|  |
| --- |
| Circle Language Spec: Execution Flow |

## Conditional Execution

### Concept

Conditional execution is a form of control over *execution flow*. Execution flow is explained in the article *Execution Flow*.

Conditional execution is a main form of execution flow, where the next step of a program is based on a decision. The If statement is the most common form of conditional execution.

There are four forms in which to express conditional execution:

- If

- Else If

- Select Case (exact value)

- Select Case (split formula)

Each form is explained in a separate article.

### Diagram

Conditional execution is a form of execution flow explained in the article *Conditional Execution*. The articles that follow only explain its expression in a diagram.

There are three forms in which to express conditional execution:

- If

- Else If

- Select Case (exact value)

- Select Case (split formula)

Each form is explained in a separate article. See the articles *If in a Diagram, Else If in a Diagram*, *Select Case in a Diagram (exact value)* and *Select Case in a Diagram (split formula)*.

### If

#### Concept

The If statement is the most common form of *conditional execution*. Conditional execution is a kind of execution flow statement, explained in the *Conditional Execution* article.

The If statement runs a command if a certain condition is met. On top of that, the If statement can run an alternate command when the condition is *not* met.

The If command takes one or two command references as a parameter. That is the command run, when a condition is met and the command run, when a condition is not met. The condition is also passed along to the If command as an argument. The condition is a reference to a Boolean.

The command to run when a condition is met is called the Then clause. The command to run when a condition is *not* met is called the Else clause.

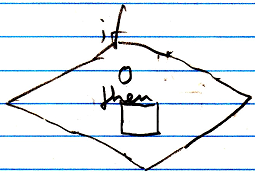
There are two versions of If. There is the If Then statement, that does not have an Else clause. And there is If Else statement, which does have an Else clause.

The implementation of the If command is quite simple. It simply calls a few machine instructions to start the right command, based on whether the Boolean is True or False.

#### Diagram

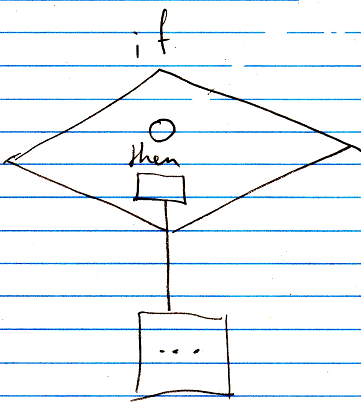
The If Then statement takes a Boolean condition and a reference to the command to run when the Boolean is True.

In a diagram this looks as follows.

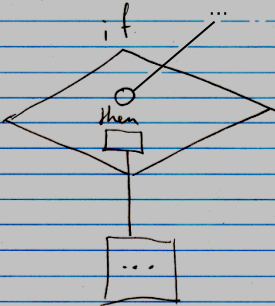


The circle is the Boolean condition. The square is the command to execute when the Boolean condition is True.

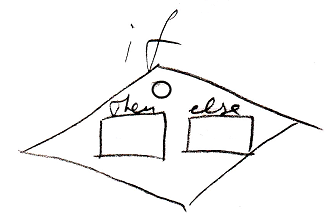
The Then command can be defined right inside the call to the If statement, but you can also define the Then *outside* the If with the aid of an esteatic reference:



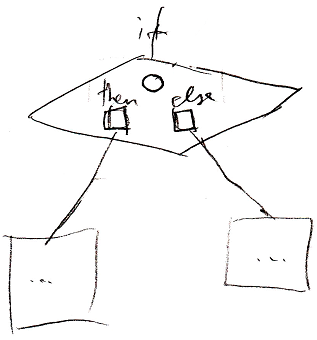
The Boolean condition is usually defined elsewhere as well, which will make the condition a pointer to another symbol.



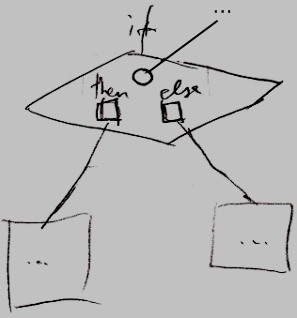
The If Else statement takes a Boolean condition, a command reference to the command to run when the Boolean is True and a command reference to the command to run when the Boolean is False.



