

PROGRAMMING



quicksheets

Conditional If Statements

This QuickSheet provides examples of using the conditional **if** statement in Mathcad programs then compares it with the inline **if** function. Conditional switches are useful for piecewise definitions, determining loop control, or trapping errors in input arguments.

These programs define functions in a piecewise fashion:

$$fp(x) := \begin{cases} 1 & \text{if } \text{mod}(\text{ceil}(x), 2) = 1 \\ -1 & \text{otherwise} \end{cases}$$

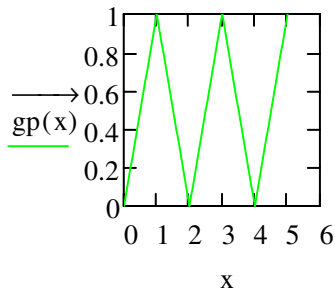
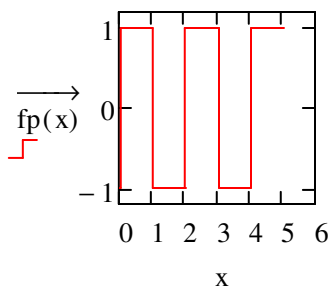
$$gp(x) := \begin{cases} x - \text{floor}(x) & \text{if } \text{mod}(\text{ceil}(x), 2) = 1 \\ 1 & \text{if } \text{floor}(x) = x \wedge \text{mod}(\text{ceil}(x), 2) = 1 \\ -x + \text{ceil}(x) & \text{otherwise} \end{cases}$$

Note: You must use the **if** operator from the Programming toolbar in programs; do not just type "if."

These simple programs are equivalent to the command-line **if** function, although they may be easier to read.

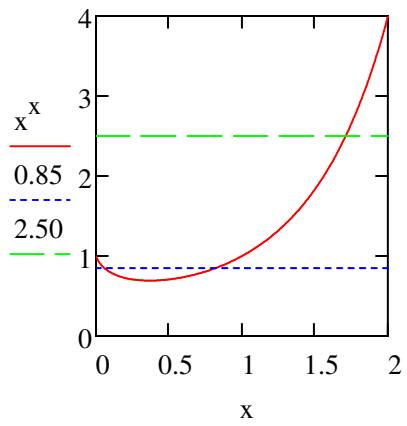
$$fil(x) := \text{if}(\text{mod}(\text{ceil}(x), 2) = 1, 1, -1)$$

$$i := 0..100 \quad x_i := i \cdot 0.05$$



This program allows different solutions to an equation for various parameter regimes.

$x := 0, 0.01 \dots 2$



```

invrsfct(y) := if 0 < y < 1
                | z ← 0.1
                | a ← root(zz - y, z)
                | z ← 0.9
                | b ← root(zz - y, z)
                | return (a b)
                | otherwise
                | z ← 2
                | c ← root(zz - y, z)
                | return c

```

U1 := invrsfct(0.85) V1 := invrsfct(2.5)

U1 = (0.05658830 0.82026299) V1 = 1.70927557

→

U1^{U1} = (0.85 0.85) V1^{V1} = 2.5

This program implements a variable rate structure, using external variables to control the conditional expressions:

rate1 := 0.069 break1 := 240

rate2 := 0.12 break2 := 540

rate3 := 0.14

```

bill(usage) := | if usage < break1
                | "lowest bracket"
                | return rate1 · usage
                | if usage < break2
                | "middle bracket"
                | return rate2 · usage
                | otherwise
                | "highest bracket"
                | return rate3 · usage

```

Note: the use of strings by themselves on lines as comments in the code.

u := 600

bill(u) = 84

The otherwise statement above only executes if both **if** statements are false. However, if several sequential **if** statements in a program are true, they all execute, even if the block ends with an "otherwise." For example:

x := 4

```

| y ← 0 = 5
| y ← y + 3 if x > 3
| y ← y + 2 if x > 2
| y ← y + 1 otherwise
| y

