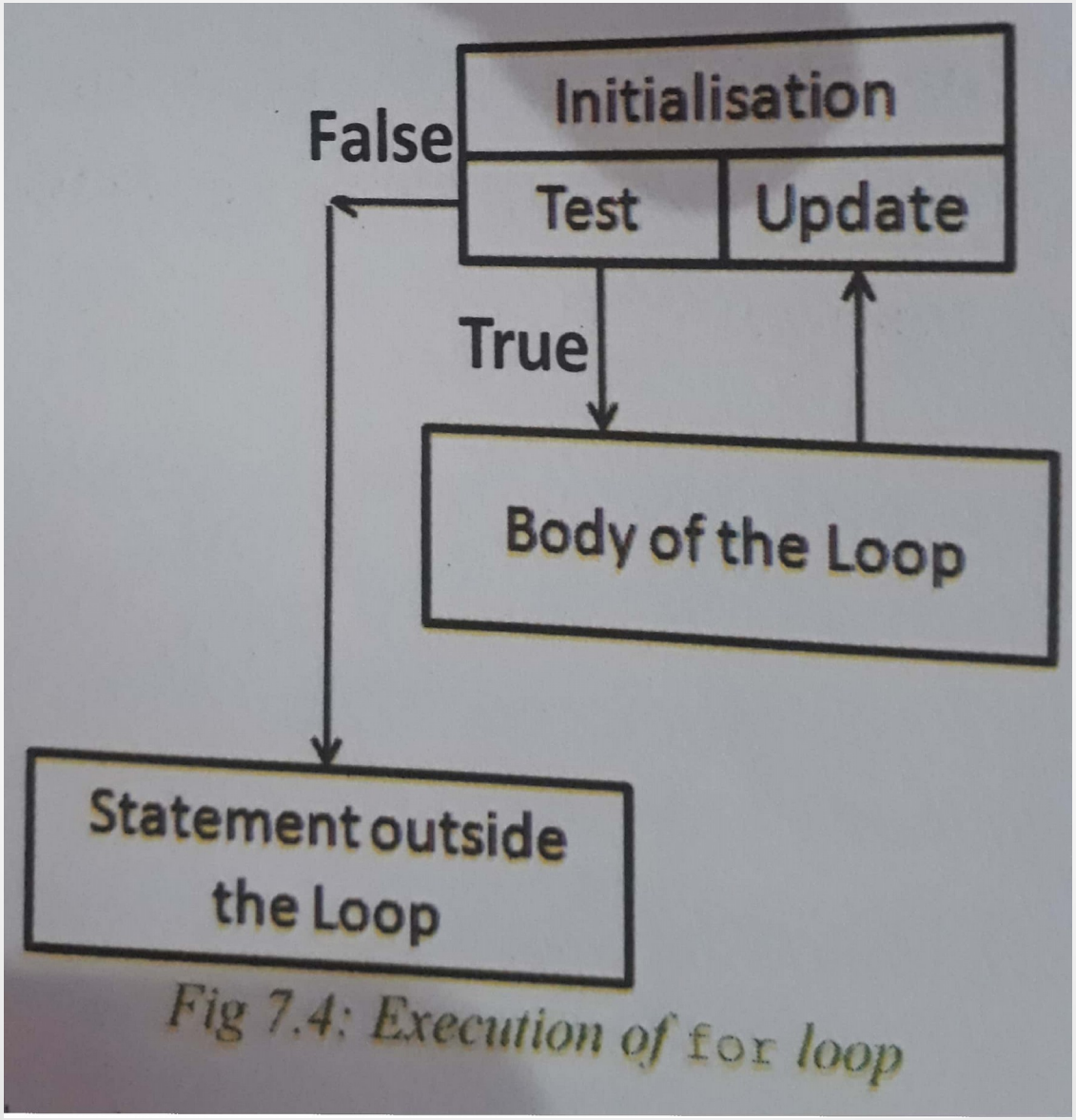


Fig 7.4: Execution of `for` *loop*

do.... while statement

```
Initialisation of loop control variable;
do
    {
        body of the loop;
        updation of loop control variable;
     } while(test expression);
```

do... while working

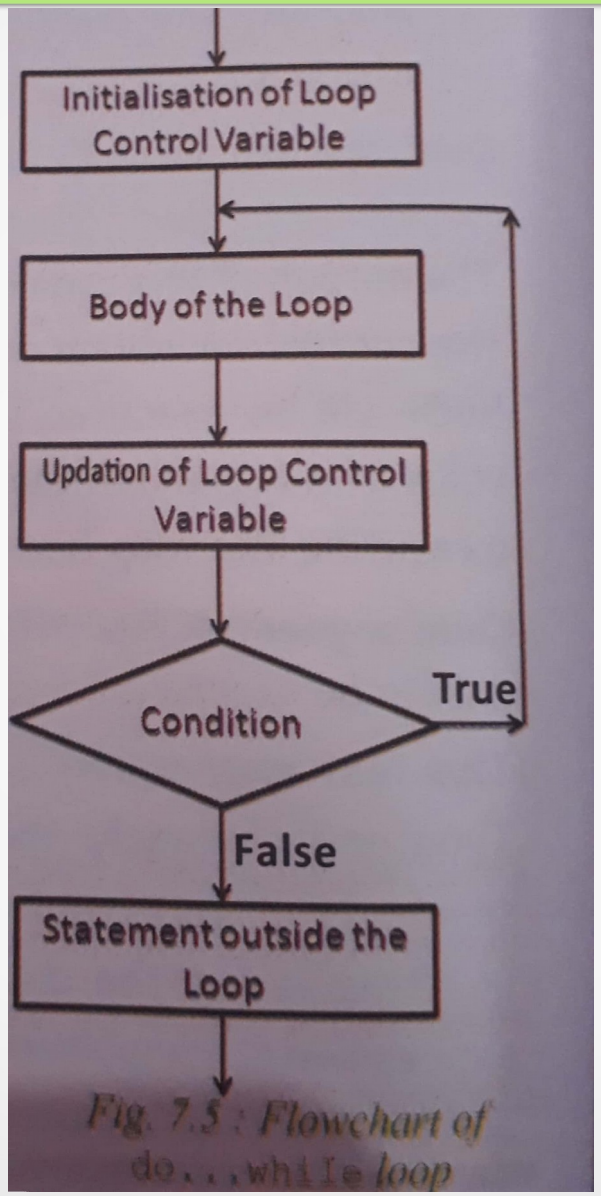


Fig. 7.5 : Flowchart of do...while loop

Types of loops

1. entry controlled loop
   (While and for loop)
2. exit controlled loop
   (do.. while loop)

Loop comparison

For loop	While loop	do... while loop
Entry controlled loop	Entry controlled loop	Exit controlled loop
Initialisation along with loop definition	Initialisation before with loop definition	Initialisation before with loop definition
No guarantee to execute the loop body at least once	No guarantee to execute the loop body at least once	Will execute the loop body at least once even though the condition is false

break continue comparison

break	continue
Used with switch and loops	Used only with loops.
Brings the program control outside the switch or loop by skipping the rest of the statements within the block.	Brings the program control to the beginning of the loop by skipping the rest of the statements within the block
Program control goes out of the loop even though the test expression is True.	Program control goes out of the loop only when the test expression becomes False.