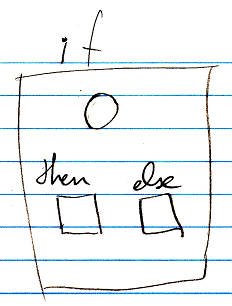
The circle is the Boolean condition. The square named Then is the command to execute when the Boolean condition is True. The square named Else is the command to execute when the Boolean condition is False. For the If Else statement, the Then and Else are also usually defined *outside* the If with the aid of an esteatic reference:



And the condition is usally defined elsewhere as well:



The definition of the If execution flow command is part of a system module of execution flow command. The public elements of the definition look like this:



In the definition, the condition and the clauses are not filled in yet.

### Else If

#### Concept

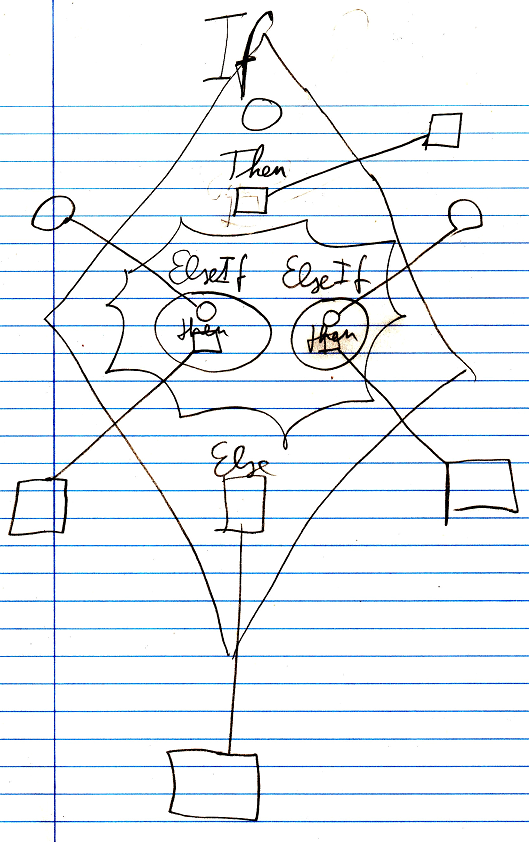
Else If form of *conditional execution*. Conditional execution is a kind of execution flow statement, explained in the article *Conditional Execution*. The Else If statement is a lot like the If statement, but then the Else has another If associated to it. If the condition of the second If is met, then the associated command is called. If the condition is not met, then the Else of the second If might have another If associated to it, and so on, until no more Else If’s are defined. If none of the Else If’s conditions are met, then the final Else is executed, if provided.

The Else If command takes an argument, that is the condition for the first If. It also takes a command reference Then as a parameter. The command is executed if the If condition is True. Furthermore, the Else If command takes a variable amount of Else If’s. Each Else-If-object contains a condition and a reference to the command to execute when the Else If’s condition returns True. The Else If command also take a reference to a command, that will be executed, if none of the Else If’s conditions are True. This alternative command is called the Else clause of the Else If statement. The Else clause of the statement can be left out, if it is not required.

The implementation of the Else If command is quite simple. It simply calls a few machine instructions to start a command, based on whether a Boolean value is True or False.

#### Diagram

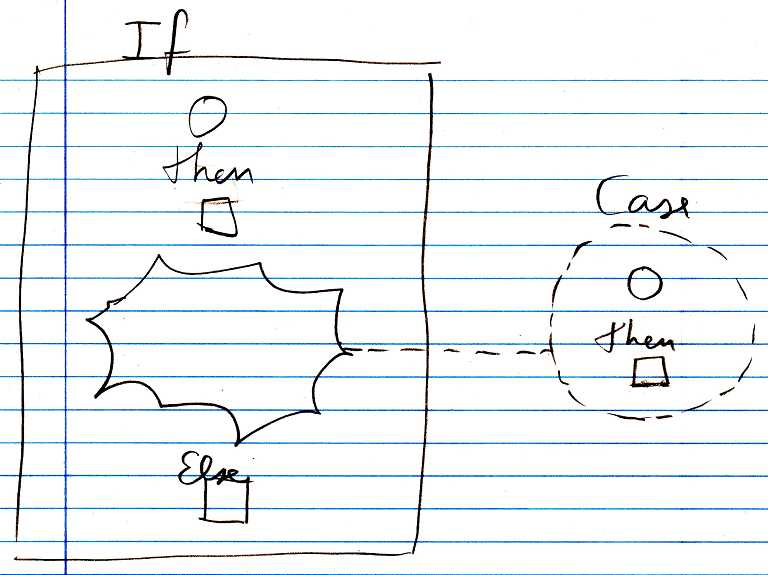
Below is an example of the diagrammatic expression of an Else If statement.



