9 Using Formulas

In this chapter:

9 Using Formulas

Formulas are an important part of Informed's data intelligence capabilities. With formulas, you can create cells whose values are calculated or checked when you fill out a form with Informed Filler. You can also use formulas to configure a dynamic tab order for cells on a form. In this chapter you'll learn about the following topics:

- the uses of formulas
- operands and operators
- constants and functions
- conditional formulas and the IF statement
- formula results
- precedence rules
- automatic type conversion

For information about entering formulas, see "Calculations," "Data Verification," and "Conditional Tabbing" in Chapter 1. Functions are explained in detail in Chapter 10.

Overview

The most common use of formulas is to calculate a cell's value based on the values of other cells on your form. Calculated cells are automatically entered for you when you fill out a form with Informed Filler or when you test your form with Informed Designer. For example, a cell can use a formula to calculate an extended price based on a quantity and a unit price.

You can use formulas to check for data entry errors. A check formula can test for different error or warning conditions. For example, you could use a check formula to make sure that the discount rate on a sales slip doesn't exceed twenty percent of the total sale. You can even use alert dialog boxes and help messages to describe error or warning conditions to the person filling out the form.

You can also use formulas to calculate the next tab position for any cell. The result of a tab formula is the name or tab position of the next cell to tab to. By using tab formulas, you can dynamically change the tab order of a form according to different conditions.

Formulas combine values, called *operands*, with special symbols, called *operators*, to derive a particular result. Each operator applies a particular action on its operands. For example, the following formula uses the addition operator (+) to add 2 to the value contained in 'Cell1.'

```
Cell1 + 2
```

The result of a formula is used according to where the formula appears. It can be the calculated value of a cell, a function parameter, a check condition, a tab condition, or it can be part of a larger formula.

Operands

The values from which a formula derives its result are called *operands*. You combine operands with operators to create a formula that produces the defined result. An operand can be:

- a value that you enter directly into a formula
- the value of a cell
- the result of a function
- the result of another formula

The following example shows a formula that uses all four kinds of operands. It adds 2 to the value in ‘Cell1’ and then multiplies that result by the sum of 2, 4, and 8.

```
(2 + Cell1) * Sum (2, 4, 8)
```

When you use an operand with an operator, the operand’s type must match the type expected by the operator. For example, the multiplication operator (*) expects number operands. If the types don’t match, Informed will try to convert the operand’s value to the appropriate type. You can also use the type conversion functions (ToNumber, ToText, and so on) to explicitly convert an operand’s value to a different type. For more information about type conversion and type compatibility, see “Type Conversion” later in this chapter.

Constants

Constants, sometimes called *literals*, are values that you enter directly into a formula. The term constant is used to describe these values because they don’t change; they remain constant. For example, the second operand in the following formula always evaluates to the numeric value -1. The result of the formula is always -1 plus the value in the cell called ‘Cell1.’

```
Cell1 + -1
```

If you use a formula where all the operands are constants, the formula will always return the same result. For example, the following formula always returns the date August 23, 1996.

```
ToDate ("August 3, 1996") + 20
```

Three kinds of constants are used in the above example: a number constant, a text constant ("August 3, 1993"), and a date constant (the result of the ToDate function). You can also use time, name, and boolean constants as operands in formulas.

Number Constants

A number constant consists of an optional sign indicator (+ or -) followed by one or more digits. A decimal point can appear anywhere in the digits. Some example number constants are:

```
45
+3.33
-.05
89.99
-545.63
-5.
```

Informed provides two predefined number constants. The values of pi (3.141592654...) and e (2.718281828...) are represented by the constants shown below.

Predefined Number Constants

Constant	Value
Pi or π or ∏	3.141592654
e	2.718281828

Text Constants

Text constants must be enclosed in single or double quotation marks. Any typeable character can appear in a text constant. If the text constant itself contains the same type of quotation marks as those enclosing it, you must precede the quotation mark with the *backslash* character (\). To enter the backslash character, type the Backslash key. The backslash character tells Informed that the next character is part of the text constant. The following table lists some text constants. Note that the constant "\t" adds a tab, and constant "\n" adds a carriage return. ASCII codes can also be entered as text constants. For example, to enter a carriage return into a text string you type "\013." The carriage return is inserted when the formula is evaluated.

Text Constants

Text Constant	Text Value
`"Received - code #546"`	`Received - code #546`
`"Received - code \t#546"`	`Received - code  #546`
`"Received - code \n#546"`	`Received - code` `#546`
`'For this office\'s use only.'`	`For this office's use only.`
`"Don't remove this label!"`	`Don't remove this label!`
`"Welcome to \"Herb's Diner\""`	`Welcome to "Herb's Diner"`
`'A.'`	`A.`

Boolean Constants

You can use two predefined boolean constants. The boolean values of "True" and "False" are represented by the same words typed without the enclosing quotation marks.