```

Units in Conditional Programs

It is not possible to execute a conditional program that returns different units depending on an input value; Mathcad returns an error when you try to execute such a program, as a program cannot be unit balanced without knowing the particular value of the inputs. The expression below can only be executed for dimensionless inputs.

$$\text{pow}(x, n) := \begin{cases} x^2 & \text{if } n < 2 \\ x^3 & \text{otherwise} \end{cases}$$

These work:

$$\text{pow}(2.5, 1) = 6.25 \quad \text{pow}(2.5, 3) = 15.625$$

These return an error:

$$\text{pow}(2.5 \cdot m, 1) = \blacksquare \quad \text{pow}(2.5 \cdot m, 3) = \blacksquare$$

Closing Conditional Blocks with Otherwise

The previous programs, which have used conditional tests to vary their results, have used **otherwise** statements as well. The use of otherwise is important, as it prevents the program from returning unpredictable or incorrect results. Take the following program:

$$f(x) := \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$$

In most cases, the program returns the expected result:

$$f(4) = 1 \quad f(\pi) = 1$$

$$f(-2) = -1 \quad f(-e) = -1$$

But if you try to evaluate the function at 0, it returns an error.

$$f(0) = \blacksquare$$

In this simple case, no definition exists for $x = 0$. The easiest solution is to initially define the return value:

$$f(x) := \begin{cases} \text{res} \leftarrow 0 \\ \text{res} \leftarrow -1 \text{ if } x < 0 \\ \text{res} \leftarrow 1 \text{ if } x > 1 \end{cases}$$

$$f(0) = 0$$

When you create more complex programs, however — especially ones involving units — you should close conditional blocks with an **otherwise**. Use of the **otherwise** operator guarantees that one of the steps in the conditional block executes.

Consider the more complex case:

$$\text{prog}(\text{Area}, \text{Length}, \text{adj}) := \begin{cases} \text{Vol} \leftarrow \text{Area} \cdot \text{Length} \\ \text{factor} \leftarrow \begin{cases} 2.5 \text{ if } \text{adj} \geq 3 \\ 2.1 \text{ if } \text{adj} = 2 \\ 1.7 \text{ if } \text{adj} < 2 \end{cases} \\ \text{Vol} \cdot \text{factor} \end{cases}$$

$$\text{prog}(2\text{m}^2, 4\text{cm}, 2) = 0.168\text{m}^3$$

In this case, the problem isn't so obvious. After all, a value of **adj** is provided that satisfies one of the tests, so there doesn't *appear* to be problem.

In this case, the problem arises from with Mathcad's static unit checking, which balances the equation before calculation to ensure consistent results. Because the definition of **factor** could fail for certain values of **adj** (for example, **adj** = 2.5), the previous result — in this case, **Vol** — "falls through" into the definition of **factor**. As a result, there is a scenario in which the program might return a result of m^6 , which doesn't agree with the result if **adj** = 2, which would be m^3 .

To draw attention to this problem, Mathcad flags the last argument in the definition prior to the problematic if block, and assumes that its units should be changed in order to force the expected result; changing **Length** from 4cm to 4cm^{-2} would return the expected result of m^3 .

Using an otherwise prevents the result prior to the if block from affecting the definition of factor:

$$\text{prog}(\text{Area}, \text{Length}, \text{adj}) := \left\{ \begin{array}{l} \text{Vol} \leftarrow \text{Area} \cdot \text{Length} \\ \text{factor} \leftarrow \left\{ \begin{array}{l} 2.5 \text{ if } \text{adj} \geq 3 \\ 1.7 \text{ if } \text{adj} < 2 \\ 2.1 \text{ otherwise} \end{array} \right. \\ \text{Vol} \cdot \text{factor} \end{array} \right.$$

$$\text{prog}(2\text{m}^2, 4\text{cm}, 2) = 0.168\text{m}^3$$