The diamond is the Else If command. the top circle inside the diamond is the condition of the first If. If the condition returns True, then the Then clause is executed. In the middle of the diamond there is a nonagon. Inside the nonagon any number of Else If’s can be specified. An Else If object also has a condition and a Then clause. If the condition returns True then the Then clause is executed. If the condition is False, then the next Else If’s condition is evaluated. If none of the Else If’s conditions return True, then the Else clause is executed, which is visible at the bottom of the diamond. The Else clause is optional. If the Else clause is not used, it can be left out of the diagram.

The conditions and the clauses can all be references to something defined outside the diamond. The conditions and the clauses can also be filled in right inside the diamond.

The *definition* of the Else If command is part of a system module of execution flow commands. The public elements of the definition look like this:



Nothing is filled in yet for the condition, the Then clause or the Else clause, and there are no Cases defined yet. But a *class* for a Case *is* defined. The Case class defines a condition and a Then clause.

There is another, separate definition of the Else If command, that is the same as the other Else If command definition, except that it does not have an Else clause in it.

### Select Case

#### Concept

The Select Case statement is a form of conditional execution. Conditional execution is a kind of execution flow statement, explained by the article *Conditional Execution*.

In a Select Case statement the next step to take is one out of several options.

There are two forms of Select Case:

- Select Case (exact value)

compares a variable to different values, to choose the next step

- Select Case (split formula)

combines one half of a formula with several other halves of the formula, to choose the next step

The two forms will be explained in separate articles.

The name Select Case is directly taken over from the programming language *Basic*. You can choose between *Basic* naming and *C* naming for execution flow statements. In the programming language *C* it is called a Switch statement. Both names are available in the new computer language, as part of the multi-lingual approach of the system.

#### Diagram

Each type of Select Case has a slight variation in diagram notation. There are three definitions of Select Case commands. They will be covered in the articles *Select Case (exact value) in a Diagram*, *Select Case (split formula) in a Diagram*. (*Split formula* has *two* variations. Hence the three definitions of Select Case.)

### Select Case (exact value)

#### Concept

There are two forms of Select Case, as mentioned in the article *Select Case*. This article explains the form of Select Case where a variable is compared to different values, to choose the next step to take.

This form of Select Case compares a given variable with several different values. If the variable equals one of the values, the step associated with that value is executed. If the variable equals none of the values, an alternative command can be executed.

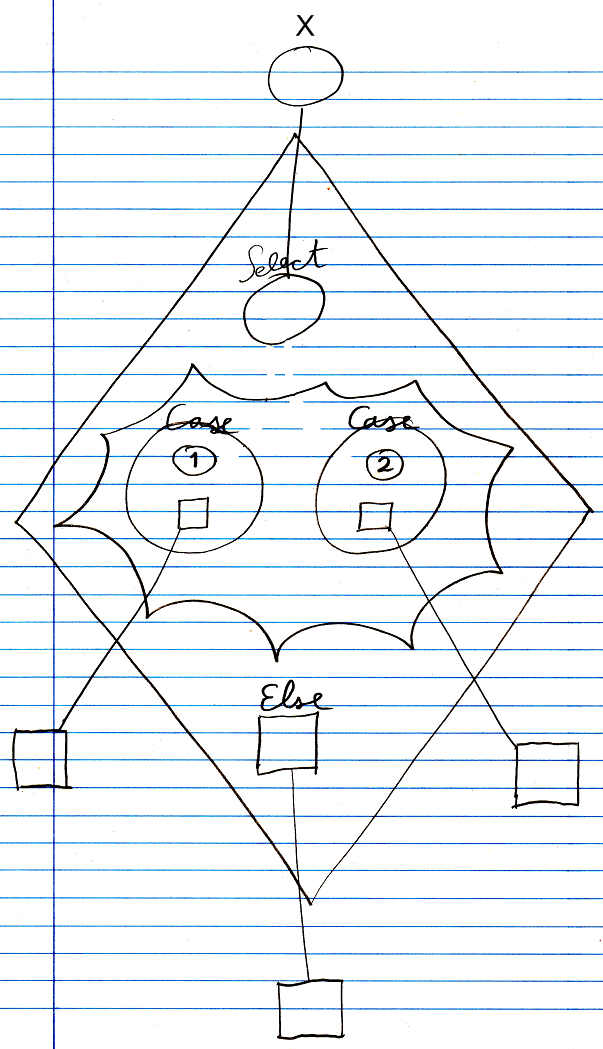
This type of Select Case only works with objects, that hold a binary value.