Name, Date, and Time Constants

To enter name, date, or time constants, you must use the ToName,ToDate, and ToTime functions. These functions convert values from other types to the name, date, and time data types. You enter a name, date, or time constant by enclosing the textual representation of the value in single or double quotation marks and entering that value as the single parameter to the appropriate function. For example, you'd enter the time constant 12:51 PM by typing ToTime ("12:51 PM").

When you enter the textual representation of a name, date, or time constant, you can enter the value in any format that Informed will recognize. For example, the two date constants below are equivalent.

```
ToDate ("January 1, 1996")
ToDate ("1/1/96")
```

If the result of the function is used as a cell's calculated value, the format of the cell determines how the value is displayed on your form. See the "Cell Types" section in Chapter 1, "Adding Intelligence to Your Forms." for more information.

Cell References

Formulas can reference information in other cells on your form. To reference a cell, you type the cell's name in the formula. When Informed evaluates the formula, the cell's name is replaced with the current value of the cell.

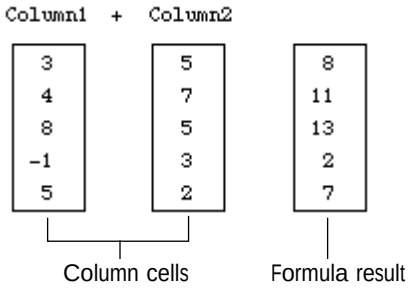
For example, suppose that your form contains a cell called 'Quantity,' which contains the number of items, and a cell called 'Price,' which contains the price of a single item. You could use the following formula to calculate the extended price.

```
Quantity * Price
```

Note Never enclose the name of a cell within quotation marks. A cell name within quotation marks would be interpreted as a text constant and not as a cell value.

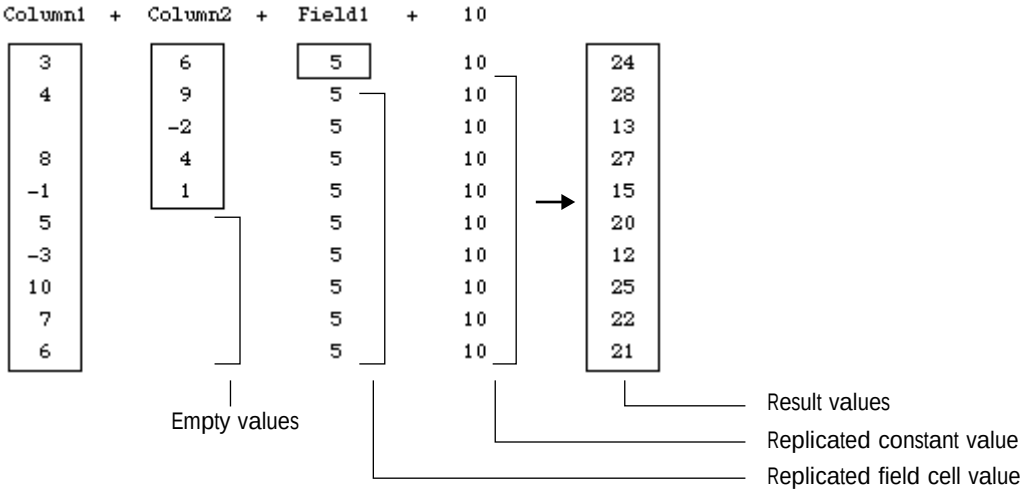
Column Cells

When you use a column cell as an operand in a formula, the formula's result will also be a column. That is, the result will consist of multiple values. For example, if you use the addition operator to add two column cells, each containing five rows, the formula's result will also consist of five rows. Each row in the result will contain the sum of the corresponding rows in the two column cells.



If the column cells in the above example were of different heights, the formula's result would contain as many values as the longest column. Informed would insert empty values in the missing rows of the shorter column cell.

If you use column cells with field cells or constants in the same formula, the field cell and constant values are applied to each row of the column cells. The figure below illustrates a formula that adds a field cell, a number constant, and two column cells.



You can use column cells in a formula only when the result of the formula is also expected to be a column. If a single value result is required, Informed Designer won't allow you to use a column cell in the formula. For example, suppose that your form contains the field cell called 'Total' and the column cells called 'Quantity' and 'Price.' Informed Designer would not allow you calculate the 'Total' cell as 'Quantity * Price.'

You can reference a specific row of a column cell in your formula. For example, to reference the third row of the column cell 'Quantity' you would enter 'Quantity[3].' The row identifier can be a numeric constant, or a formula that returns a number.

