

IF STATEMENTS IN DELPHI

How a program chooses. When else is needed. Why True is tested first.

1. The picture to start with

An If statement is a fork in the road. The program looks at a question that can only be True or False. If the answer is True, it takes the Then road. If the answer is False, it takes the Else road — but only if you built that road.

Say it out loud with the class: “If this is true, do that. Otherwise, do this other thing.” That spoken sentence is the whole structure.

```
if iAge >= 18 then
    ShowMessage('You may vote')
else
    ShowMessage('Too young to vote');
```

2. True is always first — and that is not a style choice

Delphi always works in this order:

1. Evaluate the condition.
2. If it is True, run the Then part. Stop there.
3. If it is False, run the Else part — if there is one.

You never write “False first”. The language is built as if-then-else, not if-false-then. Write the condition so that the Then part is the main / expected / “yes” action. Put the other action in Else.

Good habit: ask a positive question. “Is the password correct?” is clearer than “Is the password not incorrect?”

3. When Else is necessary

Use Else when the program must do something different when the condition is False. Two possible outcomes, two actions.

```
if sAnswer = 'Yes' then
    lblResult.Caption := 'Accepted'
else
    lblResult.Caption := 'Declined';
```

Else is also necessary when you must choose exactly one of two paths (a mark is either a pass or a fail — not neither, not both).

4. When Else is not necessary

Leave Else out when False means “do nothing extra”. The program just continues with the next line.

```
if iStock < 5 then
    ShowMessage('Reorder this item');
```

If stock is 20, there is nothing to say. No Else needed.

A useful classroom test: after the If, ask “If this is false, must the program still do a special job?” If the answer is no, skip Else.

5. One statement vs many — Begin and End

Then and Else each control only the next single statement. Two or more lines need Begin ... End.

```

if iMark >= 50 then
begin
    lblResult.Caption := 'Pass';
    Inc(iPassCount);
end
else
begin
    lblResult.Caption := 'Fail';
    Inc(iFailCount);
end;

```

Without Begin/End, only the first line is part of the If. The second line always runs. This is a very common Grade 10 bug.

6. The semicolon before Else

Do not put a semicolon on the line immediately before Else. That semicolon ends the If, so Else has nothing to belong to and the program will not compile.

Wrong: if iMark >= 50 then lbl.Caption := 'Pass'; else ...

Right: if iMark >= 50 then lbl.Caption := 'Pass' else ...

7. Comparing values — do not mix := and =

Symbol	Meaning	Example
:=	Store a value (assign)	iAge := 16;
=	Ask: are these the same?	if sCode = 'OK' then
<>	Not equal	if sCode <> 'OK' then
< > <= >=	Less / greater (or equal)	if iMark >= 50 then

:= puts a value into a variable. = asks a yes/no question. They are not interchangeable.

8. Boolean variables — you already have True or False

If the variable is Boolean, you do not need = True.

```

if bFound then           // clear
if bFound = True then    // works, but noisy
if not bFound then       // the False path, still as the condition

```

9. More than two options — Else If

When there are bands (fail / pass / merit / distinction), chain them. The first True condition wins. The rest are skipped.

```

if iMark >= 80 then
    sLevel := 'Distinction'
else if iMark >= 70 then
    sLevel := 'Merit'
else if iMark >= 50 then
    sLevel := 'Pass'
else
    sLevel := 'Fail';

```

Order matters. Test the strictest / highest band first. If you test >= 50 first, a mark of 85 is already caught as Pass and never reaches Distinction.

10. Joining conditions

```
if (iAge >= 13) and (iAge <= 19) then    // both must be True
if (sGrade = 'A') or (sGrade = 'B') then // either may be True
if not bPaid then                       // flip True/False
```

Put each comparison in its own brackets when you use and / or. Delphi needs that.

11. A 5-minute class routine

- Read the condition out as a yes/no question.
- Act the Then path with one hand, the Else path with the other.
- Ask: “If the answer is no, must we still do a job?” That decides Else.
- Trace one True input and one False input on paper before they type.
- Only then write the code. Begin/End if there is more than one instruction.

12. One-line summary for tests

If asks a True/False question. Then runs when the answer is True. Else runs when the answer is False, and only if you wrote it. Leave Else out when False means “do nothing”. True is always checked first — write the condition as the yes-question you actually care about.