Select Case takes an object, that holds a binary value as the Variable of the comparison. Furthermore, Select Case defines a variable amount of Cases. Each Case-object contains a value to compare the variable to and a reference to the command to execute when the variable equals the value. Select Case can also take a reference to a command, that will be executed, when the variable matches *none* of the values. This alternative command is called the Else clause of the Select Case statement. The Else clause of the statement can be left out, if it is not required.

The implementation of the Select Case command is not too complicated. It simply calls a few machine instructions to compare a variable to a value and to start the right command when a match is found.

#### Diagram

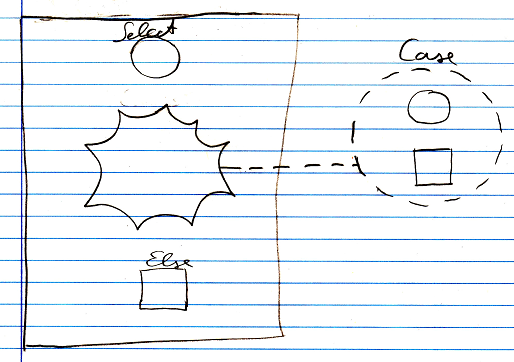
Below is an example of the diagrammatic expression of a Select Case statement for comparing exact values.



The diamond is a call to the Select Case command. The circle inside the diamond has the title Select. It points to an object outside the command call. This will be the variable to which several values will be compared. In the middle of the diamond there is a nonagon, that represents the cases: values to which the variable will be compared. The nonagon can contain any number of cases. Each circle inside a nonagon is a Case. Each case defines a value, and the command to call, if the variable matches the value. In this example, there are two Cases. One Case has the value 1 and the other Case has the value 2. Each Case has a command associated to it. Those command references are pointing to clauses defined outside the diamond. At the bottom of the diamond there is also a command reference called Else. It points to a clause defined outside the diamond. The command pointed to will be called if the variable matches none of the values defined in the Cases. If the Else clause is not used, it can be left out of the call and then it will not be shown in the diagram.

The values for the cases were entered litterly into the case. A value for a case can also be defined as a pointer to an object that holds the value. The same way, the variable could also have gotten an exact value, and not be a pointer to an object outside the diamond. The command references did not have to point to something defined outside of the diamond either. The commands could have been defined right inside the diamond, but it often looks more intuitive to define clauses outside the diamond.

The *definition* of the Select Case execution flow commands is part of a system module of execution flow commands. The public elements of the definition look like this:



Nothing is filled in yet as the Select or Else, and there are no Cases defined, but a *class* for a Case *is* defined.

There is another, separate definition of the Select Case command for comparison of exact values, that is the same as the other Select Case command definition, except that it does not have an Else clause in it.

### Select Case (split formula)

#### Concept

There are two forms of Select Case, as mentioned in the article *Select Case*. This article explains the form of Select Case where one half of a formula is combined with several other halves of the formula, to choose the next step to take.

This form of Select Case takes a fixed first part of a formula. Then it combines it to several other halves of the formula. If the complete formula returns the Boolean value True, the step associated with that other half of the formula is executed. If none of the resultant formulas renders the Boolean value of True, then an alternative step can be executed.

Half a formula can either be a value, or an operation for which one operand is yet to be filled in.

If the first half of a formula is a value, then the other halves of the formula need to be operations for which an operand is yet to be filled in. The first half of the formula will then be filled in as the operand missing in the other half of the formula.

If the first half of a formula is an operation for which one operand is yet to be filled in, then the other halves of the formula need to be values, to fill in as the operand.

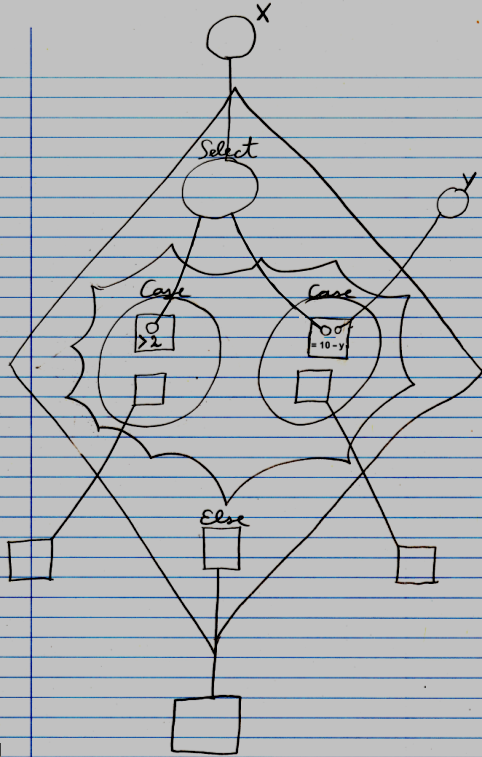
It is not limited to just mathematical formulas. You can use any command for which a parameter is to yet to be filled in. This type of Select Case is not limited to objects, that hold binary data. This type of Select Case works with all kinds of objects. However, the result of the resultant formula has to be a Boolean value.