Functions

A *function* performs a predefined calculation using a set of input values, called *parameters*. For example, the following function calculates the mean of the numbers 1, 8, and 12.

```
Mean (1, 8, 12)
```

You can use the result of a function's calculation as an operand in a formula. A function can appear in a formula anywhere a cell reference or a constant can appear. As with constants and cells, the result type of a function should match the type expected by the operator with which the function is used. See "Type Conversion" for more information.

When you use a function as an operand, the function is evaluated first. Its result value is then used to evaluate the formula. For example, the following formula calculates the sum of 7—which is the result of the 'Mean' function—and 10 to give a result of 17.

```
10 + Mean (1, 8, 12)
```

For more information about functions, parameters, and function results, see Chapter 10, "Using Functions."

Formulas as Operands

You can use a formula as an operand to another formula. This allows you to create complex formulas that consist of many operators and operands. For example, the formula 'Cell1 * Cell2' is used as the first operand to the addition operator (+) in the formula below.

```
Cell1 * Cell2 + Cell3
```

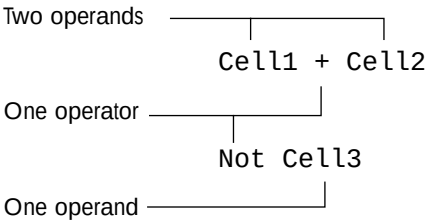
In the above example, the formula 'Cell1 * Cell2' is evaluated first. Its result value is then used as the first operand to the addition operator.

You can combine any number of operands and operators. There are rules, called *precedence* rules, that determine which formulas are evaluated before others. In the above example, the precedence rules dictate that the multiplication operator (*) is evaluated before the addition operator (+). For a complete description of the precedence rules, see "Precedence" later in this chapter.

Operators

Operators combine operands to create new values. To create a new value, each operator performs a particular action on its operands. The type of the new value depends on the operator's action and on the types of operands used. Example actions are summation, comparison, and negation.

Most operators combine two operands to produce a new result. Others manipulate only one operand, or any number of operands. For example, the addition operator (+) adds two operands, whereas the boolean negation operator (Not) negates a single operand.



Each operator requires that its operands be a certain type. For example, since the multiplication operator multiplies numbers, its operands must be number values. When an operand's type doesn't match the type required by the operator, Informed will try to automatically convert the value to the correct type. See "Type Conversion" for more information.

Informed provides five kinds of operators:

- arithmetic
- text
- comparison
- boolean
- column

Arithmetic operators perform mathematical operations such as addition and multiplication. Text operators manipulate text values to produce new text values. Comparison operators compare values to check for conditions or make choices. Boolean operators combine boolean values to produce new boolean values. The list operator ({}) produces a list of any number of values.

Arithmetic Operators

Arithmetic operators allow you to create formulas that add, subtract, multiply, divide, and exponentiate numbers. You can also use arithmetic operators to find the quotient and modulus of a division, negate a number, and convert a number to a percentage value.

The symbols for the arithmetic operators are shown in the following table. Along with each operator, the table lists the number of operands the operator requires and a simple example. The result of each example follows the arrow symbol (→).

Arithmetic Operators

Operator	# of Operands	Description	Example
+	2	addition	2 + 3 → 5
-	2	subtraction	2 - 3 → -1
*	2	multiplication	4 * 5 → 20
/	2	division	18 / 4 → 4.5
Div	2	quotient	18 Div 4 → 4
Mod	2	modulus	18 Mod 4 → 2
^	2	exponentiation	4 ^ 3 → 64
%	1	percentage	1.5% → 0.015
-	1	negation	-(2 + 3) → -5

As shown above, you use arithmetic operators to create mathematical equations. For example, the following formula uses the multiplication and percentage operators to calculate 25 percent of the value in ‘Price.’

```
Price * 25%
```

You can also use the addition and subtraction operators to add and subtract date and time values. You can add days to a date and seconds to a time. You can also subtract one date or time from another. The following table illustrates addition and subtraction of dates and times with examples. To avoid repetition, the ToDate and ToTime functions are not used to create proper date and time constants.

Addition and Subtraction of Dates and Times

Operator	Operand Types	Example
+	date + number	May 12 + 2 → May 14
	time + number	5:45:30 + 14 → 5:45:44
-	date - number	Dec 22 - 2 → Dec 20
	time - number	12:30:15 - 14 → 12:30:01
	date - date	10/24/96 - 10/14/96 → 10
	time - time	14:20:10 - 14:10:00 → 610

Subtracting a date from another date or a time from another time yields the number of days between the two dates or seconds between the two times. A negative result indicates that the first operand is an earlier date or time that the second. For example, if ‘Birth date’ is a date cell containing your date of birth, the formula ‘Today - Birth date’ calculates your age in days (the Today function returns the current date).

