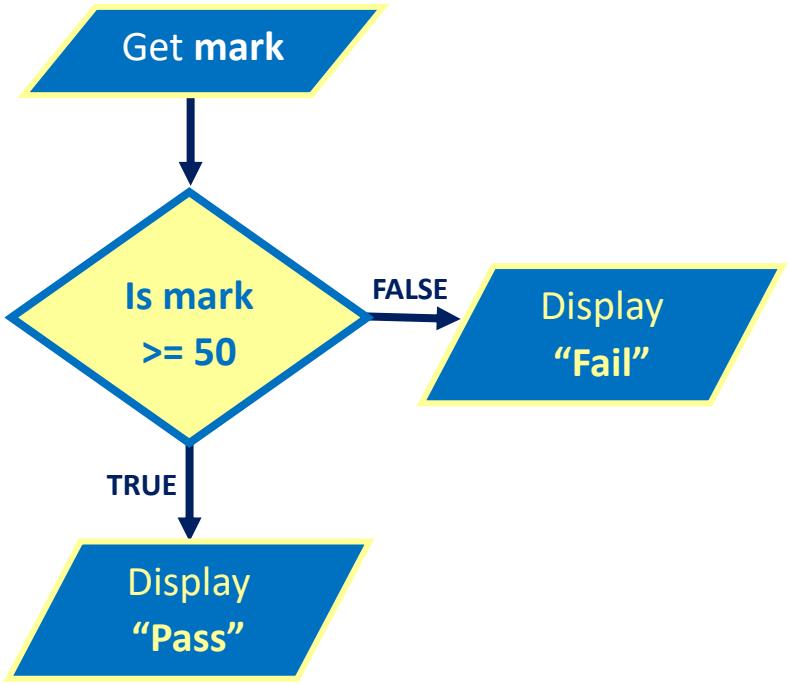
Mr Long Summary ON SELECTION PROGRAMMING

Selection Programming

- THREE ways the code is executed:
  - Sequential – when each line is executed in order from first line to the last line.
  - *Selection – when you select which code will be executed based on a condition.*
  - Iteration – repeatedly execute code for a specific number of times.

- Example of Selection using a flowchart:



IF Statement

General Structure – IF Statement

```
if <condition(s)> then
   begin
    <statement> ;
    ....
    <statement> ;
  end ;
```

- Statements will only be executed if the condition is *TRUE*
    - If condition is *FALSE*, code will ignore statements until end of IF statement

HINT: *Although you don't need the **begin end** if only executing one statement, I suggest always using them so that you don't get confused if you decide to add more statements to the IF statement later.*

What is a condition

- It's a question where the answer is either Yes (*TRUE*) or No (*FALSE*)
- Normally consists of variable or value – relational operator – variable or value
  Example: iMark > 50
  Question: Is the value in variable iMark greater than 50?
- Types of questions commonly asked and the Delphi notation that symbolises them:
    - = Are things equal
    - > Is something greater than
    - < Is something less than
    - >= Is something greater than or equal to (≥)
    - <= Is something less than or equal to (≤)
    - <> Are things NOT equal (≠)

Example

```
if iMark >= 50 then
 begin
   showmessage( 'Pass' ) ;
   Inc( iNumPasses) ;
end ;
```

IF ELSE Statement

General Structure – IF ELSE Statement

```
if <condition(s)> then
   begin
     <statements 1> ;
   end
else
   begin
     <statements 2> ;
   end ;
```

- *<statements 1>* will only be executed if the condition is *TRUE* and will ignore *<statements 2>*.
- If the condition is *FALSE* then *<statements 1>* will be ignored the code will jump to the else segment and executed *<statements 2>*.
- **RULE:** No semicolon is allowed in the line before the else.