The implementation of the Select Case command one by one calculates the Boolean results of the resultant formulas. If the result of formula is True, then the command associated with that second half of the formula is called. If multiple resultant formulas return True, then all the associated commands are executed. If all resulatant formulas were processed and none of the formulas returned True, then the alternative command is run.

#### Diagram

There are two forms of this statement: the first half of the formula is a value or the first half of the formula is an operation with an operand missing. These two forms have a different definition and look different in the diagram

Below is an example of the diagrammatic expression of a Select Case statement for split formulas, of which the first half of the formula is a value.

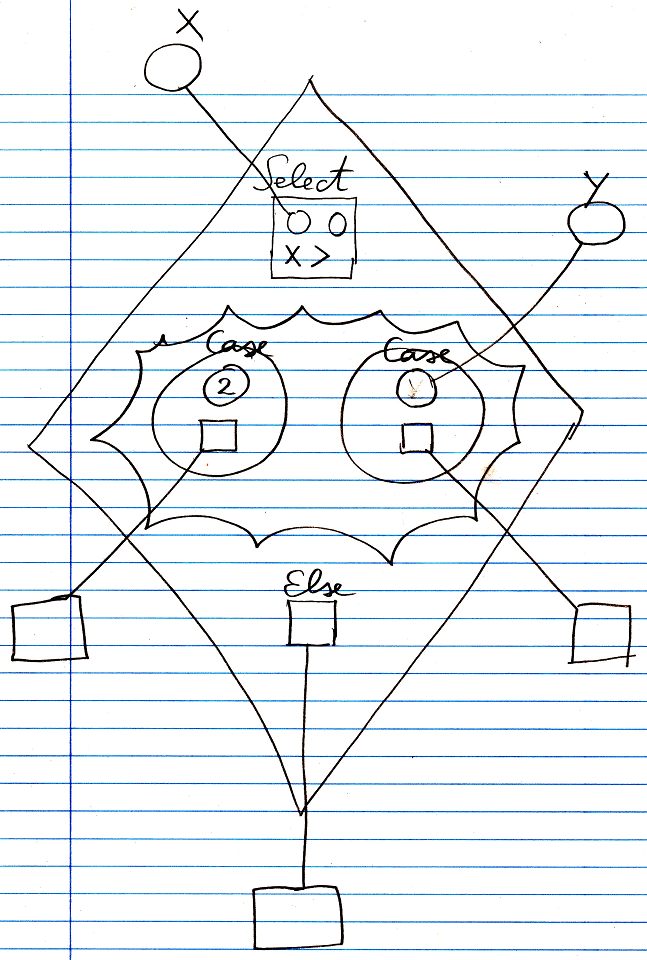


The diamond is a call to the Select Case command. The circle inside the diamond has the title Select. It points to an object outside the command call. This is called the variable. It is the value, that will be filled into the missing operands of the other halves of the formula. In the middle of the diamond there is a nonagon, that represents the cases: different operations into which the variable will be filled in. The nonagon can contain any number of cases. Each circle inside a nonagon is a Case. Each case defines a command for which the variable is filled in. Each case also defines the command to call when the result of the formula is True. In this example, there are two Cases. The other halves of the formula are not drawn out in full detail. That would obscure the picture in this demostration. The literals of the half formulas are shown. The command definitions of the half formulas are not pointed out, and the build-up of the formula’s is not fully graphically drawn out with objects connected with operations, because that would obscure the picture of this demonstration, but they do belong in the diagram, though.

One Case is the half formula > 2 . X is to be filled in as the first operand of this formula, which would make the resultant formula X > 2. The second Case is the half formula = 10 – Y. X is to be filled in as the first half of the formula, which would make the resultant formula X = 10 – Y.