Text Operators

Informed provides you with one text operator. This operator allows you to join two text values to create a new larger text value. The operator is called the *concatenation* operator. It appears in the following table along with an example.

Text Operator With Example

Operator	Operand Types	Description	Example
&	text & text	concatenation	`"In" & "formed" → "Informed"`

You use the concatenation operator to construct new text values. For example, if 'Airline' is a text cell containing the name of an airline, the formula below creates a message that indicates the airline on which a flight has been booked.

```
"Your flight is booked on " & Airline
```

You can build complex messages by using the result of one concatenation operator as the operand to another.

```
"Your flight is booked on " & Airline & ". Have fun!"
```

If the value of 'Airline' is "Sunny Airlines," the formula returns the result "Your flight is booked on Sunny Airlines. Have fun!."

Note You can also use the 'Concat' function to concatenate text values. See Chapter 10 for a complete description of this function.

Comparison Operators

Comparison operators compare two values. For example, you can compare two numbers to determine if one is greater than, equal to, or less than the other. The result of a comparison operator is always a boolean value; either True or False. The comparison operators are shown in the following table. Each operator is shown with the types of operands allowed and an example using constant values. Again, to avoid repetition, the type conversion functions are not used to create proper date, time, and name constants.

Comparison Operators With Examples

Operator	Operand Types	Description	Example

=	number = number
	date = date
	time = time
	text = text
	name = name
	boolean = boolean
		equals	`4 = 5 → False`
			`May 5 1996 = 05/05/96 → True`
			`12:30:00 = 12:30:01 → False`
			`"aaaaA" = "aaaa" → False`
			`A. New = Al New → False`
			`True = True → True`

Comparison Operators With Examples (continued)

Operator	Operand Types	Description	Example
<>	number <> number	not equals	4 <> 5 → True
	date <> date		May 5 96 <> 05/05/96 → False
	time <> time		12:30:00 <> 12:30:01 → True
	text <> text		"After" <> "Before" → True
	name <> name		A. New <> Al New → True
	boolean <> boolean		False <> False → False
<	number < number	less than	5 < 6.5 → True
	date < date		01/01/92 < 05/05/93 → True
	time < time		12:30 PM < 12:30:01 → False
	text < text		"ABC" < "abc" → True
	name < name		J Smith < A Smith → False
	boolean < boolean		True < False → False
>	number > number	greater than	8.9 > 9 → False
	date > date		10/05/93 > 05/05/93 → True
	time> time		12:30:00 > 12:30:01 → False
	text > text		"123" > "abc" → False
	name > name		J Smith > A Smith → True
	boolean > boolean		True > False → True
<=	number <= number	less than or	1 <= 0 → False
	date <= date	equal to	Jan 1 <= May 5 → True
	time <= time		15:00:00 <= 15:00:00 → True
	text <= text		"Dog" <= "cat" → True
	name <= name		J Smith <= J Jones → False
	boolean <= boolean		False <= True → True
>=	number >= number	greater than	5.5 >= 5.5 → True
	date >= date	or equal to	10/05/95 >= 10/05/93 → True
	time >= time		12:30:00 >= 12:30:01 → False
	text >= text		"say 123" >= "say 456" → False
	name >= name		J Smith >= J Jones → True
	boolean >= boolean		True >= False → True

The comparison operators compare values of the same type. If the operand types differ, Informed will convert both values to text and compare the two text values.

The comparison operators return boolean results True or False. You use comparison operators to create check formulas or formulas that set the values of boolean cells. You also use comparison operators with the IF statement to make decisions or select different actions. The following example checks if the value of ‘Quantity’ is greater than 5. If ‘Quantity’ is greater than 5, the value in ‘Price’ is discounted by 5 percent. If ‘Quantity’ is less than or equal to 5, the value in ‘Price’ is returned unadjusted.

```
If (Quantity > 5) Then
    Return Price * 95%
Else
    Return Price
End
```

When you compare two values, Informed follows the comparison rules outlined below.

- Numbers are compared numerically. Larger numbers are greater than smaller numbers, and negative numbers are less than positive numbers.
- Dates and times are compared temporally. A “later” date or time is greater than an “earlier” date or time.
- Text values are compared using the character ordering called ASCII. In general, digits are ordered before capital letters, and capital letters are ordered before small letters. Two text values are compared character by character from left to right until they differ. The relative ordering of the differing characters determines which is the “greater” text value. The result of each of the following comparisons is True.

  ```
  "Hello" = "Hello"
  "123 Street" < "Whyte Avenue"
  "Informed" > "Inform"
  "begin" < "end"
  "Design West" < "design West"
  ```

- Names are compared part by part. The parts of two names are compared in the following order: surname, first name, middle names, prefix, then suffix. If one of the names doesn’t contain a part and the other name does, the name that’s missing the part is the “lesser” name (assuming the names are equal up to that part).
- The boolean value True is greater than False.
- Pictures and Signatures. You can compare two picture values or signature values to see if they are equal.