Example

```
if iMark >= 50 then
 begin
   showmessage( 'Pass' ) ;
   Inc( iNumPasses) ;
 end
else
 begin
   showmessage( 'Fail' ) ;
   Inc( iNumFails) ;
 end ;
```

AND, OR and NOT operators

- Can make use of multiple conditions with the following operators: AND and OR

AND operator

- All conditions must be *TRUE* for the final result to be *TRUE*
- If one condition is *FALSE*, the final result will be *FALSE*

If <condition 1> **AND** <condition 2> **then**

Condition 1	Condition 2	RESULT
True	True	TRUE
True	False	FALSE
False	True	FALSE
False	False	FALSE

- **RULE:** When using AND or OR, conditions MUST be enclosed in round brackets

  Example: **If** (iNum1 > 50) **AND** (iNum2 > 70) **then**

  *// if **BOTH** iNum1 is above 50 **AND** iNum2 is above 70 then*

OR operator

- Only ONE condition must be *TRUE* for the final result to be *TRUE*
- If ALL conditions are *FALSE*, the final result will be *FALSE*

If <condition 1> **OR** <condition 2> **then**

Condition 1	Condition 2	RESULT
True	True	TRUE
True	False	TRUE
False	True	TRUE
False	False	FALSE

Example: **If** (iNum1 > 50) **OR** (iNum2 > 70) **then**

*// if **EITHER** iNum1 is above 50 **OR** iNum2 is above 70 then*

RULE: When using AND and OR operators in the same If statement, AND operators will be checked first, and then OR operators unless there are brackets.

Example: **If** (iNum1 > 50) **OR** (iNum2 > 70) **AND** (iNum3 > 40) **then**

This If statement is TRUE if iNum1 is greater than 50 OR BOTH iNum2 is greater than 70 and iNum3 is greater than 40.

Example: **If (** (iNum1 > 50) **OR** (iNum2 > 70) **) AND** (iNum3 > 40) **then**

This If statement is TRUE if BOTH iNum3 is greater than 40 AND either iNum1 is greater than 50 and iNum2 is greater than 70.

NOT operator

- Returns the opposite of the condition
  - The final result is *TRUE* if the condition is *FALSE*
  - The final result is *FALSE* if the condition is *TRUE*

If NOT (<condition>) **then**

Condition	RESULT
False	TRUE
True	FALSE

Example: **If NOT** (iNum > 50) **then**
*// if iNum is **NOT** above 50 then <same as if iNum <= 50 then>*

If NOT (iNum = 10) **then**
*// if iNum is **NOT** equal to 10 then <same as if iNum <> 10 then>*

Sets and the IN operator

The following are examples of sets:

- [1 .. 4] *meaning a 1, 2, 3 or 4*
- [1 .. 10] *meaning any integer from 1 to 10*
- [1..10, 12, 21..25] *meaning any integer from 1 to 10 or a 12 or from 21 to 25*
- [‘a’ .. ‘e’] *meaning an ‘a’, ‘b’, ‘c’, ‘d’ or ‘e’*
- [‘a’ .. ‘z’ , ‘A’ .. ‘Z’] *meaning all lowercase or capital letters*
- [‘a’ , ‘e’ , ‘i' , ‘o’ , ‘u’] *meaning all lowercase vowel letters*

- A set is a collection of values of the same ordinal date type (*integer* or *char*).
  - Ordinal data types are data types that have order. You can accurately predict the next value and the value before.
    Example: Integers are ordinal because if you consider the value 5, it is clear that the next value will be a 6 and the previous value would be a 4
    Reals are NOT ordinal because if you consider the value5, it is unclear what the next value will be: 5.1 or 5.01 or 5.0000000001

IN operator

- IN operator can be used with a set to return *TRUE* if a variable or value is contained in the set.

If variable **IN** [set of options] **then**

Example: **If** iGrade **IN** [8..12] **then**
// if iGrade is one of the values from 8 to 12 then

Other examples
- **If** cLetter **IN** ['a'..'z'] **then** *// cLetter is of variable type char*
- **If** iMonth **IN** [4, 6, 9, 11] **then**

- **RULE:** Lowest integer value in a set is 0 and the largest integer value in a set is 255.

Nested IF Statements

General Structure – Nested IF Statement

```
if <condition(s)> then
   begin
     <statements 1> ;
   end
else if <condition(s)> then
   begin
     <statements 2> ;
   end
else if <condition(s)> then
   begin
     <statements 3> ;
   end
else if …
```

- If the first condition is *TRUE* then only *<statements 1>* will only be executed and all other parts of this If statement are ignored until the end of whole nested If.
- If the first condition is *FALSE* then *<statements 1>* will be ignored the code will jump to the second condition and if this is *TRUE* then *<statements 2>* will be executed and all other parts ignored.
- If the first condition is *FALSE*, and then the second condition is *FALSE*, then next condition is checked until it is *TRUE* and then the associated statements will be executed and all other parts are ignored.
  - If ALL conditions are *FALSE*, then no statements in the nested if will be executed unless there is a final else clause at the end with no if statement.