Each Case has a command associated to it. Those command references are pointing to clauses defined outside the diamond. When a resultant formula of the case returns True, then the command associated with the case will be executed. At the bottom of the diamond there is also a command reference called Else. It points to a clause defined outside the diamond. The command pointed to will be called if *none* of the formulas results in the Boolean value True. If the Else clause is not used, it can be left out of the call and then it will not be shown in the diagram.

Below is an example of the diagrammatic expression of a Select Case statement for split formulas, of which the first half of the formula is half a formula, and the second halves of the formula are values to be filled in as the missing operand in the first half of the formula.

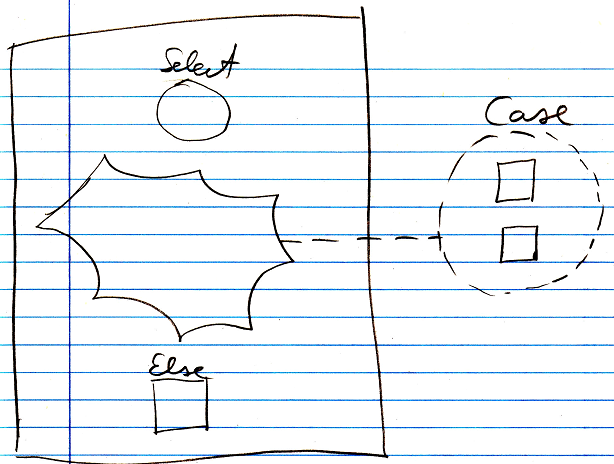


The diamond is a call to the Select Case command. The square at the top of the diamond is the first half of the formula. It is a *greater than* operation. The first operand of the formula is already filled in with the object X, by having the parameter point out of the diamond to the object called X. The second parameter of the operation is yet to be filled in by the Cases of the Select Case statement. In the middle of the diamond there is a nonagon, that represents the cases: different values to be filled in as the missing operand of the formula. The nonagon can contain any number of cases. Each circle inside a nonagon is a Case. Each case defines a value to be filled in as the missing operand of the formula. Each case also defines the command to call when the result of the formula is True. In this example, there are two Cases. One Case is the value 2. This value will be filled in as the missing operand of the formula. This makes the resultant formula X > 2. The second Case is not a fixed value, but points to the variable Y, which is defined outside the diamond. Y is filled in as the missing operand of the formula. This makes the resultant formula X > Y.

Each Case has a command associated to it. Those command references are pointing to clauses defined outside the diamond. Every command for which the resultant formula returns True is executed. At the bottom of the diamond there is also a command reference called Else. It points to a clause defined outside the diamond. The command pointed to will be called if none of the formulas results in the Boolean value True. If the Else clause is not used, it can be left out of the call and then it will not be shown in the diagram.

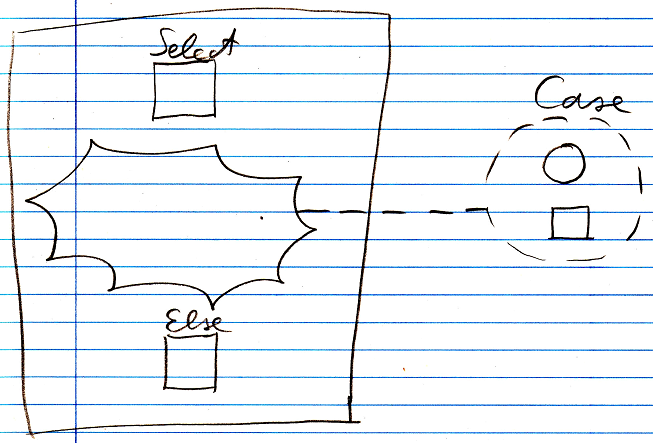
In the examples above, each value, that was litterly filled in, could also have been a pointer to something remote. Conversely, everything that was a pointer to something outside the diamond, could also have been defined directly inside the diamond.

The definition of the Select Case execution flow commands is part of a system module of execution flow commands. The public elements of the definition for a value as the first part of the formula looks like this:



Nothing is filled in yet as the Select or Else, and there are no Cases defined, but a *class* for a Case *is* defined. There is also a definition without an Else clause in it.

The public elements of the definition for half a formula as the first part of the formula looks like this:



Nothing is filled in yet as the Select or Else, and there are no Cases defined, but a *class* for a Case *is* defined.

There is another, separate definition of the Select Case command for formulas, that is the same as the other Select Case command definition, except that it does not have an Else clause in it.