Boolean operators

Boolean operators perform the logical operations of “and,” “or,” and “not.” Use boolean operators to check if different boolean values or comparisons are True or False. The following table summarizes the boolean operators.

Boolean Operators With Examples

Operator	Description	Example

And	logical “and”	`True And True → True`
		`True And False → False`
		`False And True → False`
		`False And False → False`

Boolean Operators With Examples (continued)

Operator	Description	Example

Or	logical “or”	`True Or True → True`
		`True Or False → True`
		`False Or True → True`
		`False Or False → False`

Not	logical negation	`Not True → False` `Not False → True`

You can use boolean operators to decide if one or more conditions are True. The condition could be the value of a boolean cell on your form or it could be the result of a formula that compares two values using a comparison operator. Use the And operator to check if two conditions are True simultaneously. For example, the formula below checks if the value in ‘Discount’ is greater than or equal to zero and less than 15.

```
Discount >= 0 AND Discount < 15
```

Similarly, you can use the Or operator to check if at least one of two conditions is True. The following formula checks if the value in ‘Discount’ is less than zero or greater than 15.

```
Discount < 0 OR Discount > 15
```

Boolean operators are often used with comparison operators and the ‘If’ statement to select different actions. The following example checks if ‘Override’ is not True and if ‘Size’ is less than 25. If both conditions are True, the text value “OK” is returned. If ‘Override’ is True or if ‘Size’ is greater than or equal to 25, the text value “Not OK” is returned instead.

```
If Not Override AND Size < 25 Then
    Return "OK"
Else
    Return "Not OK"
End
```

The Column Operator

The column operator consists of the two brace characters “{” and “}.” Use these characters to create a column consisting of multiple values. Simply enclose multiple values, separated with commas, within the brace characters. A column result can be used to set the value of a column cell. The example below uses the column operator to create a column containing the rows of Column1 followed by the rows of Column2.

```
{Column1, Column2}
```

The MakeColumn function can be used in place of the column operator. For information on this and other functions, see Chapter 10, “Using Functions.”

The ‘If’ Statement

The ‘If’ statement provides you with a decision making capability for your formulas. A formula can return different results based on different conditions.

Note You can also use the IFT and IFTE functions in place of the ‘If’ statement. See Chapter 10, “Using Functions” for more information.

The condition of an ‘If’ statement can be any formula or function that returns a boolean result. For example, the formula below returns the value of ‘Cell3’ multiplied by 100 only if the value of ‘Cell2’ is not equal to zero.

```
If Cell2 <> 0 Then
    Return Cell3 * 100
End
```

The condition for the above ‘If’ statement is ‘Cell2 <> 0.’ If the value of ‘Cell2’ is zero, the formula returns the empty value.

You can select between two different actions by using the word ‘Else’ with the ‘If’ statement. If the condition being checked by the ‘If’ statement is False, the formula after the ‘Else’ is evaluated. For example, the formula below returns the value “Senior” if ‘Age’ is greater than 65.

```
If Age > 65 Then
    Return "Senior"
Else
    Return "Junior"
End
```

If the value of ‘Age’ is not greater than 65 (that is, if ‘Age’ is less than or equal to 65), the formula returns “Junior” instead.

The ‘If’ statement is commonly used to create check formulas. A check formula is a formula that checks for different error or warning conditions. The result of a check formula tells Informed if the entry in a cell is valid or not. For example, the formula below checks if the value in ‘Shipping’ is at least 5 and less than 20.

```
If (Shipping >= 5) AND (Shipping < 20) Then
   Return True
Else
   Return False with Alert 'Discount out of range.'
End
```

If the ‘If’ condition evaluates to False—which means the value in ‘Shipping’ is out of range—the formula returns the value False and displays an alert message. For more information about creating check formulas, see “Data Verification” in Chapter 1, “Adding Intelligence to Your Forms.”