Case Statement

General Structure –Case Statement

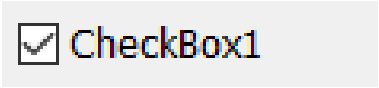
```
case variable of
   <options> : <statement 1> ;
   <options> : <statement 2> ;
   <options> : <statement 3> ;
   ...
   else <statements X> ;     // optional
end ;
```

- Case statement checks the variable with all options to find a match. When a match is found, the corresponding statements for that option is executed.
- If no matches are found, the else statement is executed if there is an else statement.
- **RULES:**
  - Have their own end statement but no begin.
  - Variable must be an ordinal type (integer or char).
  - The line before the else must no have a semicolon.
  - The same value may not be included in more then one set of options.
  - For each option, only ONE statement is executed.
    - If more statements are needed to be executed then ALL statements must be within a ***begin end***.

Example

```
case iNumber of
  < 0          :       showmessage( 'Below zero' ) ;
  1..9         :       showmessage( 'Single digits' ) ;
  10..50       :       showmessage( 'Below 50' ) ;
  51..100      :   begin
                       showmessage( 'Above 50' ) ;
                       Inc( iCount ) ;
                   end                  // no semicolon because of else
  else   showmessage( 'Invalid Number' ) ;
end ;
```

Checkbox component

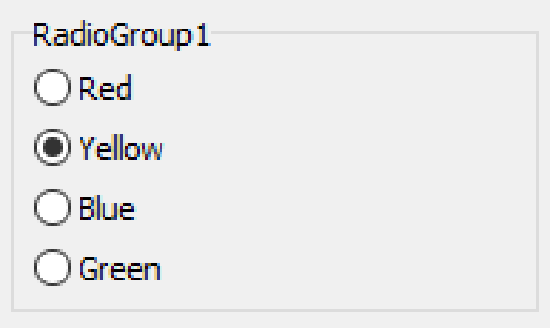


- Used to either select (tick in the block) or deselect (block is empty) an option
- Properties
  - Name – use the *cbx* prefix
  - Caption – text that is displayed next to the block
  - Checked
    - TRUE if it is selected (tick is present)
    - FALSE if it is deselected (block is empty)

Example

```
if cbxOption.Checked = TRUE then
 begin
  showmessage( 'Option is selected' ) ;
 end
else
 begin
  showmessage( 'Option is NOT selected' ) ;
 end ;
```

RadioGroup component



- Used to select from a multiple of choices but only ONE can be selected.
  - Whichever one is selected will contain a black dot in the circle next to the option
  - If another option is selected, the black dot will MOVE to the newly selected option
- **HINT:** *Use the **TRadioGroup** component. Do NOT use the TRadio**Button** component.*

- Properties
  - Name – use the *rgp* prefix
  - Caption – text that is displayed at the top of the radio group block
  - Columns – how many columns to display the options
  - Items
    - Click on ellipse (…)
    - Enter in options in String List Editor
  - ItemIndex
    - -1 means no option is selected
    - 0 means first option is selected
    - 1 means second option is selected
    - 2 means third option is selected
    - N means N + 1 option selected

- **HINT:** *Radio group is best used with a case statement, but a nested If statement is also possible.*

Example

```
case rgpExample.ItemIndex of
  0   :  showmessage( 'Option 1 is selected' ) ;
  1   :  showmessage( 'Option 2 is selected' ) ;
  2   :  showmessage( 'Option 3 is selected' ) ;
  3   :  showmessage( 'Option 4 is selected' ) ;
end ;
```

Additional Links:

- Youtube video playlist:
  https://www.youtube.com/watch?v=r5osw1tOVmo&list=PLxAS51iVMjv_Lp4N7hLU-F7rPA3hlZc9l
- Google drive resource activities:
  https://tinyurl.com/MLE-G10IT-SelectionProgramming