You can extend the ‘If’ statement further by using the word ‘ElseIf.’ With this word you can choose between several alternative actions. An ‘If’ statement can contain zero or more ‘ElseIf’ terms and zero or one ‘Else’ term. If you use an ‘Else’ term, it must appear last. The following ‘If’ statement uses several ElseIf terms and an ‘Else’ term to choose between several actions. It returns the value in ‘Price’ scaled by a factor that is determined by the value in ‘Age.’

```
If Age < 18 Then
    Return Price * 50%
ElseIf (Age >= 18) AND (Age < 45) Then
    Return Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Return Price * 150%
Else
    Return Price * 175%
End
```

You can use an ‘If’ statement as an action of another ‘If’ statement. This is commonly referred to as *nesting* (one ‘If’ statement is nested inside another). Consider the formula below.

```
If TotalSale < 5000 Then
   If Discount Rate > 0.15 Then
      Return False with Alert 'The discount rate
      cannot exceed 15%.'
   End
Else
   If Discount Rate > 0.20 Then
      Return False with Alert 'The discount rate
      cannot exceed 20%.'
   End
End
```

This formula checks for an invalid discount rate. The outermost ‘If’ statement checks if the total sale is less than $5,000. Depending on the result of that condition, the formula then checks for a discount rate that’s either greater than 15 percent or greater than 20 percent.

Precedence

Precedence rules are the rules that determine the order in which parts of complex formulas are evaluated. When you create a formula with more than one operator, the precedence rules dictate which operator is evaluated first, which is evaluated second, and so on. Without precedence rules, the same formula could yield different results, depending on the order in which its operators are evaluated. For example, the formula below could be evaluated two different ways.

```
2 * 4 + 3
```

The multiplication operator (*) could be evaluated first and its result used as the first operand to the addition (+) operator:

```
8 + 3
```

The result would be 11. Alternatively, the addition operator could be evaluated first and its result used as the second operand to the multiplication operator:

```
2 * 7
```

The result would be 14 instead. Clearly, rules that dictate how formulas are evaluated are needed to avoid confusion.

Informed uses three rules when it evaluates a complex formula. The precedence rules are based on:

- parentheses
- operator precedence
- left-to-right precedence

Before any operators are evaluated, Informed calculates all functions in a formula and uses the function result values as operands.

When you enclose part of a complex formula in parentheses, that part of the formula is evaluated first. If you nest parentheses by enclosing one set within another, the part of the formula in the "innermost" set of parentheses is evaluated first. The following example demonstrates how parentheses affect precedence.

`((3 + 5) * (6 - 3)) * 2`	innermost parentheses first
`(8 * 3) * 2`	outermost parentheses next
`24 * 2`	evaluate last operator
`48`	final result

Operator precedence is used when you don't include parentheses to explicitly indicate the order in which a formula's operators should be evaluated. Operator precedence specifies the order in which different operators in the same formula are evaluated. Operators with higher precedence are evaluated first. The following table shows Informed's operators grouped into precedence levels. The highest precedence operators come first.

Informed Operator Precedence Levels

Operators	Precedence
`% Not - (negation)`	highest
`^`	
`* / Div Mod`	
`+ - &`	
`= ≠ <> < ≤ ≤= > ≥ >=`	
`And`	
`Or`	lowest

The following example shows how operator precedence affects the order of evaluation of the operators in a formula.

```
4 * 5 + 3 < 4 - 6 / 3      evaluate multiplication and division first
20 + 3 < 4 - 2             evaluate addition and subtraction next
23 < 2                     evaluate comparison operator last
False                      final result
```

If you create a formula that uses operators of the same precedence level and no parentheses to explicitly alter the order of evaluation, the operators with the same precedence are evaluated left to right. The following examples shows how left-to-right precedence affects the evaluation of a formula.

```
5 * 2 Div 3 / 1.5          evaluate each operator from left to right
10 Div 3 / 1.5
3 / 1.5
2
```

When you enter a complex formula, you can include parentheses and any valid combination of operators and operands. The following example shows a complex formula and how Informed evaluates it using all precedence rules.

```
10 + 2 * ((5 + -2) / 3) - 5 = 8   unary negation first
10 + 2 * ((5 - 2) / 3) - 5 = 8    innermost parentheses
10 + 2 * (3 / 3) - 5 = 8          outermost parentheses
10 + 2 * 1 - 5 = 8                multiplication
10 + 2 - 5 = 8                    addition and subtraction
7 = 8                             comparison last
False                             final result
```

When operator or left-to-right precedence causes your formula to be evaluated incorrectly, you have to use parentheses to override the other precedence rules. For example, the formula below is intended to subtract from 50, the number of days between May 1, 1996 and April 25, 1996.

```
50 - ToDate ("May 1, 1996") - ToDate ("Apr 25, 1996")
```

However, left-to-right precedence causes the leftmost subtraction operator to be evaluated first (which attempts to subtract a date from a number). You could correct the formula either by switching the order of operands, or by using parentheses.

```
ToDate ("Apr 25, 1996") - ToDate ("May 1, 1996") + 50
50 - (ToDate ("May 1, 1996") - ToDate ("Apr 25, 1996"))
```

In either case, the subtraction operator subtracts one date from the other before adding the value 50. The correct result is returned.

Formula Result

Every formula returns a result of a particular type. A formula's result type depends on the operators, operands, and functions used in the formula.

The result type of a formula should match the type expected according to where the formula is used. For example, if a formula sets the value of a cell, the formula's result type should match the type of the cell, whereas a check formula should always return a boolean result.

If the result type of a formula is different than the type required, Informed will attempt to automatically convert the result value to the correct type. For example, if the calculation formula for a text cell returns a numeric value, Informed will automatically convert the number result to its textual representation when the formula is evaluated. If you want to explicitly change the result type of a formula from one type to another, use the type conversion functions (ToText, ToNumber, and so on). See "Type Conversion" for more information.

When you create a calculation, check, or tab formula, you can use the 'Return' statement to explicitly indicate the formula's return value. The 'Return' statement is optional; it allows you to create more readable formulas. For example, the two calculation formulas below are equivalent.

```
3 * Mean (Cell1, Cell2)
Return 3 * Mean (Cell1, Cell2)
```

The 'Return' statement is particularly useful with the 'If' statement. Using 'Return' makes it clear which parts of the 'If' statement are potential return values. The following 'If' statement uses several 'Return' statements.

```
If Age < 18 Then
    Return Price * 0.5
ElseIf (Age >= 18) AND (Age < 45) Then
    Return Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Return Price * 1.5
Else
    Return Price * 1.75
End
```

The 'Return' statements make it clear that the above formula can return one of four different results. Since the 'Return' statement is optional, the following formula is equivalent to the one above.

```
If Age < 18 Then
    Price * 0.5
ElseIf (Age >= 18) AND (Age < 45) Then
    Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Price * 1.5
Else
    Price * 1.75
End
```

Type Conversion

A value can be one of the following eight types: text, number, date, time, name, boolean, picture, and signature. The type of a value determines the type of information that the value can represent. For example, number values store numbers, whereas date values store dates. For a complete explanation of each type, see the “Cell Types” section in Chapter 1, “Adding Intelligence to Your Forms.”

Note When you use the Format command to select a cell’s type, you choose from nine different types. The discussion about type conversion involves only eight types. This is because the Text and Character cells both use the text type.

Type conversion refers to the process of changing a value from one type to another. Type conversion allows you to represent the same information in different forms. For example, you could convert a number value to a text value so that you could manipulate the value using Informed’s text functions.

Suppose that your form contains a name cell called ‘Customer Name’ and a text cell called ‘Comment.’ Let’s say that you want to create a calculation formula for the ‘Comment’ cell that combines the text message “Have a nice day” with the customer’s name. Since ‘Customer Name’ is a name cell, the cell’s value has to be converted to a text value before you can combine it with the text message.

Type conversion is required in the following situations:

- When the result type of a calculation formula doesn’t match the type of the cell that it sets.
- When the result type of a check formula is not boolean.
- When the result type of a tab formula is not text or number.
- When the type of an operand doesn’t match the type required by the operator with which it’s used.
- When the type of a function parameter doesn’t match the type expected by the function.
- When the result type of an ‘If’ or ‘ElseIf’ statement’s condition is not boolean.
- When you change the type of a cell on a form and use it to view old data.

You can explicitly change the type of a value using Informed's type conversion functions. For convenience reasons, however, Informed will automatically convert a value from one type to another in certain situations. The following section explains how Informed converts values.

Type compatibility

Informed can convert values only between certain types. For example, a number can be converted to a text value, but a date can't be converted to a number. If a particular type can be converted to another type, the two types are said to be *compatible*. The following table shows which types are compatible. The types labelled on those rows and columns that intersect with a check mark are compatible.

Compatible Types

Cell Type	text	number	name	date	time	boolean	picture	signature
from **text** to	√	√	√	√	√	√	√	√
from **number** to	√	√		√	√	√		
from **name** to	√		√					
from **date** to	√			√				
from **time** to	√				√			
from **boolean** to	√	√				√		
from **picture** to	√						√	
from **signature** to	√							√

All types are compatible with themselves and all types are compatible with the text type.

The fact that two types are compatible doesn't mean that all possible values will convert from one type to the other. For example, even though the text and number data types are compatible, not all text values will convert to numbers. Although text values such as '123,' '34.9328,' and '-15.50' will convert to their numeric equivalents, the values 'abc' and '123z' won't because they don't represent valid numbers.

Informed uses special rules when it converts values between the text and boolean types. The boolean values True and False are converted to the text values "True" and "False." The following table lists the text values that will convert to boolean values.

Converting Text Values to Boolean Values

Text Value	Converts To
"True"	True
"T"	True
"False"	False
"F"	False
"Yes"	True
"Y"	True

Converting Text Values to Boolean Values (continued)

Text Value	Converts To
“No”	False
“N”	False
“On”	True
“Off”	False

When a text value is compared with those values in the previous table, upper and lower case is ignored. This means, for example, that the text values “true” and “TRUE” will also be equivalent to the boolean value True.

When Informed converts a numeric value to a boolean value, the resulting boolean value will be True if the numeric value is non-zero, and False otherwise. The boolean values True and False convert respectively to the text values “True” and “False,” and to the numeric values 1 and 0.

When a date or time value is converted to a text value, the textual date or time value will format in a default manner. Date values will assume the format ‘M/D/YY,’ whereas time values will format according to ‘H:MM:SS AM.’ For more information about date and time formats, see the “Cell Types” section in Chapter 1, “Adding Intelligence to Your Forms.”

Both pictures and signatures are binary values. Binary values are values that don’t normally display as a series of characters. Informed understands how to display pictures and signatures in an appropriate representation (that is, a picture appears as a picture, whereas a signature displays as the signer’s name next to a signature icon). When Informed converts a picture or signature to text, it converts the binary information to a series of characters. It is useful to be able to convert pictures and signatures to text (and back) if you want to export or submit form data to a database, or through a medium that does not support binary information.

If you attempt to convert a value between two incompatible types, or if the conversion of a value between two compatible types fails, Informed will change the value to the empty value. For example, suppose that the cell called ‘Purchase Date’ is a date cell. The result of each formula below would be the empty value.

```
5 * Purchase Date
15 / ‘ABCDEFG’
```

In the first formula, the type of the cell ‘Purchase Date’ is incompatible with the multiplication operator. In the second formula, the text value “ABCDEFG” can’t be converted to a valid number.

Automatic Type Conversion

For convenience reasons, Informed will automatically convert a value from one type to another when it’s clear that the value must be a particular type. For example, if the result of a calculation formula doesn’t match the type of the cell that it sets, Informed will automatically convert the value to the correct type. The type conversion rules explained in the previous section apply as expected.

Automatic type conversion allows you to create formulas with more freedom. You don't have to tediously convert values from type to type in most common situations. For example, both formulas below return the numeric result 150.

```
3 * ToNumber ("50")
3 * "50"
```

Since the multiplication operator multiplies numbers only, Informed will automatically convert both operand values—if necessary—to numbers.

Suppose that a date cell called 'Ship Date' contains the value September 12, 1996. The formula below uses the concatenation operator to combine the value of this cell with a text message.

```
"Your order will ship on " & Ship Date & "."
```

The value of 'Ship Date' is automatically converted to the text value "9/12/96" and then concatenated to produce the result "Your order will ship on 9/12/96."

Some operators and function parameters allow values of different types. For example, the subtraction operator can subtract numbers, dates, or times, and the 'Choose' function can select a value of any type. To ensure that your formulas calculate correct results in these situations, Informed allows you to use special type conversion functions. These functions are explained in the following section.

Type Conversion Functions

When you create a formula, you can explicitly convert a value from one type to another. Informed provides eight type conversion functions for converting values between compatible types. The following table lists these functions.

Type Conversion Functions

Function	Description
ToText	Converts any text-compatible value to text
ToNumber	Converts any number-compatible value to a number
ToName	Converts any name-compatible value to a name
ToDate	Converts any date-compatible value to a date
ToTime	Converts any time-compatible value to a time value
ToBoolean	Converts any boolean-compatible value to a boolean value
ToPicture	Converts any ASCII represented picture to a picture value
ToSignature	Converts any ASCII represented signature to a signature value

The single parameter to any of these functions can be any value that's compatible with the destination type. For example, the formula below converts a text value to the date value April 5, 1996.

```
ToDate ("April 5, 1996")
```

You should use the type conversion functions when the type of a value can be interpreted differently. Some operators and function parameters allow values of different types. For example, the comparison operators compare values of any type, as long as the type of each operand is the same. If you want to compare two differently typed values according to the comparison rules of a particular type, you should use the type conversion functions to convert both operands—if necessary—to the correct type. The formula below compares the value of the time cell ‘Time In’ with the time constant 8:00 AM.

```
If Time In <= ToTime ("8:00 AM") Then
   Return True
Else
   Return False
End
```

If you were to compare the value in ‘Time In’ with the text constant “8:00 AM” instead, Informed would convert the value in ‘Time In’ to text and then compare two text values. By using the ToTime function, both operands are interpreted as time values. Informed therefore uses the rules for comparing time values to compare the two operands.